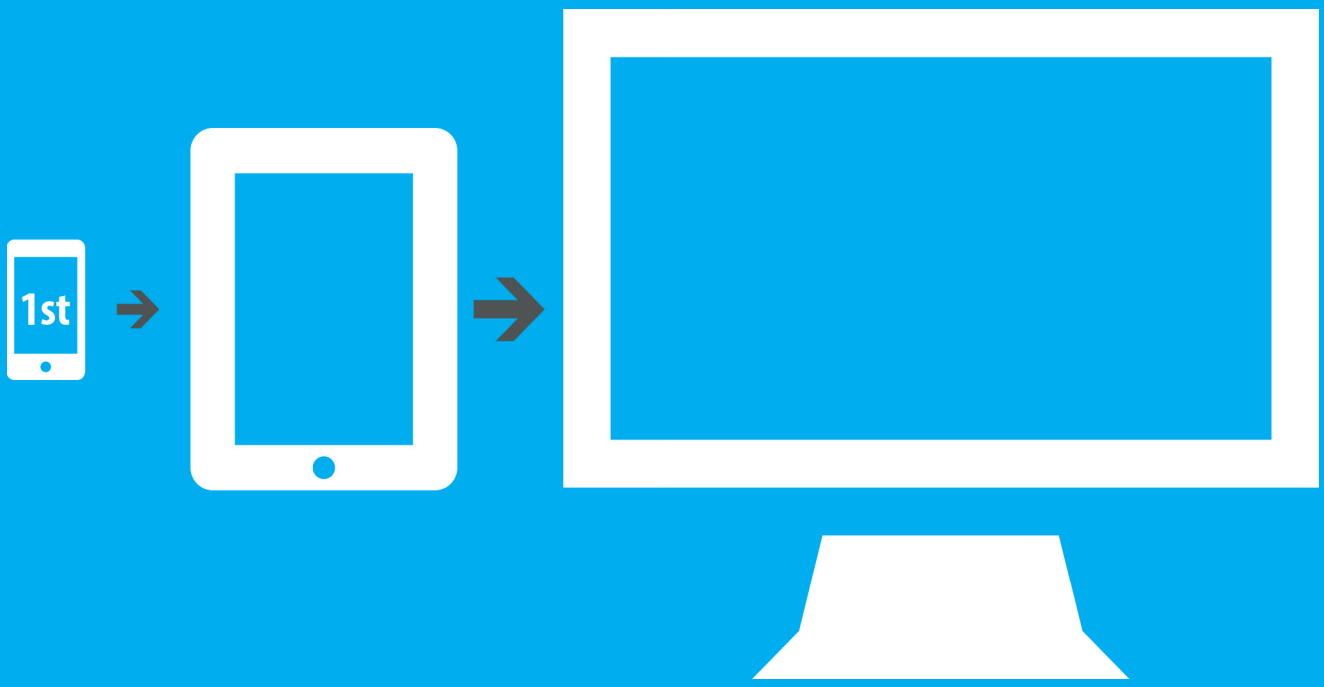
BACHELORARBEIT
IM STUDIENGANG AUDIOVISUELLE MEDIEN



MOBILE FIRST RESPONSIVE WEBDESIGN

Eine Untersuchung der Grundlagen, der gestalterischen und technischen Umsetzung sowie eine Analyse des Stands der Technik.

Vorgelegt von:
Florian Speckmaier
Matrikelnr. 23139
an der Hochschule der Medien Stuttgart
am 31.07.2013

Erstprüfer:
Dipl.-Ing. Uwe Schulz

Zweitprüfer:
Dipl.-Ing. (FH) Thomas Reimann

Eidesstattliche Erklärung

Hiermit versichere ich, Florian Speckmaier an Eides statt, dass ich die vorliegende Bachelorarbeit mit dem Titel

„Mobile First Responsive Webdesign - Eine Untersuchung der Grundlagen, der gestalterischen und technischen Umsetzung sowie eine Analyse des Stands der Technik“

selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der eidesstattlichen Versicherung und die prüfungsrechtlichen Folgen (§ 26 Abs. 2 Bachelor-SPO bzw. § 19 Abs. 2 Master-SPO der Hochschule der Medien Stuttgart) sowie die strafrechtlichen Folgen (§ 156 StGB) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.

Stuttgart, 31.07.2013

Ort, Datum

Unterschrift

Kurzfassung

Die vorliegende Bachelorarbeit untersucht die Grundlagen sowie die gestalterische und technische Umsetzung des Ansatzes „Mobile First Responsive Webdesign“ zur modernen, mobil optimierten Webentwicklung und analysiert den aktuellen Stand der Technik.

Die Gestaltung und Optimierung von Inhalten für eine Vielzahl verschiedener Auflösungen und Bildschirmgrößen rückt durch die steigende Verbreitung von Mobilgeräten wie Smartphones und Tablets immer mehr in den Fokus der Webdesigner- und Entwickler und sollte daher bei der Gestaltung von Websites zwingend oberste Priorität besitzen.

Das Ziel dieser Arbeit ist es, die Grundlagen und Herangehensweisen, welche dieser Gestaltungsansatz impliziert, aufzuzeigen und näher zu beleuchten. Besonderes Augenmerk wird auf die Unterschiede zum bisherigen Vorgehen bei der Umsetzung von Websites gelegt.

Dabei werden verwendete Begrifflichkeiten beleuchtet, Gründe für die Realisierung von Websites mit vorgestellten Ansatz erarbeitet und auf die gestalterische und technische sowie organisatorische Umsetzung des selbigen eingegangen.

Außerdem wird der aktuelle Stand der Technik analysiert und eventuell vorhandene Probleme der momentan verfügbaren Umsetzungen diskutiert. Ein Ausblick auf zu erwartende, verbesserte Implementierungen ist ebenso Bestandteil dieser Arbeit.

Schlagwörter

Mobile First, Responsive Webdesign, Webentwicklung, HTML5, CSS, Java Script, Programmierung, mobile Internetseiten, Optimierung für mobile Geräte, Webdesign

Abstract

This thesis examines the essentials as well as the creative and technical realisation of the approach to modern, mobil optimized webdesign- and development called „Mobile First Responsive Webdesign“ and analyses its state-of-the-art.

The design and optimization of mobile contents for a vast amount of screensizes gets increasingly focused by webdesigners- and devlopers due to the spreading of mobile devices and should therefore obtain highest priority when it comes to new development of websites.

The aim of this thesis is to highlight the essentials and strategies which are implied within the approach „Mobile First Responsive Webdesign“. Special attention is payed to the differences between the existing approach to realise websites.

Therefore, the used terms are defined and reasons for the implementations of websites with the new approach are given. Furthermore, the creative and technical, as well as the organisational ways of realisation are discussed.

Additionally, the state-of-the-art together with possible problems within the available realisations is analyzed. A forecast to future and improved implementations is also part of this thesis.

Keywords

Mobile First, Responsive Webdesign, Webdevelopment, Webdesign, HTML5, CSS, JavaScript, Programming, Mobile webpages, Optimization for mobile devices

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AC	Adaptive Content
API	Application Programming Interface (deutsch: Programmierschnittstelle)
AWD	Adaptive Webdesign
CSS	Cascading Stylesheets
CMS	Content Management System
HTML	Hypertext Markup Language
JS	JavaScript
ME	Mobile Endgeräte; hauptsächlich Smartphones, aber auch Tablets
MF	Mobile First
MFRWD	Mobile First Responsive Webdesign
MQ	Media Queries
RWD	Responsive Webdesign
UE	User Experience (deutsch: Benutzer-Erlebnis)
WE	Web Experience (deutsch: Web-Erlebnis)

1. Einleitung

1.1. Überblick

„Why join the navy, if you can be a pirate“ [1] - Wieso sich der Marine anschließen, wenn man auch ein Pirat sein kann. Dass es sich lohnt gegen den Strom zu schwimmen, anders zu denken und innovative Piraterie ganze Branchen revolutionieren kann, zeigte der 09.01.2007. An diesem Tag stellte Apple in Person von Steve Jobs das iPhone vor. Dieses Datum markiert einen Meilenstein für das mobile Internet, wie wir es kennen und ist gewissermaßen als dessen Geburtsstunde zu bezeichnen.

Davor bestand das mobile Web aus Textlinks, WAP 2.0/XHTML Browsern und bot aufgrund der langen Ladezeiten – UMTS wurde zu diesem Zeitpunkt kaum genutzt[2] – und der begrenzten Rechenleistung damals gängiger Geräte, wie beispielsweise dem Motorola Z3[3], kaum über Mehrwert.

Das iPhone verfügte über den fortschrittlichen Web-Browser Safari und eine, im Vergleich zu zeitgenössischen Geräten, hohe Rechenleistung[4]. Es war damit in der Lage, Internetseiten in ähnlicher Darstellungsqualität wie auf dem Desktop anzuzeigen. Plötzlich war mobiles Surfen kein quälender und zeitraubender Vorgang mehr, eine Revolution hatte ihren Lauf genommen.

Als im Oktober 2008 das HTC Dream und damit das Google-Betriebssystem Android als alternative zum iPhone und dessen iOS auf den Markt kam, die Entwicklung weiter fortschritt, wurden Smartphones für eine breite Masse erschwinglich.

Dazu kam seit der Einführung des iPads von Apple sowie diverser Android-Tablets unterschiedlicher Hersteller, die wachsende Verbreitung von Tablet-Computern.

Wohin man seinen Blick heute auch richtet, Mobilgeräte sind omnipräsent: Egal ob in der U-Bahn, im Café oder auf der heimischen Couch: Menschen informieren sich, kaufen ein oder lassen sich unterhalten. Das mobile Internet hat längst Einzug in unseren Alltag gehalten.

Jedoch ist die Übertragung der gewohnten User Experience (UE) einer herkömmlichen, also für stationäre Desktoprechner wie PC oder Laptop, entwickelten Website auf das mobile Endgerät nicht ohne Weiteres möglich, da diese grundlegende technische Unterschiede aufweisen. Daher richtet sich der Fokus von Webdesignern- und Entwicklern immer stärker auf die Gestaltung und Programmierung von Websites, die auch auf mobilen Endgeräte optimal dargestellt werden.

---

1 Steve Jobs, auf einer Klausurtagung im September 1982
2 10 Jahre UMTS-Versteigerung. http://www.bitkom.org/de/themen/54894_64598.aspx (Stand 01.06.2013)
3 Motorola Z3 WAP & JAVA Internet browsing. http://www.youtube.com/watch?v=8aaOtVJQcg0 (Stand 01.06.2013)
4 Spezifikationen des Apple iPhones Original. http://de.wikipedia.org/wiki/Apple_iPhone_(original) (Stand 01.06.2013)

1.2. Ziel der Bachelorarbeit

Im Laufe der Zeit haben sich verschiedene Ansätze und Technologien zur optimierten Darstellung von Websites auf Mobilgeräten entwickelt, also wie diese für ein mobiles Endgerät aufbereitet werden können.

Der Ansatz „Mobile First Responsive Webdesign“ stellt die Entwicklung für Mobilgeräte in den Fokus, ohne dabei Darstellung und Funktionalität auf herkömmlichen – stationären Geräten – aus den Augen zu verlieren. Die entstehenden Websites passen sich dabei dynamisch den jeweiligen Anzeigegeräten an, weshalb nur eine einzige Codebasis entwickelt werden muss.

Diese Bachelorarbeit untersucht diesen Ansatz sowie die gestalterischen, technischen und organisatorischen Herangehensweisen, die damit verbunden sind. Diese werden aufgezeigt und näher beleuchtet. Zudem sollen Gründe für die Umsetzung des Ansatzes erarbeitet und dessen Begrifflichkeiten und Denkweisen dargelegt werden.

Besonderes Augenmerk wird auf die Unterschiede zur bisherigen Realisierung von Websites gelegt und inwiefern ein Umdenken auf gestalterischer sowie technischer Ebene von Nöten ist.

Dabei fließen konkrete Programmierbeispiele neben Codeausschnitte und Empfehlungen aus meiner eigenen persönlichen Erfahrung ebenso mit ein, wie auch der aktuelle Stand der Technik innerhalb der Entwicklergemeinde.

Dieser soll zudem analysiert und eventuell vorhandene Probleme momentan verfügbarer Umsetzungen diskutiert werden. Des Weiteren wird ein Ausblick auf zu erwartende, verbesserte Implementierung gegeben.

Ziel dieser Arbeit ist es, dem Leser die Notwendigkeit zur Umsetzung des vorgestellten Ansatzes zu vermitteln, neue Impulse zu liefern und ihn in die Lage zu versetzen, bei zukünftigen Projekten bisherige Herangehensweisen kritisch zu betrachten. Die Vorteile von Mobile First Responsive Webdesign (MFRWD) sollen ersichtlich werden und dazu motivieren, in Zukunft mobil optimierte, reaktionsfähige Websites zu entwickeln.

1.3. Abgrenzung der Arbeit und Definition der Zielgruppe

Die vorliegende Arbeit behandelt ausschließlich den Ansatz „Mobile First Responsive Webdesign“ und somit die Realisierung von Websites mittels HTML5, CSS3 und JavaScript. Explizit außen vor bleiben daher die Technologien nativer oder hybrider Anwendungen sowie Web-Apps und die Entwicklung von eigenständigen mobilen Websites.

Die Leserschaft der Thesis sollte über fundierte Grundkenntnisse in Sachen Webdesign- und Entwicklung – also in Technologien wie HTML, CSS und JavaScript – verfügen und bereits eigene

Webprojekte realisiert haben, um den vorgestellten Techniken und Ideen folgen zu können.

Die Arbeit richtet sich also primär an professionelle Webentwickler- und Gestalter oder Leser, die über einen informationstechnischen oder gestalterischen Hintergrund verfügen.

Außerdem hat die Thesis keinen Anspruch auf absolute Vollständigkeit hinsichtlich der vorgestellten Inhalte, Techniken und Tools. Um jeden Teilbereich der Entwicklung mit MFRWD komplett abzudecken, ist der Umfang nicht ausreichend. Die nachfolgenden Inhalte wurden nach bestem Gewissen ausgewählt. Der Stand der Technik beschränkt sich +/- zwölf Monate in Bezug auf das Abgabedatum dieser Thesis, da womöglich zu einem späteren Zeitpunkt – auf Grund der rasanten Entwicklung und der großen Entwicklergemeinde – bereits neue Techniken die hier dargelegten überholt haben könnten.

1.4. Motivation für diese Arbeit

Bereits seit meiner ersten Ausbildung zum „Technischen Assistenten für Informatik“ interessiere ich mich für die Konzeption und Realisierung von Websites, Online- und Interaktiver Medien. Diese Leidenschaft erweiterte sich nach Erwerb der Fachhochschulreife in der anschließenden Ausbildung zum „Mediengestalter digital“, die ich bei der G+P Glanzer und Partner Werbeagentur in Stuttgart absolvierte.

Während des Studiums der Audiovisuellen Medien an der HdM Stuttgart konnte ich dort weiterhin im Rahmen einer studentischer Mitarbeit, Teil des Teams „Webunit - Neue Medien“ bleiben und somit an zahlreichen Projekten mitwirken, die den gewachsenen Anforderungen des WWWs Rechnung tragen mussten.

Ich glaube an die Wichtigkeit des mobilen Webs und dass es eine der großen Herausforderungen unserer Zeit ist, Informationen schnell und optimal aufbereitet zur Verfügung zu stellen. Viele Aufgaben, die vor kurzem noch offline zu erledigen waren – angefangen beim Zeitunglesen bis hin zum Schreiben von Einkaufszetteln – verlagerten sich erst in das stationäre, dann in das mobile Internet. Dabei gelang die Transformation hin zu sinnvollen Anwendungen mit Mehrwert nicht immer. Viel zu oft stößt man auf schlecht gestaltete Informationsstrukturen, mangelnde Usability und unzureichend implementierte technische Lösungen. Ich finde es extrem spannend den Anforderungen, die durch die Vielfältigkeit des Internets erwachsen, zu begegnen und es besser als die zahlreichen Negativbeispiele zu machen.

Um es bildlich zu formulieren: Ich atme das Web und alles was sich in gestalterischer und technischer Hinsicht darin befindet. Meine Leidenschaft begründet die Wahl des Themas vorliegender Arbeit.

2. Begriffsdefinitionen und Denkansätze

In den folgenden Abschnitten werden die verwendeten Begrifflichkeiten „Mobile First", „Responsive Webdesign" sowie die Verschmelzung beider Begriffe zu dem Terminus „Mobile First Responsive Webdesign" und die damit einhergehenden Denkansätze definiert und erläutert.

2.1. Mobile First

Bei „Mobile First" (MF), handelt es sich um einen Denk- und Gestaltungsansatz, der vorschlägt, bei der Entwicklung von Websites zuerst für mobile Endgeräte zu gestalten und zu programmieren, diese also in den Fokus der Aufmerksamkeit der Gestalter und Entwickler zu stellen. Der Begriff „Mobile First" wurde durch den Webdesigner- und Entwickler Luke Wroblewski geprägt[5].

Jahrelang, seit Anbeginn des Internets, wurden Websites für die Anzeige auf Desktoprechner wie PCs oder Laptops, also für hohe Bildschirmauflösungen gestaltet und optimiert. Dieses Vorgehen war solange richtig, bis sich das mobile Internet etablierte und das Verlangen nach mobilen Websites und die Anforderungen an diese wuchsen.

Es entwickelten sich mehrere Ansätze[6] um Websites für mobile Geräte zu gestalten oder Inhalte aufzubereiten, darunter die Entwicklung separater mobiler Websites, Web-Apps[7] oder nativer Anwendungen.

Als weitere Strategie ist die Anpassung bzw. Verkleinerung einer originär, also eine für große Bildschirmauflösungen, gestalteten Website zu einer mobil optimierten Anzeige zu nennen, welche auch auf – die im *Abschnitt 2.2* erläuterte – Technik des Responsive Webdesigns (RWD) zurückgreift.

MF dreht dieses Vorgehen um. Ausgehend von einer mobil optimierten Website wird hin zu einer Desktop-Version gestaltet und entwickelt, somit die Darstellung auf mobilen Endgeräte präferiert. Daher wird Inhalt, Layout bzw. Screendesign sowie Programmierung zunächst und primär für die Anzeige auf mobilen Geräten optimiert.

Worin die Gründe für dieses Vorgehen liegen, wird in *Abschnitt 3* genauer beleuchtet.

5 Wroblewski, Luke. Mobile First. New York: A book Apart 2011. S. 1
6 Maurice, Florence. Mobile Websiten. Strategien, Techniken, Dos und Don'ts für Webentwickler. München: Carl Hanser Verlag 2012. S. 17 - 29
7 Remick, Jarel. What is a Web App? Here is our definition. http://web.appstorm.net/general/opinion/what-is-a-web-app-heres-our-definition/ (Stand 02.06.2013)

2.2. Responsive Webdesign

Der Begriff „Responsive Webdesign" – zu deutsch reaktionsfähiges Webdesign – wurde im Jahre 2010 von Ethan Marcotte in dessen gleichnamigen Artikel, der in dem bekannten Online-Webdesignmagazin „A list a part" erschien, geprägt[8].

Marcotte adaptiert dabei eine Idee aus der Architektur, die sogenannte „Responsive Architecture", nach welcher Gebäude und physische Räume auf Menschen, die sie betreten, reagieren und sich flexibel anpassen. So experimentieren die Architekten beispielsweise mit sich beugenden Wänden oder Glasscheiben, die blickdicht werden, wenn sich eine bestimmte Zahl an Menschen in einem Raum befinden, um diesen mehr Privatsphäre zu ermöglichen[9].

Diese reaktionsfähige Architektur löst sich also aus ihrer Starre, wenn gewisse Veränderungen in ihrem Umfeld eintreten. Diesen Ansatz überträgt Marcotte auf Websites, da durch das rapide Wachstum des mobilen Internets auch für Onlinemedien die Notwendigkeit zur dynamischen Anpassung, also zu reaktionsfähigem Verhalten immer wichtiger wird: Räume und äußere Begebenheiten – also Bildschirmgrößen und Anforderungen – ändern sich, betrachtet man Websites auf unterschiedlichen Geräten.

Nach seinem Ansatz „Responsive Webdesign" soll sich eine Website also flexibel an die Auflösung anpassen, Elemente wenn nötig anders positionieren und sich nicht durch ein pixelperfektes Layout, also einer 1:1 Umsetzung der Gestaltungsvorlage, limitieren lassen. Dieses Vorgehen hatte sich ausgehend von der Printindustrie, in der feste Raster und Definitionen vorherrschen, mit dem Wunsch nach mehr gestalterischer Kontrolle auch auf das Webdesign übertragen[10].

Mit RWD lässt sich eine Website also mit gleicher Code- und Gestaltungsbasis für die Anzeige auf verschiedenen Geräten – herkömmliche Computerbildschirme in verschiedentlicher Auflösung, Smartphones oder Tabletcomputer – anpassen, um eine optimale gerätespezifische Darstellung zu gewährleisten.

Das Vorgehen das Marcotte vorschlägt, ist eigentlich eine Rückbesinnung auf das, was Websites und HTML-Dokumente schon immer konnten: Mit prozentuale und flexible Größen, egal ob Flächen, Inhaltsbereiche oder Schriften, zu arbeiten[11].

Daher verhält sich eine mit RWD gestaltete Website in ihrer Größenänderung dynamisch, das heißt Spalten, Inhaltselemente oder Bilder vergrößern sich fließend für jede erdenkliche Auflösung und nicht nur sprunghaft, wie das beim artverwandten „Adaptive Webdesign" (AWD) der Fall ist[12].

8 Marcotte, Ethan. Responsive Webdesign. http://alistapart.com/article/responsive-web-design (Stand 02.06.2013)
9 Ebenda.
10 Zilligens, Christoph. Responsive Webdesign. München: Carl Hanser Verlag 2013. S. 8
11 Ebenda.
12 Hellwig, Jonas. Adaptive Website vs. Responsive Website. http://blog.kulturbanause.de/2012/11/adaptive-website-vs-responsive-website/ (Stand 02.06.2013)

Das AWD stellt somit eher die Optimierung für bestimmte Viewports, also Bildschirmgrößen in den Vordergrund, wohingegen RWD versucht, alle Auflösungen gleichermaßen flexibel und dynamisch zu bedienen.

Jedoch sind die Grenzen hierbei fließend, weil auch das RWD mit den sogenannten Breakpoints – also den Grenzen, die den Übergang von verschiedenen Geräteauflösungen definieren – arbeitet, wie im Laufe dieser Arbeit noch genauer beleuchtet wird.

Zusammenfassend lässt sich also sagen, dass es sich bei RWD zum einen, wie auch bei MF, um ein Gestaltungsparadigma handelt. Zum anderen geht damit aber auch eine Vielzahl von verschiedenen Techniken einher, die es den Entwicklern ermöglichen, diesen Ansatz auch konkret technisch zu realisieren. Die technische Umsetzung wird im *Abschnitt 4* detailliert behandelt.

Die nachfolgenden Abbildungen illustrieren das Prinzip des RWDs noch einmal bildlich:

Abbildung 1 - Prinzip des Responsive Webdesigns

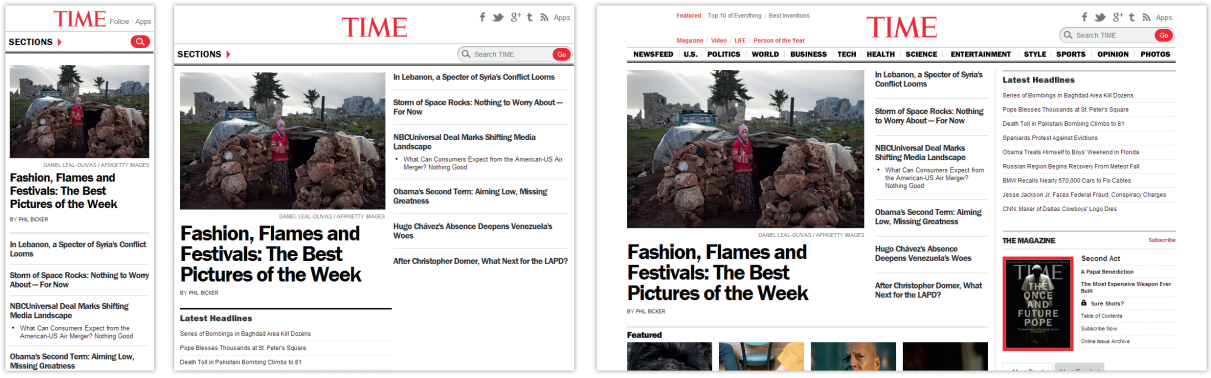


Abbildung 2 - Responsive Design des Nachrichtenmagazins Time

2.3. Mobile First Responsive Webdesign

„Mobile First Responsive Webdesign“[13] stellt also eine Verschmelzung der obig beleuchteten Ansätze dar, um eine Website zuerst für mobile Endgeräte optimal aufzubereiten und auf deren Anforderungen und Spezifikationen einzugehen. Dabei soll die Darstellung auf größeren Auflösungen stationärer Desktoprechner aber nicht aus den Augen gelassen werden.

Der weit verbreitete Ansatz, eine Website zunächst für große Auflösungen zu gestalten wird, wie bereits in *Abschnitt 2.1* erwähnt, umgedreht:

- Der normale Inhalt und der grundsätzliche Aufbau der Website ist mobil optimiert. Dies bezeichnet mal als „Basisunterstützung.“[14]
- Auf Geräten mit größerem Bildschirm und erweiterter Unterstützung der eingesetzten Technologien wird ein verbessertes Layout angezeigt, es ist vom „Mobilen Kontext[15]“ die Rede.
- Zu guter Letzt wird eine Desktop-Variante der Website dargestellt, wenn die Auflösung des verwendeten Geräts dies zulässt. Hierbei handelt es sich um den „Desktop Kontext“[16]

Man spricht von Progressive Enhancement[17], also progressiver Verbesserung. Funktionalitäten werden, wenn möglich hinzugefügt. Das herkömmliche Verfahren verfolgte den Ansatz der Graceful Degration[18], in welchem Funktionalitäten einfach weggelassen wurden. Somit wurde die Website in verminderter Qualität ausgeliefert.

Die progressive Verbesserung gilt vielen Webentwicklern als das zu bevorzugende Paradigma, da es die Basis jeder Website – den Inhalt – in den Mittelpunkt stellt, was zu einem besseren Weberlebnis führt[19].

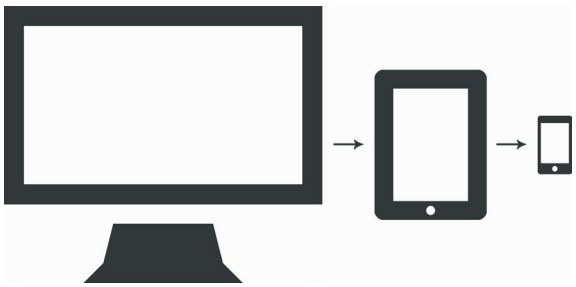


Abbildung 3 - Graceful Degration

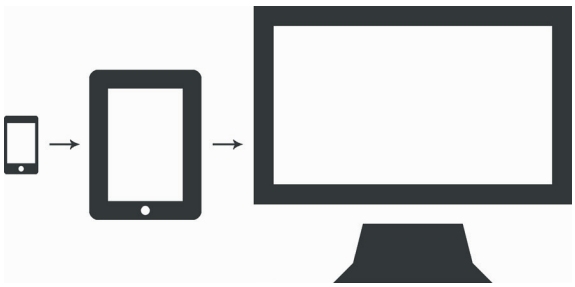


Abbildung 4 - Progressive Enhancement

13 Rieger, Bryan. Website zur Demonstration von Mobile First Responsive Webdesign. http://yiibu.com/about/site/index.html (Stand 03.06.2013)
14 Ebenda.
15 Ebenda.
16 Ebenda.
17 Gustafson, Aaron. Understanding Progressive Enhancement. http://alistapart.com/article/understandingprogressiveenhancement (Stand 03.06.2013)
18 Borowska, Paula. Graceful Degration vs. Progressive Enhancement. http://designsuperstars.net/graceful-degradation-versus-progressive-enhancement-in-web-design-who-will-win/ (Stand 03.06.2013)
19 Gustafson, Aaron.

3. Gründe für Mobile First Responsive Webdesign

„The simple guideline is, whatever you are doing – do mobile first“[20] – Egal was du machst, mach es zuerst mobil. Das Zitat von Google CEO Eric Schmidt unterstreicht die Bedeutung der Mobilfähigkeit von Websites- bzw. Anwendungen.

Im Nachfolgenden sollen nun die drei Hauptgründe – Mobiles Wachstum, Einschränkungen als Chance, Erweiterung der Web Experience (WE) durch das Mobilgerät – für die Umsetzung des Ansatzes MFRWD näher beleuchten werden.

3.1. Mobiles Wachstum

Wohin unser Blick im urbanen Raum auch schweift: Smartphones oder Tablets sind omnipräsent. Egal ob im Café, der Straßenbahn oder in der Fußgängerzone. Menschen unterschiedlichster Alters-, Interessen oder Berufsgruppen informieren sich mit ihren Geräten in der Hand mittels mobiler Webangebote, unterhalten sich mit Spielen, schreiben E-Mails und Nachrichten oder tummeln sich in sozialen Netzwerken. So gaben 50% der Befragten einer Studie von Google zur Nutzung von mobilen Endgeräten (ME) an, in den letzten sieben Tagen ihre Geräte täglich benutzt zu haben, 64% davon gehen nicht ohne Smartphone außer Haus[21].

Dieses Phänomen macht auch vor den eigenen vier Wänden nicht halt. So nutzen immer mehr Menschen ihr mobiles Gerät auch zu Hause, was auch nachfolgendes Diagramm[22] zeigt:

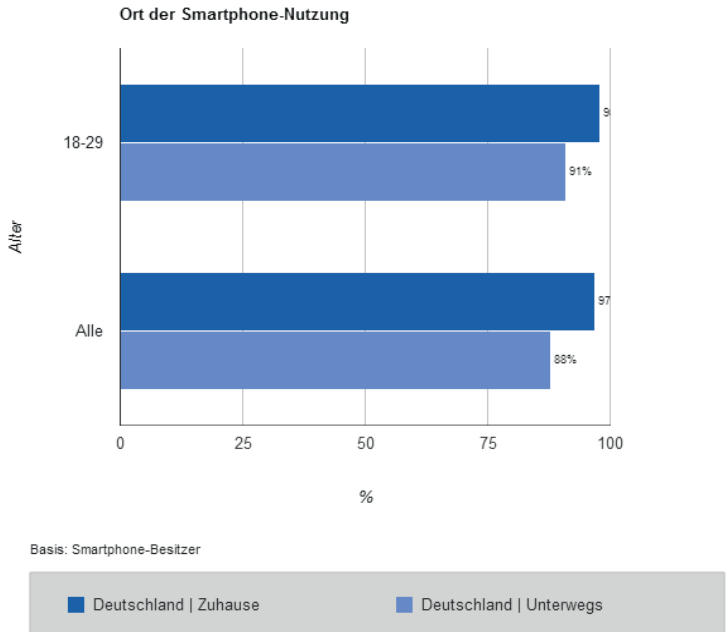


Abbildung 5 - Balkendiagramm Ort der Smartphone-Nutzung

20 Wroblewski, Luke. Google CEO Eric Schmidt on Mobile First. http://www.lukew.com/ff/entry.asp?1270 (Stand 05.06.2013)

21 Our Mobile Planet. Smartphone Studie von Google. Endbericht 2012. http://services.google.com/fh/files/blogs/our_mobile_planet_germany_de.pdf . S. 8 (Stand 05.06.2013)

22 Our Mobile Planet. S. 9

Das US-amerikanische Marktforschungsunternehmen Gartner geht davon aus, dass in diesem Jahr die Zahl der Internetzugriffe mit mobilen, die der stationären Geräte ein- bzw. überholen wird[23]. Dadurch wird das Mobilgerät zum Standardgerät wenn es um den Zugriff auf das Web geht.

Allein seit der Einführung des iPhones 2007 bis zur Veröffentlichung des Nachfolgemodels iPhone 3G hat sich der mobile Datentransfer des Mobilfunkanbieters AT&T um das 50-fache gesteigert[24].

Eine mobil optimierte Umsetzung von Websites gewinnt also aufgrund der gestiegenen Zugriffszahlen immer mehr an Bedeutung. Ist eine mobile Website schlecht umgesetzt, so wechseln 22% der Besucher sofort zur Konkurrenz, sofern diese ein besseres Angebot bietet. Die restlichen 78% versuchen den Zugriff ein zweites, aber kein drittes Mal[25]. Ist eine Website mobil optimiert entwickelt, kann eine Vielzahl von Nutzern gewonnen und gebunden werden. Ein beeindruckendes Beispiel hierfür stellt die Webpräsenz des sozialen Netzwerkes Facebook dar, die nach dem MF-Ansatz entwickelt wurde und 425 Millionen aktive Nutzer pro Monat[26] zu verzeichnen hat. So werden 19% aller Facebook-Posts über diese Seite generiert[27] und damit die nativen Anwendungen überholt. Vaughan Smith, „Vice President of Corporate Development" bei Facebook, sagte auf der „Global Mobile Internet Conference"[28]:

> *„Since January we've completely changed the way we do product development. We've trained all our engineers to do mobile first."*

Das Ende der Verbreitung von mobilen Endgeräten und der somit gesteigerten Notwendigkeit der Entwicklung mobil optimierter Websites ist nicht abzusehen, da die Geräte immer leistungsfähiger und günstiger werden[29]. Außerdem müssen Platzhirschen wie Apple und Google die Konkurrenz zahlreicher anderer Betriebssystemanbieter (Firefox, Ubuntu oder Windows Phone) fürchten, was den Absatzmarkt noch weiter ankurbeln dürfte.

---

23 Mobile phones will overtake PCs as the most common Web access device worldwide [Gartner]. http://www.nextbigwhat.com/mobile-phones-will-overtake-pcs-as-the-most-common-web-access-device-worldwide-gartner-297/ (Stand 05.06.2013)

24 Wroblewski, Luke. Designing for today's web. http://static.lukew.com/TodaysWeb_09302010.pdf (Stand 05.06.2013)

25 New Study Reveals the Mobile Web Disappoints Global Consumers. http://www.compuware.com/d/release/592528/new-study-reveals-the-mobile-web-disappoints-global-consumers (Stand 26.02.2013)

26 Perez Sarah. Facebook's Mobile Monthly Active Users Grew 21% Over Past Four Months. http://techcrunch.com/2012/02/01/facebook-has-425-million-mobile-monthly-active-users-up-from-350-million-in-september/ (Stand 05.06.2013)

27 Ebenda.

28 Costine, Josh. Facebook VP: We Pivoted To Create The Right Mobile Experience First, The Desktop Can Catch Up Later. http://techcrunch.com/2012/10/19/facebook-mobile-first/ (Stand 05.06.2013)

29 Preisentwicklung bei verschiedenen Smartphone-Modellen in Deutschland von 2010 bis März 2011 (in Euro). http://de.statista.com/statistik/daten/studie/182567/umfrage/preisentwicklung-bei-smartphones-in-deutschland/ (Stand 05.06.2013)

Zusammenfassend lässt sich also sagen[30], dass

- Die Verbreitung und Benutzung von ME ständig zunimmt
- Somit das mobile Internet zentraler Bestandteil des täglichen Lebens geworden ist
- ME überall verwendet werden

Es wird deutlich erkennbar, dass das Internet längst mobil geworden ist und dessen Wachstum einen schwerwiegender Grund für die Umsetzung von MFRWD darstellt. Durch diese Tatsache, hat sich die Landschaft der verfügbaren Geräte deutlich verändert, was nachfolgende Abbildung verdeutlicht[31].

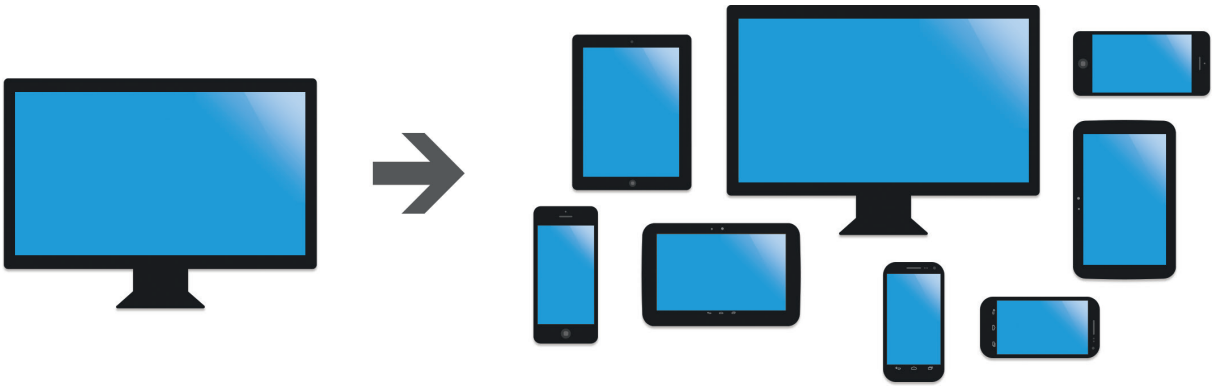


Abbildung 6 - Das Web vor und nach 2007

3.2. Einschränkungen als Chance

Mobile Geräte weisen eine Vielzahl von Unterschieden und Einschränkungen im Vergleich zu stationären Geräten auf[32]. Der Ansatz MFRWD schlägt vor, aufgrund dieser Limitationen elegante Lösungen zu entwickeln, die diese umgehen und zu einem insgesamt besseren Weberlebnis – sowohl auf dem mobilen also auch auf dem stationären Endgeräte – führen. Die Hemmnisse müssen also als Chance und nicht als Einschränkungen begriffen werden[33].

3.2.1. Display-Größen und der Zwang sich auf das Wesentliche zu fokussieren

Die erste und signifikanteste Einschränkung, die es zu umschiffen gilt, ist die der kleinen Bildschirme. Im Vergleich zu stationären Geräten verfügen die meisten ME nur über eine beschränkte Bildschirmgröße. Somit steht für den Inhalt der Website wesentlich weniger Platz zur Verfügung. Da, wie bereits in *Abschnitt 2.1* beschrieben, bei MFRWD vom Mobilgerät ausgehend hin zum Desktoprechner entwickelt wird, ist es nötig, den Benutzer und dessen Bedürfnisse explizit in den Mittelpunkt zu stellen.

30 Our Mobile Planet. S. 6-9
31 Grafik angelehnt an: Frost, Brad. This is the web. http://bradfrostweb.com/blog/post/this-is-the-web (Stand 10.06.2013)
32 Maurice, Florence. S. 21
33 Wroblewski, Luke. S. 18

Es erwächst der Zwang sich auf das Wesentliche zu konzentrieren, die Notwendigkeit eine Website einfach zu halten: Was braucht der Benutzer wirklich, jetzt in diesem Moment? Wird eine erweiterte Sidebar mit animierter Werbeeinblendung gebraucht, welche die eigentlich wichtige Information in den nicht sichtbaren Bereich verdrängt? Ist das fünfte Composing tatsächlich notwendig und muss die Navigation so kompliziert sein?[34]. Diese Fragen sind zu verneinen. Der Inhalt, die Botschaft steht im Vordergrund.

Die nachfolgende Grafik[35] verdeutlicht anhand des Beispiels einer Universitäts-Website, was die Benutzer erwarten und was sie tatsächlich vorfinden:

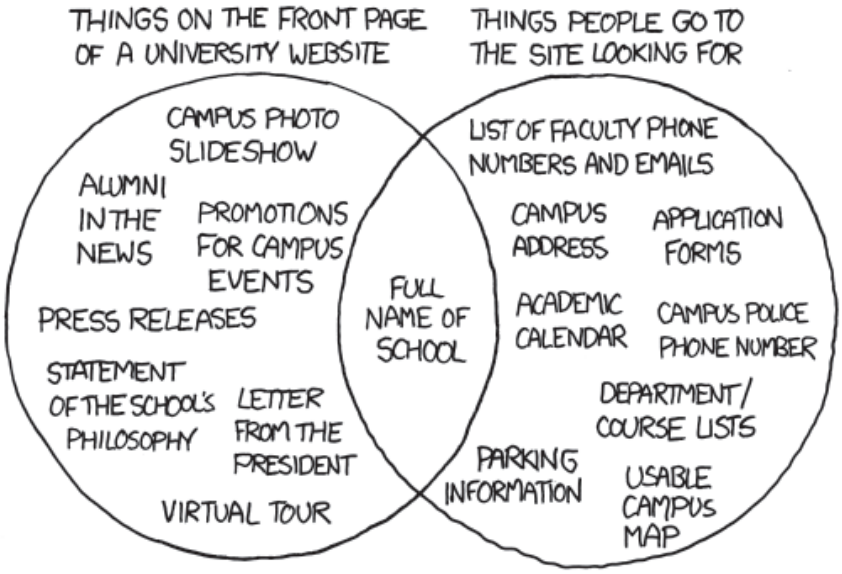


Abbildung 7 - Erwartungen vs. Realität einer Hochschulwebseite

Es ist also wichtig die Aufgaben und Anforderungen an eine mobile Website nach Relevanz zu ordnen und zu strukturieren, worauf in *Abschnitt 4.5* noch genauer eingegangen wird.

Wird die Fokussierung auf das Wesentliche richtig umgesetzt, so profitiert davon auch die Darstellung der Website auf Desktopgeräten, da beispielsweise Navigationen einfacher bedient oder Inhalte schneller aufgefunden werden können. Die Einschränkungen eines kleinen Bildschirms ist also als Chance zur Verbesserung des gesamten Weberlebnisses – mobil wie stationär – zu sehen.

3.2.2. Limitierte Bandbreite und Datenübertragungsgeschwindigkeit

Die Datenübertragung eines mobiles Gerät ist im Normalfall schlechter und somit langsamer als auf Desktoprechner. Die Geschwindigkeit die beim Übertragen einer Ressource (HTML-Dokument, CSS-Datei, Bild etc.) erreicht wird, hängt zum einen davon ab, welche Datenmenge auf einmal übertragen werden kann, zum anderen wie hoch die Latenzzeit für eine einzelne Verbin-

34 Wroblewski, Luke. Mobile First. S. 19
35 Bildquelle: University-Website. http://xkcd.com/773/ (Stand 10.06.2013)

dung ist[36]. Diese Latenzzeiten sind bei Mobilgeräten höher als bei Desktopgeräten, weshalb die Datenmenge einer Website möglichst gering gehalten werden sollte.

Die Verbindungstechnologien, die momentan eingesetzt werden, sind die Folgenden[37]:

Name	Downlink	Latenzzeit
GPRS	53,6 kBit/s	500 ms und mehr
EDGE	236,8 kBit/s	300 bis 400 ms
UMTS	384 kBit/s	170 bis 200 ms
HDSPA	1,8 MBit/s - 7,2 MBit/s	50 bis 70 ms
LTE (LTE Advanced)	50 - 500 MBit/s	50 ms

Tabelle 1 - Downlink und Latenzzeiten mobiler Verbindungstechnologien

Es empfiehlt sich daher eine Website ressourcenschonend zu gestalten und somit die Ladezeit zu verringern, was durch diverse Techniken, wie beispielsweise in *Abschnitt 4.4.2.1* dargelegt ,zu erreichen ist.

Kann eine Website nun also auf einem mobilen Endgerät schnell geladen werden, so ist dies umso schneller auf stationären Rechnern möglich. Die Limitation der Verbindungsgeschwindigkeit und die Maßnahmen, die dagegen zu ergreifen sind, steigern also auch die Performance auf herkömmlichen Rechnern und sind somit als Chance zu betrachten.

3.2.3. Zeitliche Begrenzung

Die mobile Nutzung findet in kürzeren Zeitintervallen statt, als dies auf dem Desktop der Fall ist,. Der Benutzer hat daher weniger Zeit, relevante Informationen aufzunehmen und zu verarbeiten[38], was ein weiterer Grund ist, sich auf das Wesentliche – den Inhalt der im Moment für den Benutzer wichtig ist – zu fokussieren.

Nach Googles „Mobile User Experience Strategy“ kann zwischen drei mobilen Benutzertypen unterschieden werden[39], deren Bedürfnisse in den Mittelpunkt gestellt werden müssen:

- Repetetive Now - Jemand, der in wiederkehrenden Zyklen nach ein und der selben Information sucht (Informationsabruf bezüglich Wetter, Aktienkurs)
- Bored now - Der Benutzer, der aus Langweile eine Website besucht, um sich unterhalten zu lassen (Facebook, Instagram)
- Urgent now - Die Gruppe an Usern, die jetzt in diesem Moment eine spezifische Information benötigt (Suche nach der nächste Apotheke)

36 Maurice, Florence. S.22
37 Datenübertragung im Mobilfunk. http://www.elektronik-kompendium.de/sites/kom/0910141.htm (Stand. 12.06.2013)
38 Is mobile affecting when we read? http://readitlaterlist.com/blog/2011/01/is-mobile-affecting-when-we-read/ (Stand 12.06.2013)
39 Wellman, Stephen. Google lays out it's Mobile User Experience Strategy. http://www.informationweek.com/mobility/business/google-lays-out-its-mobile-user-experien/229216268 (Stand 12.06.2013)

Werden diese verschiedenen Benutzergruppen in die Überlegungen zur Entwicklung einer Website eingebracht, so profitieren auch hier wieder die mobile und stationäre Umsetzung in gleichem Maße, da Inhalte mit größerer Sorgfalt erstellt werden müssen.

3.2.4. Fazit

Werden nun also die Hemmnisse und Einschränkungen der mobile Endgeräte umschifft und eine mobil optimierte, durchdachte Website kreiert, wird man auch auf stationären Rechnern von der entwickelten Lösung profitieren. Auch wenn nicht explizit eine derartige Website konzipiert und realisiert werden soll, so hilft MFRWD dennoch, das Weberlebnis „normaler" Websites durch klare Informationsstrukturen, hoher Performance und der richtigen Information zur richtigen Zeit zu verbessern.

3.3. Mobile Fähigkeiten

Mobile Endgeräte weisen aber nicht nur Einschränkungen auf, sondern bieten durch ihre Eigenheiten und Technologien auch eine Vielzahl neuer Möglichkeiten, das Weberlebnis zu verbessern. Im Nachfolgenden werden beispielhaft zwei gängige Technologien beleuchtet und ein Ausblick auf die zukünftige Entwicklung gegeben.

3.3.1. Geolocation

Informationen können in Abhängigkeit des Aufenthaltsortes des Nutzers zur Verfügung gestellt werden, wenn auf die Geolocation des Benutzers zurückgegriffen wird. Dies ist für mobile Websites durch die „Geolocation API" des W3C[40] einfach zu realisieren. Anders als bei bisherigen Technologien wird hier nicht nur über die IP des Clients festgestellt, woher die Anfrage kam, sondern auch über weitere Parameter wie die SS-IDs umliegender W-LAN Netzwerke und natürlich auch über ein möglicherweise vorhandenes GPS-Modul des mobilen Geräts. Mit diesen Informationen können unter anderem Webservices wie Hotel- oder Filialsuchen wesentlich komfortabler und intuitiver gestaltet werden, da der Benutzer nicht erst – unter Umständen sogar in einer ihm unbekannten Umgebung – mühevoll seinen Standort mitteilen muss. Auch Suchmaschinen oder Wetterdienste können von diesen Daten profitieren und dem Benutzer die Informationen bereitstellen, die an Ort und Stelle für ihn relevant sind.

3.3.2. Touch

Durch den Einsatz von Touch-Bildschirmen hat der Benutzer die Möglichkeit unmittelbar mit dem Medium zu interagieren, ohne den Zwischenschritt mit Maus und Tastatur bemühen zu müssen. Dies entspricht eher unserem natürlichen Verhalten, Dinge durch Bewegung zu steuern

40 W3C Editors Draft. Geolocation API Specification. http://dev.w3.org/geo/api/spec-source.html (Stand 16.06.2013)

und trägt somit zu einem besseren, intuitiveren Weberlebnis bei[41].

So können beispielsweise Gesten eingesetzt werden, um bestimmte Aktionen auf Websites zu steuern. Gängige Beispiele sind „Pull to refresh" – also das Ziehen einer Website um bestimmte Bereiche neu zu laden – oder „Swipen", das Wischen um beispielsweise Seitenbereiche zu Wechseln oder eine Bildergalerie zu durchstöbern.

3.3.3. Ausblick

Zusätzliche Features wie der Zugriff auf die Daten der Kamera eines mobilen Endgeräts, NFC, Bluetooth usw. sind momentan noch nativen Apps vorbehalten, werden aber in naher Zukunft auch den Weg in den Browser finden. Diese Fähigkeiten werden die Interaktion mit mobilen Websites nachhaltig verändern, das „Feeling" verbessern[42] und weitere Möglichkeiten bieten, kreative Lösungen zu realisieren.

3.4. Zusammenfassung der Gründe für MFRWD

Die dargelegten Gründe zeigen deutlich, wieso es sich lohnt und es für zukünftige Website-Projekte unumgänglich ist, den Ansatz MFRWD zu verfolgen. Nicht nur in technischer Hinsicht werden Websites besser und leistungsstärker, sondern auch der Inhalt und somit der Benutzer rückt in den Vordergrund. „Mobile First" bedeutet also auch „Content First" und somit ebenso „User First"[43].

Zusammenfassend lässt sich sagen, dass Websites die mit MFRWD entwickelt werden

- Zukunftssicher sind und sich dynamisch den jeweiligen Auflösungen anpassen
- Nur eine Codebasis benötigen um eine Vielzahl von Anzeigegeräte zu unterstützen
- Hinsichtlich Performance, Usability, Content etc. deutliche Vorteile aufweisen

Selbst Websites, deren primäre Zielgruppe nicht aus Nutzern mit mobilen Endgeräten besteht, profitieren von „Mobile First Responsive Webdesign":

> *„If you can support the mobile web, you can support anything".*

Mit diesen kurzen Worten bringt der Webentwickler und Front-End Designer Brad Frost die Gründe für die Entwicklung von Websites mit MFRWD auf den Punkt[44]. Es wird ersichtlich, dass MFRWD kein Ansatz exklusiv für Mobilgeräte, sondern für alle Geräte ist.

41 Wroblewski, Luke. Mobile First. S. 42
42 Wroblewski, Luke. Mobile First. S. 44
43 Wroblewski, Luke. An Event Apart: Content First. http://www.lukew.com/ff/entry.asp?1598 (Stand 05.07.2013)
44 Frost, Brad. Mobile First Responsive Webdesign. http://bradfrostweb.com/blog/web/mobile-first-responsive-web-design/ (Stand 06.07.2013)

4. Umsetzung von Mobile First Responsive Webdesign

Die große Herausforderung bei der Umsetzung von MFRWD besteht darin, dass eine große Anzahl von Geräten unterstützt werden muss. Deshalb müssen die Elemente einer Website weitgehend flexibel und transformationsfähig sein, um sich zu verschieben, zu verkleinern und auf die jeweilige Umgebung reagieren zu können.

Nachfolgende Kapitel stellen die Grundlagen und das Vorgehen zur Realisierung der wichtigsten Teilbereiche einer Website nach MFRWD vor. Hierbei wird das Augenmerk darauf gelegt, welche Faktoren zu beachten sind, inwiefern bisher Umsetzungen existieren und wie diese funktionieren. Des Weiteren werden deren Schwächen analysiert und geklärt, ob und in welcher Form verbesserte Implementierungen in Zukunft zu erwarten sind.

4.1. Media Queries

4.1.1. Grundlagen

Media Queries (MQs) bilden den Unterbau von MFRWD und dienen zum Detektieren und Auslesen von Eigenschaften der jeweiligen Anzeigegeräte. Auf Basis dieser festgestellten Eigenschaften kann dann reagiert und Bereiche einer Website je nach Bedarf angepasst werden. Man spricht also von medienabhängigen Stylesheets[45]. Sie sind Teil der CSS3 Spezifikationen, die zwar nach wie vor nicht offiziell durch das W3C verabschiedet wurden, jedoch aufgrund der sehr guten Browserunterstützung – aus der Riege aktueller Browser unterstützt nur Internet Explorer 8 MQs nicht[46] – als De-facto-Standard angesehen werden dürfen.

Bereits mit CSS 2.1 wurde der Vorläufer der MQs, die sogenannten „Media Types“ eingeführt, mit denen es möglich war, auf relativ abstrakte Art unterschiedliche Anzeige- bzw. Ausgabegeräte zu erkennen, um beispielsweise ein eigenes, zusätzliches Print-Stylesheet für den Ausdruck einer Website zur Verfügung zu stellen:

```
<link rel="stylesheet" media="print" href="print.css" />
<link rel="stylesheet" media="screen" href="screen.css" />
```

Die Angabe nach *media=* definiert hierbei das Ausgabegerät. Der Browser entscheidet mit Hilfe dieses Schlüsselworts, welches Stylesheet aktuell zu verwenden ist. Die Formatierungen der einzelnen Ausgabemedien können aber auch innerhalb eines einzelnen Stylesheets definiert werden:

45 W3C Recommendation. Media Queries. http://www.w3.org/TR/css3-mediaqueries/ (Stand 05.07.2013)
46 Browserunterstützung von Media Queries. http://caniuse.com/#feat=css-mediaqueries (Stand 05.07.2013)

```
@media print {
  /* Formatierungen für Drucker */
}

@media screen {
  /* Formatierung für Bildschirme */
}
```

Neben den Angaben *screen* und *print* gibt es noch *handheld*, *tv* und *projection*, die von den Browsern jedoch kaum bis gar nicht unterstützt werden. So registrieren sich Mobilgeräte beim Browser ebenso wie Desktoprechner als *screen*-Geräte.

Da im MFRWD primär die Bildschirmgröße und weitere Parameter von Interesse sind, reichen die Media Types bei weitem nicht aus, um die benötigten Informationen zu erhalten. MQs erweitern die Abfragemöglichkeiten um die folgenden, relevanten Parameter:

width	Breite des Viewports in px
height	Höhe des Viewporsts in px
device-width	Displaybreite in px
device-height	Displayhöhe in px
orientation	Ausrichtung: portrait oder landscape
aspect-ratio	Verhältnis aus width und height
device-pixel-ratio	Verhältnis von Gerätepixeln zu geräteunabhängigen Pixel (nicht Teil des W3C Standards, benötigt daher browserspezifische Prefixe)
resolution	Displayauflösung in dpi

Tabelle 2 - Abfragemöglichkeiten einer CSS3 Media Query

Mit diesen gerätespezifischen Informationen können nun die Anpassungen auf die Eigenschaften der einzelnen Anzeigegeräte vorgenommen werden.

4.1.2. Anwendung und Syntax

MQs können ebenso wie die Stylesheets für einzelne Ausgabemedien als seperate Dateien eingebunden werden. Dieses Vorgehen ist aber nicht zu empfehlen, da sich dadurch die Anzahl der http-Requests erhöht und somit die Ladezeit der Website zu- und die Performance abnimmt. Sinnvoller ist es, die MQs innerhalb einer einzigen CSS-Datei zu definieren. Zum Entwickeln empfiehlt es sich aber durchaus, je nach Komplexität der MQs, seperate Dateien zu verwenden und diese später bei der Veröffentlichung zu einer einzelnen zusammenzuführen. Für solche Aufgaben empfiehlt sich der Einsatz eines Build-Systems (hierbei sind LESS[47] in Kombination mit Grunt.js[48] interessante Ansätze), welches auch andere Operation wie beispielsweise die Minimierung von Dateien oder die Veröffentlichung der Website selbst übernehmen kann.

47 LESS. Eine dynamische Stylesheet-Sprache. http://lesscss.org/
48 Grunt.js. Ein auf JavaScript und node.js basierendes Build System. http://gruntjs.com/

Syntaktisch ähneln die MQs den Media Types. Das folgende Grundraster zeigt den prinzipiellen Aufbau einer MQ:

```
/* Styles für Smartphone */
@media only screen and (min-width: 980px) {
  body {
   color: red;
  }
}
```

Diese MQ würde die Schriftfarbe des Bodys auf Rot setzen, wenn die Breite des Viewports mindestens 980px beträgt.

Allgemein gilt:

- Eine MQ beginnt immer mit dem Schlüsselwort *@media*.
- Die Angabe *only* verhindert die Interpretation der CSS Anweisungen, wenn ältere Browser MQs nicht unterstützen
- Mit *and* können Bedingungen logisch UND verknüpft werden, hier der Media-Typ *screen* und die Mindestbreite. Will man logisch ODER verknüpfen, so trennt man die MQs mit einem Komma voneinander, wie es für CSS üblich ist
- Mit den Präfixen *min-* bzw. *max-* können Bereiche definiert werden
- Innerhalb der MQs wird normales CSS definiert

Die Angabe des Media-Typs wird innerhalb einer MQ eigentlich nicht benötigt, da wie bereits erwähnt, alle relevanten Geräte den Wert *screen* ausgeben und eine weitere Unterscheidung nicht notwendig ist.

Damit die MQ tatsächlich funktioniert und die kontextspezifischen Anpassungen erfolgen können, muss zusätzlich zu den eigentlichen CSS-Defintionen, im <*head*> des HTML-Dokuments der Viewport definiert werden. Dies hat historische Gründe[49]: Als das erste iPhone auf den Markt kam, wurden Websites zunächst für eine imaginären Desktop-Breite gerendert[50] und dann auf die mobile Breite verkleinert. Diesen Umweg gehen mobile Browser auch heute noch, weshalb es äußerst wichtig ist den korrekten Viewport über einen *meta*-tag zu definieren.

```
<meta name="viewport" content="width=device-width initial-scale=1.0,
maximum-scale=1.0, user-scalable=0" >
```

Der *viewport*-tag wurde exakt für diesen Zweck von Apple entwickelt und wird mittlerweile von allen gängigen Browsern beherrscht. Hier wird die „imaginäre"-Breite auf die tatsächliche Breite des Gerätebildschirms gesetzt. Durch die Anweisungen *initial-scale*, bzw. *maximum-scale=1.0*

49 Zillgens, Christoph. S. 47
50 Ebenda.

und *user-scalable=0* wird ein Zoomen durch den Benutzer verhindert. Dies macht – trotz kontroverser Diskussion in der Entwicklergemeinde – Sinn, da eine mobil optimierte Seite im Normalfall nicht gezoomt werden muss.

Nach der Philosophie von MFRWD ist es sinnvoll, alle Formatierungen, welche die Website in der Basisunterstützung – also im mobilen Kontext – benötigt, ohne MQ zu definieren und die Anpassungen hinzu höheren Auflösungen über spezielle MQs vorzunehmen:

```
/* MF Styles */
...
h1 {
 font-size: 18px;
}

/* Desktop Styles */
@media (min-width: 768px){
  ...
  h1 {
   font-size: 25px;
  }
}
```

Wie MQs im Detail, also beispielsweise zur Definition von Layout oder zur Detektion von hochauflösenden Bildschirmen zu verwenden sind, wird in den jeweiligen Kapiteln näher beleuchtet.

4.1.3. Kritik und Ausblick

MQs und (MF)RWD erfreuen sich großer Akzeptanz innerhalb der Entwicklergemeinde, was durch die enorme Anzahl an neu entstandenen responsiven Websites, den unzähligen Blogeinträgen- und Technikdiskussionen sowie den Themenschwerpunkten wichtiger Webentwicklerkonferenzen[51] belegt ist. Dennoch gibt es auch, teilweise zurecht, kritische Meinungen.

Einer der Hauptkritikpunkte richtet sich gegen die potentiellen Performance-Einbußen bei der Verwendung von MQs[52]. Dadurch, dass größere CSS-Dateien geladen werden müssen, verzögere sich der Aufbau der Website. Ferner ginge die Verkleinerung von Websites und die Skalierung von Bildern zu Kosten der CPU-Auslastung. Diese Tatsachen sind unbestreitbar richtig, jedoch sind MQs nicht deren Ursache. Zum einen kann die Dateigröße einer CSS-Datei mittels Kompression und Konkatenation zwischen 70-90% minimiert werden[53], sodass die erhöhte Anzahl an Code für MQs nicht mehr ins Gewicht fällt. Zum anderen nimmt die Leistungsfähigkeit der

51 Hier ist beispielsweise die SmashingConference zu nennen. http://smashingconf.com/speakers
52 Grigsby, Jason. CSS Media Query for Mobile is Fool's Gold. http://blog.cloudfour.com/css-media-query-for-mobile-is-fools-gold/ (Stand 05.07.2013)
53 Yahoo! Developer Network. Best Practises for Speeding Up Your Websites. http://developer.yahoo.com/performance/rules.html (Stand 05.07.2013)

Prozessoren mobiler Endgeräte permanent zu[54].

Wird zudem der Ansatz MFRWD korrekt umgesetzt, sprich alle notwendigen Formatierungen werden zuerst und ohne MQ für den mobilen Kontext definiert (siehe oben), muss die Website auch nicht nachträglich durch Anpassungen verkleinert werden. Das Skalieren und Bereitstellen von mobil optimierten Bildern wird im *Abschnitt 4.4* thematisiert.

Ein weiterer Punkt der kritisiert wird, ist die Tatsache, dass MQs vielmehr an die Eigenschaften des Browser bzw. Geräts gebunden sind, und nicht auf HTML-Elemente übertragen werden können. Die Forderung nach „Element Queries“[55] steht im Raum.

So sollte es beispielsweise möglich sein, ein Seitenelement umzugestalten, wenn der umfassende Container und nicht der Viewport, eine Mindestbreite von 500 Pixel hat, was folgender Code und Screenshot[56] illustrieren.

```
.container (min-width: 500px) {
   .element {
      /* Element Styles */
   }
}
```

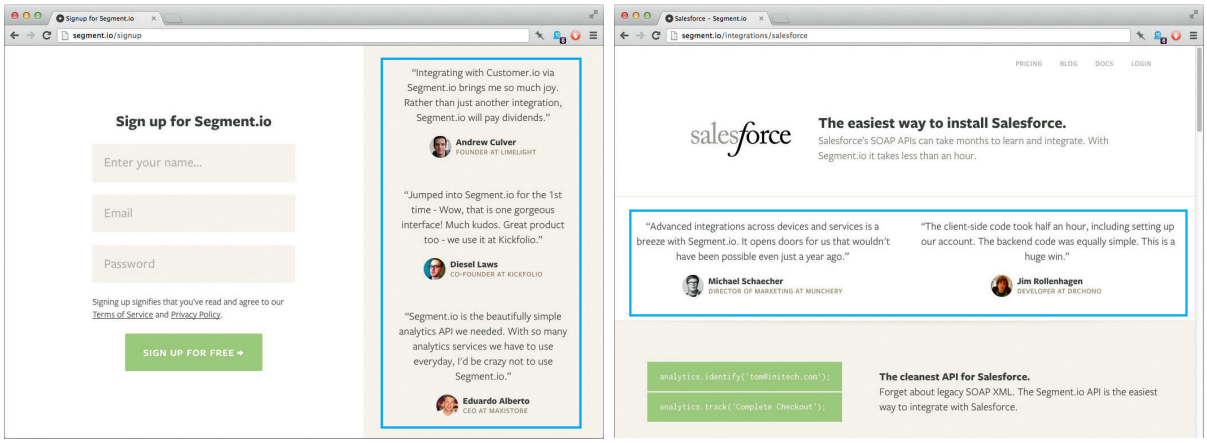


Abbildung 8 - Beispiel für den sinnvollen Einsatz von Element Queries

Dies deckt sich mit der Forderung anstatt Seiten, zukünftig Module zu gestalten, die sich dann zu einem Ganzen zusammensetzen[57]. Natürlich ist es auch möglich, diese Parameter über eine weitere CSS-Klasse zu verändern, soll ein- und dasselbe Modul innerhalb eines anderen Kontextes inkludiert werden. Jedoch bricht dies die grundsätzliche Idee der Modularisierung und führt zu mehr Code. „Element Queries“ oder elementbezogene MQs sind also eine berechtigte, sinnvolle Forderung. Leider sind diese bisher weitgehend nur theoretische Überlegungen und le-

54 Fastest Processor. Cell Phone Top List. http://www.phonegg.com/Top/Fastest-Processor-Cell-Phones.html (Stand 06.07.2013)
55 Matanich, Tyson. Media Queries are not the answer. http://coding.smashingmagazine.com/2013/06/25/media-queries-are-not-the-answer-element-query-polyfill/ (Stand 05.07.2013)
56 Screenshot der Website Segment.io. https://segment.io/ (Stand 13.07.2013)
57 Matanich, Tyson. Media Queries are not the answer.

diglich als konzeptionelle Polyfills[58] vorhanden. Es wird vermutlich noch einige Zeit in Anspruch nehmen, bis sich das W3C dieser Problematik annimmt. Die Diskussion darüber stellt in jedem Falle einen Schritt in die richtige Richtung dar.

4.2. Layout

4.2.1. Layout vor MFRWD

Wie bereits in *Kapitel 2. Gründe für Mobile First Responsive Webdesign* dargelegt wurde, hat sich die Anzahl unterschiedlicher Anzeigegeräte um ein Vielfaches multipliziert. Jedoch sind nach wie vor viele Website-Layouts auf den Viewport 1024x768 Pixel optimiert. Vor 2007 war dies ein durchaus legitimes Vorgehen, auch wenn bereits zu diesem Zeitpunkt andere Bildschirmgrößen verfügbar waren. Diese waren aber zumeist größer und konnten somit Websites, die für die „Standard-Bildschirme" gestaltet waren, problemlos anzeigen. Die Layouts wurden einfach zentriert, links und rechts davon Weißraum angezeigt. Das Problem, Websites auf kleinere Displays zu portieren gab es nicht, da die Geräte schlicht und einfach nicht existierten. Man konnte also mit einem Layout die meisten Betrachtungsgeräte[59] abdecken.

Leider werden teilweise bis heute eine große Zahl von Websites mit einer Seitenbreite von +/- 960 bis 980 Pixel gestaltet. Häufig ist es noch gängige Arbeitspraxis, wie man sie auch aus dem Printdesign kennt, Layouts in einem grafischen Editor zu gestalten und dann 1:1 pixelperfekt in HTML umzusetzen. Dabei werden Werte wie Schrift- oder Elementgrößen absolut in Pixel definiert. Die technische Möglichkeit, prozentuale Angaben zu machen, bleibt aus Bequemlichkeit, der Angst vor Verlust der gestalterischen Kontrolle oder auf Grund der Unwissenheit um die Wichtigkeit des mobilen Internets, oftmals ungenutzt[60].

Derartige Websites werden auf mobilen Endgeräten wie folgt angezeigt[61]:

Abbildung 9 - Darstellung einer nicht MFRWD-Website auf einem iPhone

Die Website wird auf Basis einer imaginären Desktop-Breite gerendert und dann in den vorhan-

58 Matanich, Tyson. Element Query Polyfill. https://github.com/tysonmatanich/elementQuery (Stand 05.07.2013)
59 Zillgens, Christoph. S 2
60 Ebenda.
61 Screenshot der Website Casalinga. http://casalinga.de/ (Stand 13.07.2013)

den Raum skaliert. Das führt dazu, dass sie lediglich in einer „Totalen", also wie von sehr weit entfernt, dargestellt wird. Navigation und Inhalt bleiben weitgehend unlesbar. Der Benutzer muss, will er mehr erfahren und die Website bedienen, umständlich mittels „pinch-to-zoom" vergrößern. Wechselt er dann eine Seite innerhalb der Website, so muss der gleiche entnervende Prozess wiederholt werden.

Glücklicherweise wurde vielen Webdesignern- und Developern klar, dass ein Umdenken hinsichtlich Gestaltung und Umsetzung von Websites erfolgen muss. Der Entwickler Ethan Marcotte definierte in seinem richtungsweisenden Buch „Responsive Webdesign" folgende Grundbestandteile um zukünftig flexible Layouts zu entwickeln[62]:

- Flexible Raster (Grids)
- Flexible Bilder und Medien
- Media Queries

Der Anspruch von MFRWD ist es, Websites auf allen Auflösungen und an das jeweiligen Medium angepasst darzustellen. Mit Hilfe der drei Hauptbestandteile nach Ethan Marcotte ist es möglich, dies zu erreichen.

4.2.2. Layout mit MFRWD

Die Grundidee des RWD besteht darin, Elemente auf kleinen Auflösungen anders anzuordnen als auf großen. Ändert sich also der Kontext oder die Größe des Browserfensters, so verbreitern sich die Elemente der Website dynamisch und nehmen mehr Platz ein, da ihre Größe prozentual definiert ist. Sie müssen umpositioniert werden, um den Anforderungen des jeweiligen Umfeldes gerecht zu werden. Im mobilen Kontext sind Inhalte zumeist übereinander gestapelt, wenn die Breite nicht ausreicht, um beispielsweise mehrere Spalten nebeneinander anzuordnen.

4.2.2.1. Layout Patterns

Um ein Layout an verschiedene Bildschirmgrößen anzupassen, bedarf es Strategien, wie sich Elemente umpositionieren bzw. transformieren sollen. Layout Patterns, also Vorgaben, wie ein Layout sich auf verschiedenen Auflösungen verhält, helfen bei der Gestaltung und Umsetzung von Websites mit MFRWD.

Bisher haben sich hauptsächlich folgende drei Pattern, die durch Luke Wroblewski geprägt wurden[63], in der Entwicklergemeinde etabliert. Deren Vor- und Nachteile können nicht pauschal genannt werden, da die Wahl eines Layout Pattern immer in Abhängigkeit des Inhalts getroffen werden muss. Sie stellen sich wie folgt dar:

62 Marcotte, Ethan. Responsive Webdesign. New York: A book Apart 2011. S. 9
63 Wroblewski, Luke. Multi Device Layout Pattern. http://www.lukew.com/ff/entry.asp?1514 (Stand 06.07.2013)

Mostly Fluid

Dieses Layout verwendet mehrere Spalten, die auf höheren Auflösungen größere Abstände besitzen. Inhalte und Bilder werden beim Übergang von einer in die andere Anordnung skaliert. Im mobilen Kontext sind die Spalten übereinander gestapelt. Wird dieser verlassen, ändert sich das Layout im Grunde nicht mehr, daher auch die Bezeichnung „Mostly Fluid" - größtenteils fließend.

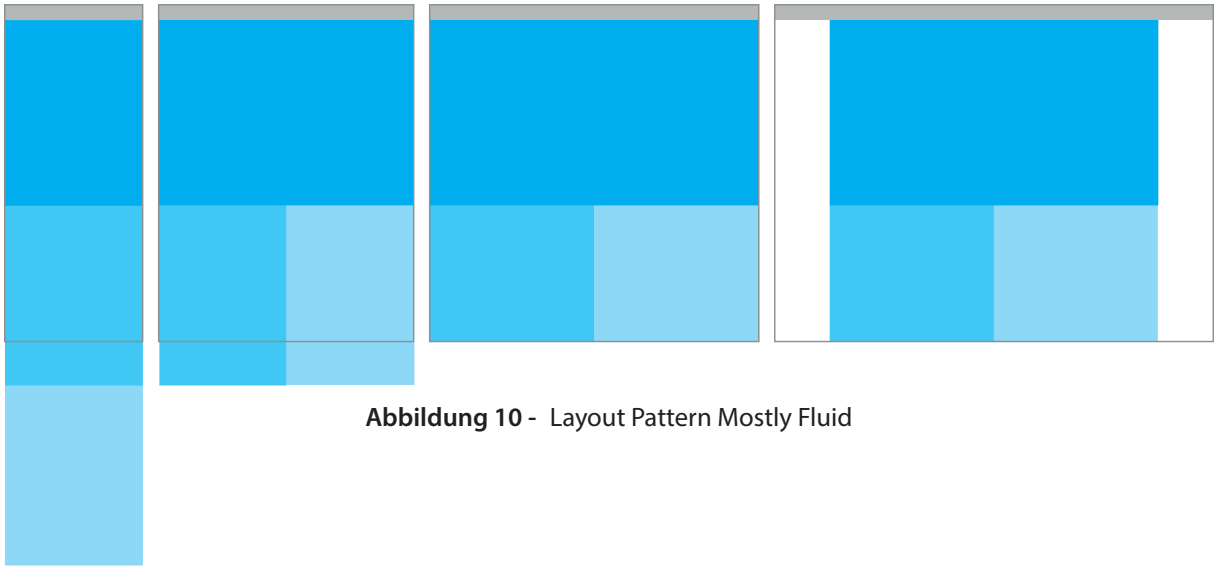


Abbildung 10 - Layout Pattern Mostly Fluid

Column Drop

Betrachtet man das Layout ausgehend von der größten Auflösung hin zur kleinsten, so werden die Spalten nach und nach übereinander gestapelt, was namensgebend für dieses Pattern war.

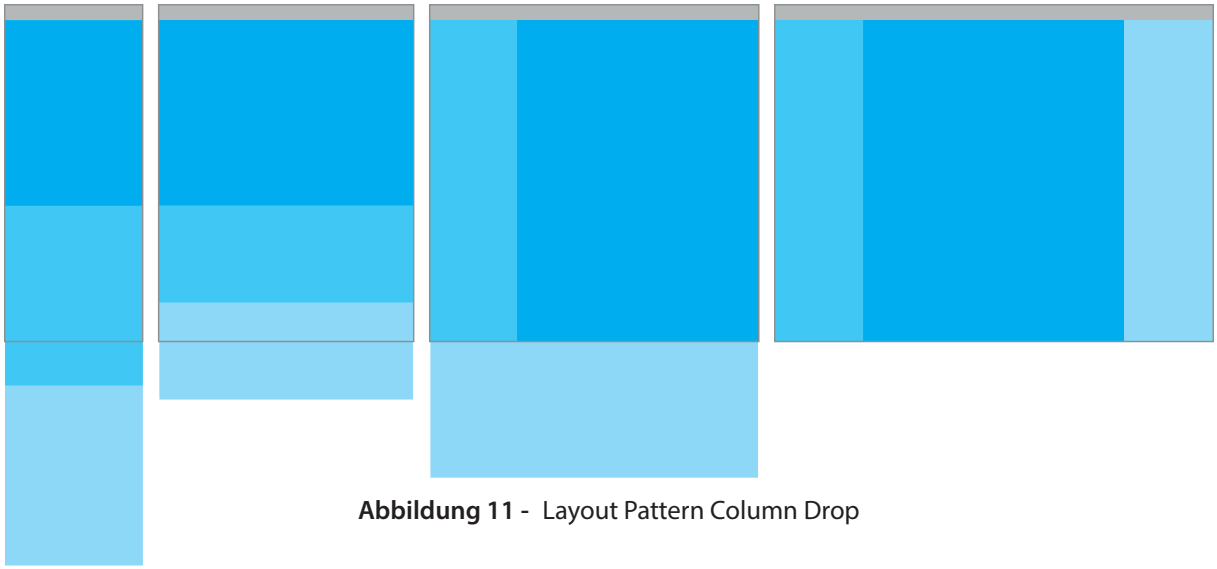


Abbildung 11 - Layout Pattern Column Drop

Off-Canvas

Die Idee dieses Layouts besteht im Gegensatz zu den anderen hier vorgestellten Pattern nicht darin, alle Inhalte auf einer Seite anzuzeigen, sondern weniger wichtige Inhalte temporär, bis der Nutzer sie benötigt, außerhalb der Anzeigefläche zu verstecken. Die nicht sichtbaren Bereiche können im mobilen Kontext auf Wunsch des Users angezeigt werden. „Off-Canvas" kommt der Philosophie von MFRWD, Inhalte zu priorisieren (Mobile First = Content First) am nächsten und

ist daher das momentan wohl populärste Layout Pattern.

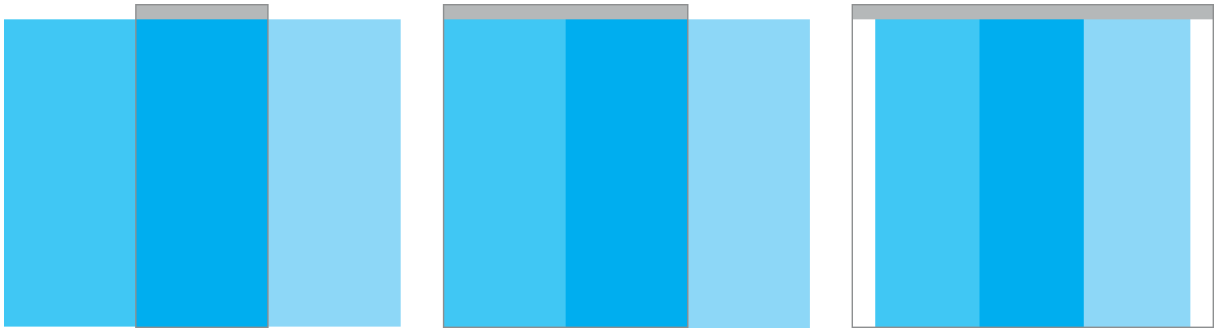


Abbildung 12 - Layout Pattern Off-Canvas

4.2.2.2. Grids und Breakpoints

Bei einem Grid (Raster) handelt es sich um ein Gestaltungshilfsmittel, welches dem Printdesign entstammt, sich aber auch bei der Entwicklung von Websites großer Beliebtheit erfreut, da es bereits in der Gestaltung und später in der technischen Umsetzung bei der Positionierung von Elementen hilft. Dabei wird der zur Verfügung stehende Raum in mehrere gleich große Spalten (Cols) mit gleich großen Abständen eingeteilt. Da sich im MFRWD die Elemente einer Website flexibel anpassen, muss das Grid prozentual definiert sein. Es ist natürlich auch möglich, auf selbiges zu verzichten und nur einzelne Spalten mit Prozentwerten zu versehen. Es ist aber deutlich einfacher, auf ein komplettes Raster zurückzugreifen, um für komplexe Layouts gewappnet zu sein.

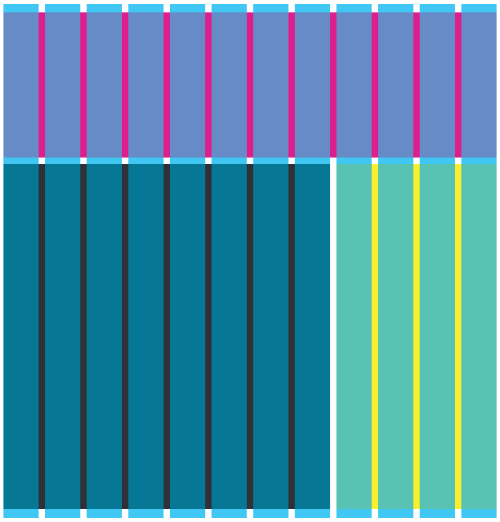


Abbildung 13 - Beispielhaftes 12-Col Grid

Je höher der abgebildete Detailgrad und je filigraner die Positionierungsmöglichkeiten innerhalb eines Grids sein sollen, desto höher muss die Anzahl der einzelnen Rasterspalten gewählt werden. Viele Entwickler greifen auf ein Grid mit zwölf Spalten zurück, da ein solches einen guten Kompromiss aus gestalterischer Freiheit und Komplexität darstellt. In der oben abgebildeten Grafik sieht man, dass die einzelnen Bereiche immer genau eine bestimmte Anzahl an Spalten (Cols) belegen. Wird die Website nun vergrößert, so expandieren die Cols und die Abstände zwischen ihnen, auch Gutter genannt, einfach prozentual.

Die Sidebar nimmt in diesem Beispiel vier, die Hauptspalte acht und der Header alle zwölf Spalten ein. Geht man davon aus, dass der jeweilige Container ein Klassenattribut besitzt, welches angibt wie viele Spalten des Grids er einnimmt, sieht ein grober HTML-Aufbau wie folgt aus:

```
<div class="header col-12">
    ...
</div>

<div class="main col-8">
    ...
</div>

<div class="sidebar col-4">
    ...
</div>
```

Ein Ausschnitt aus dem korrespondierenden Stylesheet zeigt diesen Aufbau:

```
[class^=col-] {
 margin-left: 2.127659574468085%;
}

.col-12 {
    width: 100%;
}

.col-8 {
    width: 65.95744680851064%;
}

.col-4 {
    width: 31.914893617021278%
}
```

Zur Berechnung dieser Prozentwerte kann von einer Website mit beliebiger Seitenbreite ausgegangen werden. Geht man von einer „klassischen" und starren Breite von 940 Pixel aus, so ergibt sich in einem Grid mit zwölf Cols eine Spaltenbreite von 60 Pixel und ein Spaltenabstand von 20 Pixel. Das flexible Grid berechnet sich nach der Formel:

Zielgröße ÷ Kontext x 100 = gesuchter Prozentwert[64].

So stellt sich die Rechnung für eine Spalte, die sich über 4 Cols (Zielgröße = 300 Pixel bestehend aus Spaltenbreiten inklusive Abstände) erstreckt, folgendermaßen dar:

300 Pixel ÷ 940 Pixel x 100 = 31.914893617021278%

Mit Hilfe dieser simplen Gleichung ist es möglich, ein ehemals starres in ein flexibles Grid zu verwandeln, welches sich dynamisch an die Seitenbreite der jeweiligen Bildschirmgröße anpasst. Natürlich müssen auf gleiche Art und Weise auch die Abstände der Cols zueinander berechnet werden.

64 Marcotte, Ethan. S. 41

Befindet man sich nun im mobilen Kontext, so würde dieses soeben berechnete Grid Darstellungsprobleme verursachen. Durch die prozentualen Breiten, würden die Spalten viel zu klein skaliert werden, ihr Inhalt wäre horizontal gestaucht und somit unlesbar. Daher werden die Spalten ab einer gewissen Breite, je nach verwendeten Layout Pattern, übereinander gestapelt.

Es müssen sogenannte Breakpoints definiert werden, die den Übergang von einem zum anderen Kontext markieren. An diesen Punkten „bricht" das aktuelle Layout auf und transformiert sich. Dies kann beispielsweise bedeuten, dass die Spaltenabstände schmäler werden, oder sich im mobilen Kontext die Spalten stapeln. Aber auch Anpassungen der Typografie, die Größenveränderung von Klickflächen oder das Ein- bzw. Ausblenden für den aktuellen Kontext irrelevanter Inhalte sind mögliche Transformationsaufgaben. Die Breakpoints sollten sich nur grob an den Geräteauflösungen orientieren. Vielmehr ist es wichtig, sich der korrekten Darstellung der Inhalte unter unterschiedlichen Bedingungen zu widmen[65].

Bei der technischen Realisierung der Breakpoints kommen die bereits im *Abschnitt 4.1* beleuchteten MQs zum Einsatz:

```
@media (max-width: 767px) {
 [class^=col-] {
   display: block;
   margin-left: 0;
   width: 100%;
 }
}
```

Durch obige MQ werden im mobilen Umfeld alle vorher definierten Spalten auf eine Breite von 100% gesetzt und durch den Befehl *display: block;* übereinander gestapelt.

Bei der Abfrage der Breite empfiehlt es sich, die Eigenschaften *width* und *height* (Browsergröße) statt *device-width* und *device-height* (Bildschirmgröße) zu bevorzugen, da die MQs in diesem Falle auch bei der Größenänderung des Desktop-Browsers greifen, was eine stufenlose Anpassung ermöglicht.

Die Angabe der Breite bzw. Höhe kann unabhängig von der Pixeldichte bzw. Auflösung erfolgen. Viele mobile Geräte verfügen über die doppelte Anzahl an Pixel im Vergleich zu normalen Desktop-Bildschirmen: Apples iPhone 4 hat beispielsweise eine doppelt so hohe Auflösung wie das Vorgängermodel iPhone 3. Die vielfachen „Gerätepixel" entsprechen aber nach wie vor einem „CSS-Pixel". Dadurch ist gewährleistet, dass ein Element von 100 Pixel Breite auf beiden Geräten die gleiche physikalische Größe einnimmt[66].

Es wird erkennbar, dass mit relativ wenig Aufwand eine flexible, dynamische Darstellung kontextübergreifend nach den Prinzipien des MFRWD möglich ist.

65 Maurice, Florence. S. 247
66 Zillgens, Christoph. S. 49

4.2.2.3. Responsive Typography

Neben der Makro- ist auch die Mikroebene der Layoutgestaltung in MFRWD von Relevanz. Nicht nur die prozentuale Definition von ganzen Layoutblöcken, sondern auch von Typografie ist möglich und empfehlenswert.

Bisher wurden Schriftgrößen hauptsächlich in absoluten Pixelwerten definiert. Im MFRWD sollte auf relative Angaben zurückgegriffen werden. Das hat den Grund, dass es zum einen Konsistenz in Bezug auf die ansonsten verwendeten prozentualen Angaben schafft, zum anderen ist es dadurch möglich, Schriftgrößen websiteübergreifend in Relation zu setzen. Dies ist vor allem bei größeren Projekten sinnvoll[67].

```
h1 {
    font-size: 2em;
 }
```

Obenstehende Zeile definiert für eine Headline, ausgehend von seinem Eltern-Element, die 2-fache Schriftgröße. Ist in keinem Eltern-Element eine Schriftgröße definiert, so wird von der Basisschriftgröße, welche durch die Browserhersteller vorgeben wird, ausgegangen. Diese beträgt in der Regel 16 Pixel[68]. Dementsprechend hätte die Headline hier eine Schriftgröße von 32 Pixel.

Kompliziert wird es, wenn die Anzahl der Verschachtlung und die dazugehörigen Schriftdefinitionen zunehmen. Oftmals wird der Überblick, auf welches Eltern-Element sich gerade bezogen wird, verloren. Abhilfe schafft hier die in CSS3 integrierte Einheit *rem*[69] (root em). Diese bezieht sich immer auf die Schriftgröße des root/html Elements.

```
html {
    font-size: 16px;
}

.container {
  font-size: 14px;
}

.container h2 {
  font-size: 1em; /* Entspricht 14px, bezieht sich auf .container */
}

h1 {
    font-size: 1rem; /* Entspricht 16px, bezieht sich auf root/html */
 }
}
```

Die Wahl relativer Schriftgrößen ermöglicht also eine sehr schnelle Anpassung an unterschied-

67 Zillgens, Christoph. S. 25
68 Ebenda.
69 W3C Candidate Recomendation. CSS Values and Units Module Level 3. http://www.w3.org/TR/css3-values/#font-relative-lengths (Stand 07.07.2013)

liche Bildschirmgrößen, indem die Schriftgröße des Eltern-Elements geändert wird. Welche Einheit, ob *em* oder *rem*, gewählt wird, hängt von den persönlichen Vorlieben und vom jeweiligen Projekt ab.

Des Weiteren ist es wichtig, die richtige Schriftgröße in Bezug auf den Kontext zu wählen. Es gilt einen Kompromiss zwischen Lesbarkeit und Raumbedarf zu finden. Im mobilen Kontext beispielsweise, kann die Schriftgröße des Fließtextes etwas kleiner gewählt werden, als auf stationären Geräten, da der Abstand der Augen zum Bildschirm verringert ist.[70] Weitere Parameter die beachtet werden müssen sind Zeilenlänge und Zeilenabstand, da diese die Lesbarkeit beeinflussen. Eine nähere Betrachtung responsiver Typografie ist aber aus Gründen des Umfangs dieser Arbeit nicht möglich.

4.2.3. Kritik und Ausblick

Die Hauptkritik an der Art und Weise des Layoutens in MFRWD bezieht sich auf die Tatsache, dass das Layout immer an die Struktur des HTML-Markups gebunden ist. Mit den aktuell verwendeten Techniken ist es lediglich möglich, über *floats* und den *display-styles* Einfluss auf die Anordnung der einzelnen Elemente zu nehmen, sind sie einmal fest definiert. Das Layout ist nach seiner Ausgestaltung also relativ unflexibel und die Möglichkeit Elemente komplett neu zu arrangieren nicht gegeben. Die Kritik ist dem Wunsch erwachsen, den Anforderungen des jeweiligen Umfelds besser gerecht werden zu können. Beispielsweise im mobilen Kontext sollte es möglich sein, Elemente an einer physikalisch komplett anderen Position anzuzeigen, als auf dem Desktop, was nachfolgende Grafik verdeutlicht. Hier ist die Navigation im stationären Umfeld bei entsprechender Breite an einer anderen Stelle positioniert, als im mobilen:

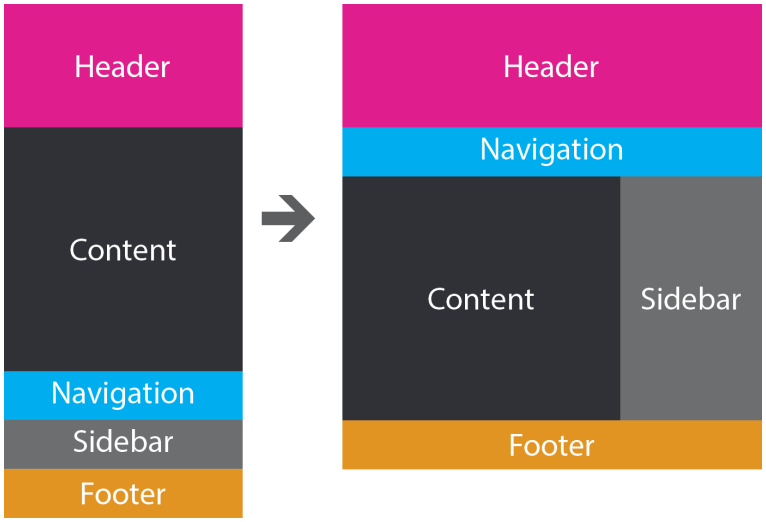


Abbildung 14 - Erstrebenswerte Möglichkeit zur Umsortierung von Elementen

Browserübergreifend ist dies momentan nur durch zwei Techniken zuverlässig möglich: Zum einem, indem Inhalte doppelt angelegt werden und dann über MQs ab einem bestimmten Breakpoint ein- oder ausgeblendet werden. Dabei muss natürlich mehr Markup und somit mehr

70 Zillgens, Christoph. S. 181

Ladezeit in Kauf genommen werden. Zum anderen mit JS-basierten Lösungen[71], die Elemente aus dem DOM auslösen und an anderer Stelle wieder einfügen.

In den Spezifikation für CSS3 finden sich jedoch vielversprechende Ansätze, die Abhilfe für dieses Problem schaffen sollen. Eine dieser potentiellen Lösungen ist das sogenannte „Flexible Box Layout Model[72]".

Vereinfacht gesagt werden hierbei lediglich die Abstände und die Größenverhältnisse der Boxen zueinander definiert. Der Browser berechnet dann die endgültige Größe der Boxen mittels der definierten Verhältnisse und des zur Verfügung stehenden Raumes innerhalb des Elternelements. Ändert sich das Umfeld und somit die Bildschirmgröße, werden die Boxen proportional angepasst. Sie sind also ohne die prozentuale Angabe der Breite flexibel. Die Syntax stellt sich wie folgt dar:

```
.parent {
   display: flex;
   display: -webkit-flex;
}

.child-1 {
  flex: 1;
}
/* 2x so breit wie child-1*/
.child-2 {
  flex: 2;
}
```

Abbildung 15 - Beispiel Flex-Box

Eine Umstrukturierung kann erreicht werden, in dem eine Sortierreihenfolge definiert wird. Viele weitere nützliche Einstellungen wie die Ausrichtung als Zeile oder die Umkehrung der Schreibrichtung, beispielsweise für asiatische oder arabische Inhalte, sind möglich.

Die Browserunterstützung für „Flexbox" ist relativ gut[73], jedoch können die CSS-Angaben bisher häufig nur mit Vendor-Prefixes (-webkit-flexbox, -moz-flexbox), also browser-spezifischer Syntax definiert werden, da die Implementierungen auf keinem verabschiedeten Standard beruhen. Daher ist es auch möglich, dass sich die Syntax, wie bereits mehrmalig geschehen, erneut ändert. Zusammenfassend lässt sich sagen, dass es sich bei „Flexbox" um einen äußert interessanten Ansatz handelt, es aber sehr wahrscheinlich noch einige Zeit in Anspruch nehmen wird, bis eine Verabschiedung seitens des W3Cs und eine standardisierte Implementierung der Browserhersteller erfolgen kann.

71 relocate.js. Helfer um Elemente innerhalb des DOMs zu verschieben. https://github.com/edenspiekermann/minwidth-relocate (Stand 09.07.2013)
72 W3C Candiate Recommendation. CSS Flexible Box Layout Module. http://www.w3.org/TR/css3-flexbox/ (Stand 09.07.2013)
73 Browserunterstützung von Flexbox. http://caniuse.com/#feat=flexbox (Stand 10.07.2013)

Eine weitere Möglichkeiten das Layout von Websites zu gestalten, bietet das „CSS3 Template Layout“[74]. Mit diesem können Elemente auf einem unsichtbaren, definierbaren Grid ausgerichtet werden, indem ihnen eine Position in selbigem zugewiesen wird.

```
.parent {
   display: „ab“
            „cd“
}

.child-1 {
  position: a;
}

.child-2 {
  position: b;
}
```

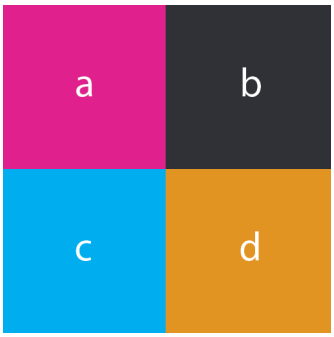


Abbildung 16 - CSS3 Template Layout

Eine der signifikantesten Schwächen des „Template Layouts“ ist die fehlende Möglichkeit Abstände zwischen den einzelnen Bereichen des Grids zu definieren.

Es gibt noch weitere Techniken, die zukünftig eine Rolle spielen könnten, wie beispielsweise die „CSS3 Multiple Columns“[75] oder das „CSS3 Grid Layout“[76], welches eine Art Erweiterung des „Template Layouts“ darstellt. Deren Verabschiedung und übergreifende Browserimplentierung steht jedoch noch aus.

Bis die vielversprechenden Ansätze tatsächlich Realität werden, wird es noch dauern. Bis dahin muss mit den bisherigen Möglichkeiten vorlieb genommen werden, die trotz aller Kritik, sinnvolle und stabile Lösungen darstellen.

4.3. Navigation

4.3.1. Anspruch und Grundlagen

Der wichtigste Grundsatz bei der Gestaltung und Umsetzung einer MFRWD-Navigation lautet „Content over Navigation“[77], Inhalt ist wichtiger als Navigation. Das heißt Inhalte müssen eine höhere Priorität besitzen und somit auch mehr Raum einnehmen als die Navigation, weil sie – im mobilen Umfeld umso mehr – den Grund für den Aufruf eines Internetauftritts darstellen. Der Besuch einer Website im mobilen Kontext findet, wie bereits in *Abschnitt 3.2.3 Zeitliche Begrenzung* dargelegt, in kurzen Intervallen statt. Daher hat der User wenig Zeit noch Motivation sich mit der

74 W3C Working Draft. CSS Template Layout Module. http://www.w3.org/TR/2009/WD-css3-layout-20090402/ (Stand 10.07.2013)
75 W3C Candidate Recommendation. CSS Multi-column Layout Module. http://www.w3.org/TR/css3-multicol/ (Stand 10.07.2013)
76 W3C Working Draft. CSS Grid Layout. http://www.w3.org/TR/css3-grid-layout/ (Stand 10.07.2013)
77 Wroblewski, Luke. Mobile First. S. 49

Navigation zu beschäftigen, sondern sollte unmittelbar an die für ihn relevante Information gelangen können, egal ob er Aktienkurse überprüfen oder sich über das Wetter informieren möchte.

Im mobilen Umfeld kann bereits eine Navigation mit vier bis fünf Navigationspunkten, wenn sie falsch und nicht kontextgerecht aufbereitet wurde, den eigentlichen Inhalt verdrängen und die Steuerung der Website erschweren. Dabei ist es möglich, dass der Benutzer auf den ersten Blick nicht einmal weiß, welchen Zweck die Website erfüllen soll oder nur durch die Änderung des Aktivzustands in der raumgreifenden Navigation erkennt, dass er eine Seite gewechselt hat.

In den meisten Fällen muss daher das eigentliche Menü versteckt und durch ein einfach aufzufindendes, repräsentatives Icon ersetzt werden. Durch Klick des Benutzers auf dieses, kann die Navigation angezeigt werden.

Hierfür gibt es verschiedene Techniken und Strategien, was in nachfolgendem Kapitel näher beleuchtet wird. Wird der mobile Kontext verlassen und steht ausreichend Platz zur Verfügung, kann die Navigation in vollem Ausmaße angezeigt werden[78].

Abbildung 17 - Transformation einer Navigation von mobilen hinzu stationären Kontext

Den Unterschied zwischen schlechter und gelungener mobiler Navigation zeigt nachfolgende Abbildung[79]:

Abbildung 18 - Schlechtes und gelungenes Beispiel einer MFRWD-Navigation

78 Screenshot der Website Heimatland http://heimatland-entdecken.de/ (Stand 13.07.2013)
79 Screenshots der Websites fiafo und Heimatland. http://www.fiafo.com/ und http://heimatland-entdecken.de/ (Stand 13.07.2013)

Im linken Screenshot ist im Vergleich zum rechten bei gleicher Seitenhöhe keinerlei Inhalt zu sehen. Dadurch ist es für den User auf den ersten Blick beinahe unmöglich überhaupt festzustellen, um welche Art von Website es sich handelt. Der rechte Screenshot hingegen zeigt bereits ein mobil optimiertes Keyvisual, Headline und Teile des Fließtexts, da die Navigation ausgeblendet ist und bietet somit bereits im ersten Moment deutlich mehr Information.

Es gilt also, die Anzahl der Navigationspunkte gering, somit die Navigation einfach, logisch sowie klar strukturiert zu halten und sich auf das Wesentliche zu fokussieren. Werden aufgrund des Platzmangels im mobilen Umfeld einfach Navigationspunkte weggelassen, so stellt sich die Frage, ob diese auf dem Desktop überhaupt eine Daseinsberechtigung haben[80]. Aus der Beschränktheit der Navigation erwächst jedoch für den Inhalt an sich ein Vorteil, da dieser sich in seiner Klarheit und Kürze an selbige anpassen muss, was Thema des *Kapitels 4.5* ist.

Unabhängig von der Anzahl der Navigationspunkte muss des Weiteren bedacht werden, dass sich die Art der Interaktion im mobilen Umfeld komplett ändert. Die Steuerung der Website erfolgt zumeist über Touch-Displays. Daher ist es wichtig, klickbare Flächen für die Größe der agierenden Finger zu optimieren um eine problemlose Bedienbarkeit zu gewährleisten. Während man auf dem Desktop mit einer Maus relativ präzise Navigationspunkte ansteuern kann, werden diese im mobilen Umfeld mit einem einzigen Finger sogar oftmals überlagert. So empfehlen Apple[81] und Microsoft[82] (siehe nachfolgende Abbildung[83]) in ihren Interface Guidelines für klickbare Flächen eine durchschnittliche Mindesthöhe- und breite von 7-10 mm (zwischen 22 und 31 CSS Pixel) mit einem Mindestabstand von 2 mm (rund 7 CSS Pixel), da laut Nokia eine durchschnittliche Fingerspitzengröße ca. 10x10 mm misst[84]. Neuere Untersuchungen des MIT gehen sogar von einer Fingerspitzengröße von 15-20 mm aus, woraus 45 - 57 CSS Pixel resultieren[85].

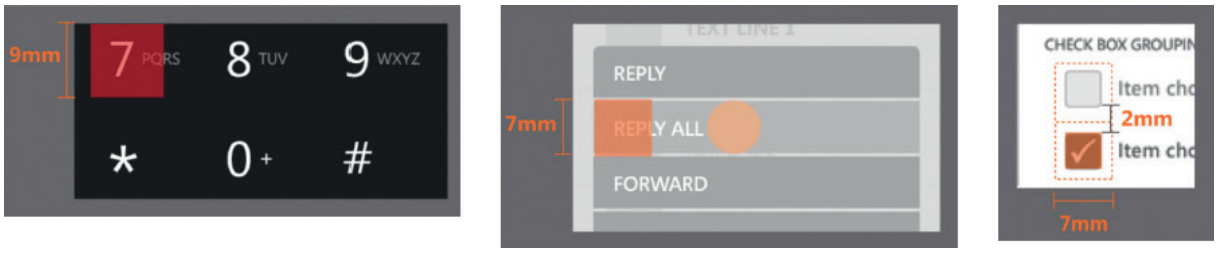


Abbildung 19 - Ausschnitt aus den User Experience Guidelines von Microsoft

Diese Maße sind folgerichtig auch bei der Gestaltung von Buttons und anderen Eingabeelementen anzuwenden. Es gilt also, je größer desto besser.

80 Zillgens, Christoph. S. 233
81 iOS Human Interface Guidlines. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html (Stand 14.07.2013)
82 Windows Phone UI Design and Interaction Guide. http://go.microsoft.com/?linkid=9713252 (Stand 14.07.2013)
83 Screenshot aus den Windows Phone UI Design and Interaction Guide.
84 Wroblewski, Luke. Mobile First. S. 69
85 Finger-Friendly-Design. http://uxdesign.smashingmagazine.com/2012/02/21/finger-friendly-design-ideal-mobile-touchscreen-target-sizes/ (Stand 14.07.2013)

4.3.2. Aktuelle Navigationskonzepte

Bei der Gestaltung und Realisierung von Navigationen nach MFRWD haben sich bestimmte Strategien und Techniken bewährt, ähnlich wie die bereits beleuchteten Layout Pattern. Das Prinzip nachfolgend vorgestellter Realisierungen ist immer das gleiche: Abhängig vom Platzbedarf und vom jeweiligen Kontext werden MQs definiert, welche nach Überschreitung ihrer Breakpoints greifen und die Navigation transformieren oder die umfeldspezifischen Lösungen ein- und das bisherige Menü ausblenden. Die Herausforderung besteht darin, eine Navigation mit optimaler Bedienbarkeit bei gleichzeitig minimalen Raumbedarf zu gestalten. Einige der Konzepte können mit reinem HTML und CSS realisiert werden, andere benötigen zusätzlich JS.

4.3.2.1. Top-Navigation

Dieses Navigationskonzept erfordert den wenigsten technischen Aufwand, da das horizontale Menü im mobilen Kontext prinzipiell genau so aussieht wie auf dem Desktop. Hier gilt es lediglich die Abstände und die Größe der Klickflächen anzupassen. Viele Websites mit überschaubaren Inhalt nutzen diese Herangehensweise, so auch der Auftritt des Webdesigners Javier Lo[86].

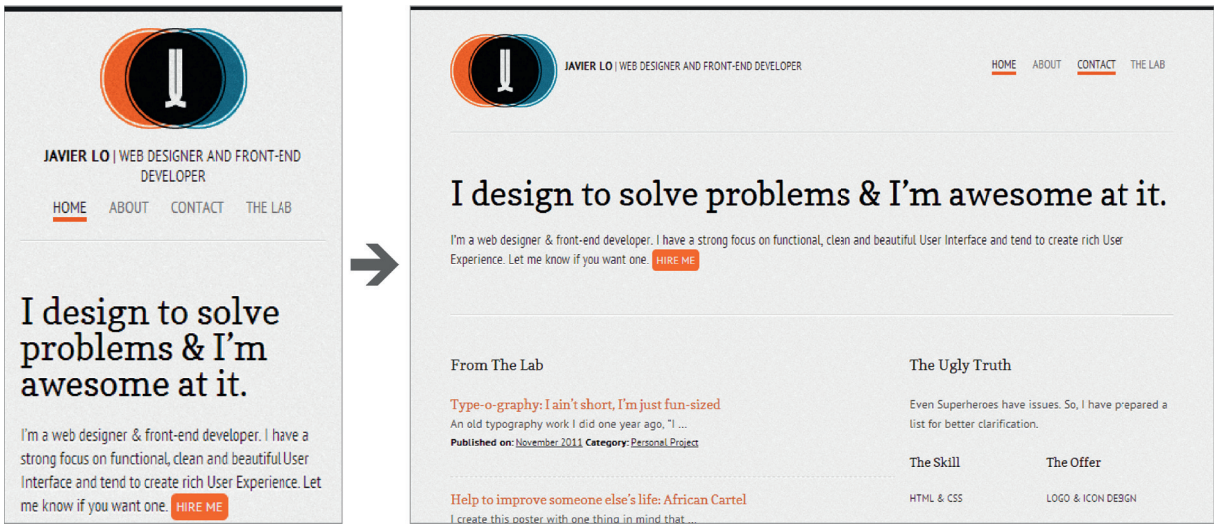


Abbildung 20 - Navigationskonzept Top-Nav

Vorteile

- Einfache Implementierung mittels HTML und CSS
- Kein JS notwendig

Nachteile

- Nimmt sehr viel Platz im Kopfbereich der Website ein
- Problematisch bei der Erweiterung der Navigation um weitere Punkte
- Kaum Abstand zwischen den einzelnen Navigationspunkten
- Schwierige, ungenaue Bedienung im mobilen Kontext
- Unterschiedliche Schriftdarstellung der Browser kann die Navigation umbrechen

86 Screenshot der Website von Javier Lo. http://www.javierlo.com/ (Stand 13.07.2013)

Die Verwendung dieses Navigationskonzepts empfiehlt sich nur, wenn man sich sicher ist, dass keine weiteren Navigationspunkte hinzugefügt werden und daher die Klickflächen in eine akzeptable Größe skaliert werden können. Ein Testen auf verschiedenen Geräten und Browsern ist aufgrund der genannten Problematik bezüglich der unterschiedlichen Schriftdarstellung unverzichtbar.

4.3.2.2. Select-Menu

Das „Select-Menu" verwandelt auf kleinen Auflösungen die Navigation in eine Select-Box und verhindert somit das Problem der raumgreifenden Navigation, wie es bei der „Top-Navigation" der Fall ist. Es entstammt der Anfangszeit des RWD und wird daher nur noch relativ selten verwendet. Das Beispiel zeigt die Website des Entwicklers Viljami Salminen[87].

Abbildung 21 - Navigationskonzept Select-Menu

Vorteile

- Sehr platzsparend
- Navigation komplett im Header der Website zu finden
- Relativ einfach zu realisieren
- Leicht als Navigation zu erkennen, da die erste Option der Select-Box beispielsweise mit „Menü" kenntlich gemacht werden kann
- Optionsauswahl der Select-Box nativ durch den Browser implementiert

Nachteile

- Select-Box mit reinem CSS kaum zu stylen, wirkt wie ein Fremdkörper
- Eventuelle Subnavigation nicht abzubilden
- Kann verwirrend wirken, da man ein Formularelement im Kopf der Website nicht erwartet
- Semantischer Missbrauch eines HTML-Elements
- Benötigt den Einsatz von JS

87 Screenshot der Website von Viljami Salminen. http://viljamis.com/ (Stand 14.07.2013)

Um das Konzept „Select-Menu“ zu implementieren bietet sich die Nutzung des leichtgewichtigen jQuery-Plugins „Tiny Nav“[88] an. Dieses ist durch einen einzigen Methodenaufruf zu aktivieren und bietet diverse Konfigurationsmöglichkeiten.

4.3.2.3. Footer-Anchor

Dieses Konzept verlagert die Navigation in den Footer, also ans Ende der Website. Sie ist über einen simplen Ankersprung zu erreichen, in dem auf einen im Header positionierten Link oder Icon geklickt wird, wie die Website von Luke Wroblewski demonstriert[89].

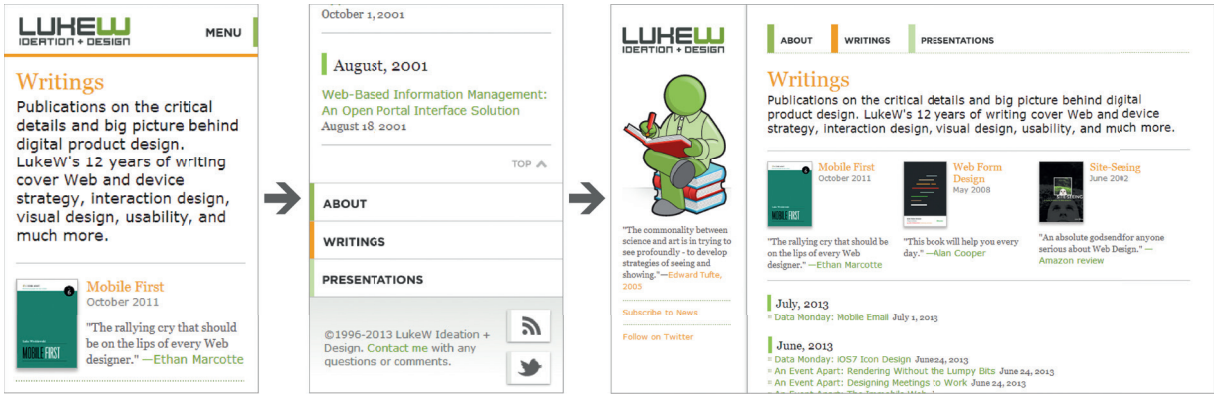


Abbildung 22 - Navigationskonzept Footer-Anchor

Vorteile

- Sehr einfache Implementierung mittels Ankersprung und MQ
- Kein JS notwendig
- Schafft viel Freiraum im Header der Website, Content First optimal realisierbar
- Alleinstehende Navigation am Ende der Website
- Abbildung einer Subnavigation möglich
- Große Klickflächen realisierbar

Nachteile

- Der plötzliche Sprung ans Ende der Seite kann zu Verwirrung beim User führen
- Wirkt unelegant

Die Vorteile die das Konzept des „Footer-Anchors“ bietet, wiegen schwerer als die wenigen Nachteile. Zudem empfiehlt es sich einen Ankersprung zurück an den Anfang der Website zu integrieren, falls der User die Seite doch nicht wechseln möchte, um ihm das umständliche Rückwärts-Scrollen zu ersparen.

88 jQuery-Plugin „Tiny Nav“. http://tinynav.viljamis.com/ (Stand 14.07.2013)
89 Screenshot der Website von Luke Wroblewski. http://www.lukew.com/ff (Stand 14.07.2013)

4.3.2.4. Toggle-Menu

Das „Toggle-Menu“ besticht einerseits durch den geschaffenen Raum, andererseits durch seine elegante Integration in das Gesamtbild einer Website. Die Navigation ist anfangs komplett versteckt, lediglich ein platzsparendes Icon weißt auf sie hin. Wird dieses geklickt, so gelangt, wie im obigen Beispiel der schwedischen UNICEF dargestellt[90], die Navigation in den Vordergrund. Dabei wird der eigentliche Inhalt wahlweise nach unten geschoben oder überlagert. Die Anzeige der Navigation geht meist mit einer Animation einher, was dieses Konzept wertig erscheinen lässt.

Abbildung 23 - Navigationskonzept Toggle-Menu

Vorteile

- Kein Sprung ans Seitenende notwendig
- Platzsparen, schafft viel Freiraum für andere Gestaltung
- Vollständige gestalterische Kontrolle
- Elegante und hochwertige Wirkung
- Größtenteils durch HTML und CSS zu realisieren
- Abbildung einer Subnavigation möglich
- Große Klickflächen realisierbar

Nachteile

- Performance kann auf älteren Geräten unter der Animation leiden
- Benötigt den (minimalen) Einsatz von JS
- Mehraufwand bei der Gestaltung

Das „Toggle-Menu“ findet sich wegen der zahlreichen Vorteile auf einer stetig zunehmenden Anzahl von Websites wieder. Des Weiteren lässt es sich durch die Verwendung von Plugins wie „FlexNav“[91] oder das von anderen Bibliotheken unabhängige „Responsive Nav“[92] sehr einfach realisieren.

90 Screenshot der Website der schwedischen Unicef. http://unicef.se/ (Stand 15.07.2013)
91 jQuery-Plugin „FlexNav“. http://jasonweaver.name/lab/flexiblenavigation/ (Stand 15.07.2013)
92 Plugin „Responsive Nav“. https://github.com/viljamis/responsive-nav.js (Stand 15.07.2013)

4.3.2.5. Off-Canvas

„Off-Canvas“ bezeichnet nicht nur ein Layout Pattern, sondern ebenso ein damit einhergehendes Navigationskonzept. Wenn dieses in Reinform implementiert wird, befindet sich die Navigation im Desktop-Kontext auf der linken Seite und wird beim Übergang in das mobile Umfeld außerhalb des sichtbaren Bereichs, also „Off-Canvas“, positioniert. Viele Websites interpretieren das klassische Konzept aber ein wenig anders und verstecken auf kleinen Auflösungen auch horizontale Menüs außerhalb des Viewports, wie das nachfolgende Beispiel der Website von Heimatland zeigt[93]. Wird auf das in der linken oberen Ecke angebrachte Icon geklickt, so schiebt die Navigation von links her kommend den Inhalt zur Seite.

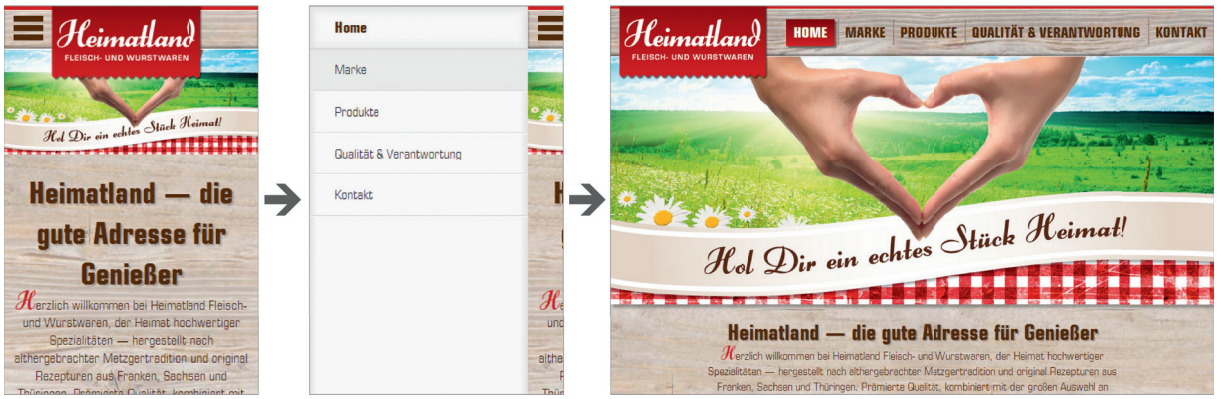


Abbildung 24 - Navigationskonzept Off-Canvas

Vorteile

- Schafft viel Freiraum für andere Gestaltung
- Optimale Lösung für viele Navigationspunkte
- Wirkt durch Animation hochwertig und elegant
- Dem User durch die häufige Verwendung in zahlreichen Websites und auch in nativen Apps sehr geläufig

Nachteile

- Performance intensiv
- Kann auf den ersten Blick verwirren
- Relativ aufwendige Implementierung
- Benötigt den Einsatz von JS
- Wenn nicht als reines „Off-Canvas“ implementiert, aufwendige Transformation hinzu anderem Kontext

Nachdem Facebook das Konzept als eine der ersten Websites implementiert hatte, folgten zahlreiche Entwickler dem Beispiel. Die Implementierung ist relativ komplex, da mit absoluter Positionierung gearbeitet werden muss. Der Einsatz von JS ist unumgänglich. Die Plugins „Sidr“[94],

93 Screenshot der Website von Heimatland. http://heimatland-entdecken.de/ (Stand 16.07.2013)
94 jQuery-Plugin „Sidr“. http://www.berriart.com/sidr/ (Stand 16.07.2013)

„offCanvasMenu“[95] oder die komplette Implementierung des Layout Patterns wie es Jason Weaver auf einer Demoseite[96] demonstriert, helfen bei der Umsetzung des Navigationskonzeptes.

4.3.3. Probleme und Ausblick

Das Hauptproblem der Realisierung einer Navigation nach MFRWD besteht in der Unterschiedlichkeit der Umsetzungen. Der Benutzer muss sich bei jedem Besuch einer Website aufs Neue mit der Navigation beschäftigen, die unter Umständen immer unterschiedlich implementiert ist. Dies kann zur Verwirrung und zu Schwierigkeiten bei der Bedienung führen.

Werden Konzepte eingesetzt, die im mobilen Umfeld einen Link oder Icon an Stelle der Navigation darstellen, besteht das Problem der uneinheitlichen Darstellungen derselben:

Abbildung 25 - Unterschiedliche Darstellung des Navigation-Buttons

Es stellt sich die Frage, welches dieser Icons bzw. Buttons am ehesten erkennbar ist. Da es keinen Standard gibt, sind die hier dargestellten Varianten lediglich ein exemplarischer Auszug aus der breiten Varietät, die sich nach kurzer Recherche auftut. Da das W3C sich lediglich mit konkreten Implementierungen und nicht mit Empfehlungen bezüglich Gestaltung und Usability beschäftigt, wird die Schwierigkeit solange bestehen, bis sich die Entwicklergemeinde auf eines dieser Icons einigt. Ein technischer Standard seitens des W3Cs ist ebenfalls nicht in Aussicht.

Jedoch ist es mit den vier dargelegten Navigationskonzepten möglich, vernünftige mobile Lösungen zu implementieren. Dabei scheinen sich die Konzepte „Toggle-Menu“ und „Off-Canvas“ aufgrund ihrer Wertigkeit durchzusetzen, aber auch der Ansatz des „Footer Anchors“ stellt eine interessante Option dar, legt man hohen Wert auf Performance und soll der Einsatz von zusätzlichem JS vermieden werden. Des Weiteren ist es mit diesen Realisierungen möglich, auch Subnavigationen beispielsweise in Form von aufklappenden Akkordeons[97] abzubilden.

Die anderen beiden Herangehensweisen „Top-Navigation“ sowie „Select- Menu“ empfehlen sich nur, wenn eine Navigation wenige Unterpunkte benötigt und sind im Vergleich zu den drei anderen, die für den mobilen Kontext am wenigsten geeigneten Realisierungen.

95 jQuery-Plugin „offCanvasMenu“. https://github.com/cloudfour/offCanvasMenu (Stand 16.07.2013)
96 Demoseite zu „Off-Canvas“. http://jasonweaver.name/lab/offcanvas/ (Stand 16.07.2013)
97 Beispiel zur Realisierung einer responsiven Subnavigation. http://webdesigntutsplus.s3.amazonaws.com/tuts/378_tessa/tessa-lt-dropdowns-21c7868/index.html (Stand 16.07.2013)

4.4. Bilder und Grafiken

4.4.1. Anforderungen im MFRWD

Bilder und Grafiken spielen in der Gestaltung und Usability von Websites eine enorm große Rolle. Zum einen erleichtern sie an vielen Stellen die Bedienbarkeit durch die Visualisierung von komplexen Sachverhalten. Man denke an das Navigationsicon des vorangegangen Kapitels oder an Piktogramme wie die Lupe einer Suche oder das Schloss für geschützte Bereiche. Zum anderen sind sie für die Auflockerung textlicher Inhalte oder für das Storytelling innerhalb einer Website unerlässlich. Der Trend zeigt eine klare Tendenz von text- hinzu bildlastigen Websites[98]. Um es mit einer alten Metapher auszudrücken: „Ein Bild sagt mehr als tausend Worte".

Daher ist es eine der wichtigsten Aufgaben sowie zugleich eine der größten Herausforderungen des MFRWD Bilder und Grafiken kontextgerecht darzustellen, was sich wie im Folgenden dargelegt, nicht immer als einfach erweist.

4.4.1.1. Geringe Ladezeit

Wird vom Desktop-Kontext ausgegangen spielen die Kriterien Datenmenge, Datenrate und Geschwindigkeit eine geringere Rolle. Erfolgt der Aufruf einer Website mittels Laptop oder stationärem Rechner, kann meist davon ausgegangen werden, dass eine ausreichend schnelle Internetverbindung verfügbar ist[99]. Im mobilen Kontext ist die Verfügbarkeit einer schnellen Leitung wesentlich unwahrscheinlicher, da die Verbindungsgeschwindigkeit vom aktuellen Standort und dem Mobilfunkvertrags des Nutzers abhängig ist. Bilder und Grafiken müssen im MFRWD also eine geringe Datenmenge aufweisen, um schnell zur Verfügung gestellt werden zu können, bedenkt man, dass sie 62% der Gesamtdatenmenge[100] einer durchschnittlichen Website ausmachen. Dabei muss ein Kompromiss aus Kompression und Qualität gefunden werden. Ist ein zügiger Download und somit ein schneller Aufbau der Website möglich, so profitiert diese auch im Desktop-Kontext durch die gesteigerte Performance. Im mobilen Umfeld ist dies unerlässlich, da durch lange Ladezeiten die meisten Besucher verloren werden[101].

98 Hedmann, Falk. Webdesign Trends. 10 Dinge, die uns 2013 erwarten. http://t3n.de/news/webdesign-trends-2013-10-dinge-424796/ (Stand 17.07.2013)

99 Qualitätsstudie der Bundesnetzagentur zur Datenübertragung. http://www.bundesnetzagentur.de/cln_1912/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Breitband/Dienstequalitaet/qualitaetsstudie/qualitaetsstudie-node.html;jsessionid=A26A459969C2C61B7A1604F39A1B3801 (Stand 17.07.2013)

100 Internetstatistik zu Dateigrößen. http://httparchive.org/interesting.php (Stand 17.07.2013)

101 Work, Sean. How Loading Time Affects Your Bottom Line. http://blog.kissmetrics.com/loading-time/ (Stand 17.07.2013)

4.4.1.2. Anpassung an hochauflösende Displays

92.8% aller ME[102] verfügen mittlerweile über die mindestens 1.5-fache Pixeldichte eines herkömmlichen, stationären Bildschirms. Apple nennt diese Eigenschaften ihrer Geräte „Retina", Google verwendet den Terminus „XHDPI". Beide stehen für dasselbe – das Vielfache der gewöhnlichen Auflösung. So besitzen „Retina-Displays" beispielsweise die doppelte Pixeldichte. Daraus resultiert, dass ein Bild die 4-fache Anzahl an Pixel besitzen muss als bisher, um eine optimale Darstellung auf derartigen Bildschirmen zu gewährleisten, da sich die Verdopplung der Bildpunkte auf Breite und Höhe bezieht. Es muss also doppelt so groß wie bisher zur Verfügung gestellt werden.

Abbildung 26 - Darstellung von Bitmap-Pixel auf einem Retina-Display

Durch die Erhöhung der Pixeldichte kann das menschliche Auge zwischen den einzelnen Bildpunkten nicht mehr unterscheiden, Bilder wirken schärfer. Natürlich ist es auch möglich, Bilder mit „normaler" Auflösung auf Retina bzw. XHDPI-Bildschirmen anzuzeigen. Diese werden jedoch skaliert dargestellt: Ein Bildpixel wird dabei auf mehrere Gerätepixel verteilt, was eine Unschärfe zur Folge hat, was nachfolgende Abbildung zeigt[103]:

Abbildung 27 - Vergleich zwischen nicht-Retina und Retina Grafik

Eine Website kann dadurch massiv an Wertigkeit einbüßen. Daher ist die Darstellungsqualität auf hochauflösenden Displays, eine weitere Anforderung an Bilder und Grafiken im MFRWD.

102 Rupert, Dave. Ughck. Images. http://daverupert.com/2013/06/ughck-images/ (Stand 17.07.2013)
103 Abbildung vom Blog „Kulturbanause" von Jonas Hellig. http://blog.kulturbanause.de/2012/04/websites-und-bilder-fur-high-resolution-displays-retina-optimieren/ (Stand 17.07.2013)

4.4.1.3. Prozentuale Größendefinition

Da sich das gesamte Layout im MFRWD durch prozentuale Definitionen dem jeweiligen Kontext anpasst, muss dies natürlich auch für Bilder und Grafiken gelten. Bei Bildern, die direkt im Markup definiert sind, ist dies relativ einfach durch folgenden beispielhaften Code zu erreichen:

```
<figure>
   <img src="bildpfad/bild.jpg" alt="Bildbeschreibung" />
   <figcaption>Ein Bild</figcaption>
</figure>
```

```
figure {
   width: 50%; /* Beispielhafte Breite des umgebenden Elements */
}

img {
   max-width: 100%;
   height: auto;
}
```

Durch die Angabe von *max-width* passt sich das Bild immer flexibel im Verhältnis zu seinem übergeordneten Element an, überschreitet dessen Breite aber nie. Die Definiton *height: auto;* ist an sich nicht zwingend notwendig, da das Bild in modernen Browsern proportional skaliert wird, überschreibt aber eine etwaige Höhendefinition im Markup () , die ansonsten zu einer unproportionalen Skalierung führen könnte.

Werden Grafiken als Hintergrundbilder eingesetzt, ist eine flexible Skalierung ebenfalls möglich:

```
<div class="logo"></div>
```

```
.logo {
   width: 100%;
   height: 100%;
   background: transparent url(„logo.png") no-repeat 0 0;
   background-size: 100% auto;
}
```

Mit der CSS3 -Eigenschaft *background-size* kann die Größe des Hintergrundbildes prozentual definiert werden. Diese Attribut benötigt im Moment zwar noch die jeweiligen Vendor-Präfixe, wird aber ansonsten von den verschiedenen Browsern sehr gut unterstützt[104].

Handelt es sich bei den Hintergrundbildern um sich wiederholende Strukturen, ist die Anforderung der Skalierung natürlich hinfällig.

104 Browserunterstützung von background-size. http://caniuse.com/#feat=background-img-opts (Stand 17.07.2013)

Obige Codeblöcke sind lediglich als Beispiele zu interpretieren, zeigen jedoch auf, dass der Anspruch der prozentualen Größendefinition von Bilder und Grafiken in MFRWD, relativ einfach zu erfüllen ist.

4.4.2. Aktuelle Implementierungen

Die nachfolgend vorgestellten Implementierungen, versuchen allesamt auf ihre eigene Art und Weise den Anforderungen, die im vorherigen Kapitel definiert wurden, gerecht zu werden. Dabei gilt es hauptsächlich den Anspruch der geringen Ladezeit und der Darstellungsqualität auf hochauflösenden Displays gerecht zu werden, da wie bereits dargelegt, die prozentuale Skalierung mit verhältnismäßig geringem Aufwand zu realisieren ist. Die hier beleuchteten Lösungen sind lediglich ein exemplarischer Auszug aus einer Vielzahl von ähnlichen Realisierungen und sollen die prinzipiellen Herangehensweisen verständlich machen.

4.4.2.1. Meeting the Middle

„Meeting the Middle" ist eher als Art der Speicherung von JPGs zu verstehen, als eine tatsächliche technische Implementierung. Gerade darin liegt aber die Stärke, dieser Vorgehensweise begründet, da keinerlei zusätzlicher Code benötigt wird, außer man verwendet die Bilder als „Image-Sprite"[105], was eine Anpassung der *background-size* nach sich ziehen könnte. Sie basiert auf den Beobachtungen des niederländischen Webdesigners Daan Jobsis, dass eine hohe Kompressionsrate bei hochauflösenden Bildern weniger Einfluss auf die Qualität nimmt, da durch die höhere Pixeldichte Artefakte skaliert werden und somit kaum ins Auge fallen[106]. Ergänzt wird dieses Vorgehen von dem Vorschlag, statt der vollen 2-fachen für Apple-Geräte, lediglich die 1.5-fache Auflösung als Kompromiss zu verwenden[107] und diese Bilder auch auf normalen Displays einzusetzen. Es wird also keinerlei Unterscheidung zwischen den Arten der Displays getroffen. So wird auf hochauflösenden Displays eine höhere Bildqualität erreicht, ohne zu Lasten der Dateigröße bei normalauflösenden Bildschirmen zu gehen.

Das hat zudem den Vorteil, dass im Vergleich zu anderen Implementierungen, keine multiplen Bilder – also Varianten für die unterschiedlich auflösenden Displays – erzeugt werden müssen.

Folgende Abbildungen zeigen exemplarisch (Maße der Testdateien entsprechen nicht den hier Abgebildeten) die erreichbaren Dateigrößenunterschiede, wenn Bilder mit obig beschriebener Methode gespeichert werden. Für einen tatsächlich erlebbaren Vergleich empfiehlt es sich, die Blogveröffentlichung von Daan Jobsis mit einem hochauflösenden Display zu betrachten.

105 CSS Image Sprite. http://www.w3schools.com/css/css_image_sprites.asp (Stand 18.07.2013)
106 Jobsis, Daan. Retina Revolution. http://blog.netvlies.nl/design-interactie/retina-revolution/ (Stand 18.07.2013)
107 Matanich, Tyson. Images in a responsive Web. http://www.matanich.com/2012/11/06/picture-polyfill/ (Stand 18.07.2013)

Standardauflösung (300x300 Pixel)	**1.5-fache Auflösung** (450x400 Pixel)

Abbildung 28 - JPG Qualität 80 / baseline / 47,9 KB

Abbildung 29 - JPG Qualität 80 / progressive / 92,7 KB

Abbildung 30 - JPG Qualität 80 / baseline / 47,9 KB

Abbildung 31 - JPG Qualität 40 / progressive / 34,8 KB

Abbildung 32 - JPG Qualität 80 / baseline / 47,9 KB

Abbildung 33 - JPG Qualität 25 / progressive / 27,3 KB

Man erkennt anhand der Bildabfolge, dass bereits bei einer JPG-Qualität von 40, was einer mittleren Qualität entspricht, das hochauflösende Bild eine kleinere Dateigröße als die Standardauflösung aufweist.

Außerdem empfiehlt es sich, JPGs progressiv zu speichern[108]. Im Gegensatz zu normal (baseline) gespeicherten JPGs werden diese nicht von oben nach unten, sondern in verschiedenen Quali-

108 Robson, Ann. Progressive jpegs: a new best practise. http://calendar.perfplanet.com/2012/progressive-jpegs-a-new-best-practice/ (Stand 18.07.2013)

tätsstufen gerendert. Das hat zur Folge, dass die Bilder wesentlich eher angezeigt werden, wenn auch zunächst in verminderter Qualität. Dennoch erwecken sie den Eindruck schneller geladen zu werden, was einen Vorteil bezüglich der UE nach sich zieht.

In Hinblick auf die zu erwartende Zunahme von hochauflösenden Displays auch auf stationären Rechnern (man denke an Apples MacBook Pro als Vorreiter) ist dieses Vorgehen eine folgerichtige Maßnahme in dem Bestreben zukunftssichere Websites mit MFRWD zu gestalten.

4.4.2.2. Vektorgrafiken - SVG und Icon-Fonts

Vekorgrafiken sind ein Ansatz, auflösungsunabhängige Bilder und Grafiken im MFRWD einzusetzen. Dieses Vorgehen empfiehlt sich speziell bei der Verwendung von Icons, UI-Elementen, Pattern und Vektor-Illustrationen. Es beschränkt sich also auf tatsächliche Vektorgrafiken und kann nicht für pixelbasierte Rastergrafiken eingesetzt werden. Der Vorteil der Vektorgrafiken liegt auf der Hand: Ändert sich die Auflösung des Bildschirms oder die Größe des umgebenden Containers und müssen die Grafiken somit angepasst werden, so geschieht dies ohne Qualitätsverlust, da Vektoren im Grund nichts anderes als mathematische Anweisungen sind, die neu berechnet werden.

Aktuell werden hauptsächlich zwei Arten von Vektorgrafiken eingesetzt: SVG und Icon-Fonts.

Bei SVG[109] (Scalable Vector Graphics) handelt es sich um ein vom W3C entwickeltes XML-basiertes Format, welches von einer Vielzahl an Browsern unterstützt wird[110]. Eine große Anzahl an frei verfügbaren Icon-Sets oder UI-Elementen liegen bereits als SVG vor oder können – insofern sie vektorisiert sind – einfach konvertiert werden. Die Verwendung ist die gleiche wie bei herkömmlichen pixelbasierten Grafiken, die Einbindung kann per *<img/>*-tag oder mittels *background-property* erfolgen Jedoch können SVGs auch in ihrer eigenen Schreibweise in das Markup eingebracht werden:

```
/* Im HTML-Markup */
<svg xmlns:svg="http://www.w3.org/2000/svg" xmlns="http://www.
w3.org/2000/svg" version="1.0">
....
</svg>
```

Icon-Fonts basieren auf einer bereits in CSS2 integrierten Technologie der Schrifteinbettung mittels *@font-face*, die es ermöglicht eigene Schriften in Websites zu integrieren. Zur damaligen Zeit bot jedoch lediglich der IE 5 diese Möglichkeit, da Schriften im proprietären .eot-Format (Embedded Open Type) vorliegen mussten, welches kein anderer Browserhersteller unterstützte[111].

109 W3C Recommendation. Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/2003/REC-SVG11-20030114/ (Stand 19.07.2013)
110 Browserunterstützung von SVG. http://caniuse.com/#feat=svg (Stand 19.07.2013)
111 CSS3-Info. Web-Fonts with @font-face. http://www.css3.info/preview/web-fonts-with-font-face/ (Stand 19.07.2013)

Mittlerweile haben andere Browser diese Spezifikation ebenfalls übernommen und ermöglichen nun auch die Verwendung gängigerer Schriftformate.[112]

Diesen Vorteil macht sich die Technik der Icon-Fonts zu nutze. Dabei werden Icons, die als Vektor vorliegen müssen, als eigene Schriftart definiert, welche dann über *@font-face* in die Website integriert wird. Dabei wird jedem Icon entweder eine Unicodenummer oder ein Zeichen zugewiesen, was die exemplarischen Code-Auschnitte zeigen:

```
<a href="#" class="twitter-icon"> Linktext </a>
<a href="#" data-icon="&#x27;"> Linktext </a>
```

```
@font-face {
    font-family: „iconfont“;
    src:url(„iconfont.otf“);
    ...
}

/* Verwendung mittels Klasse */
.twitter-icon:before {
   font-family: „iconfont“;
   content: „/27“;
}

/* Verwendung mittels data- Attribut */
[data-icon]:before {
   font-family: „iconfont“
   content: attr(data-icon);
}
```

Abbildung 34 - Beispiel zur Verwendung von Icon-Fonts

Mittlerweile haben sich zahlreiche Plattformen etabliert, die eine Fülle von Iconfonts und zusätzlich Schriftgeneratoren zur Erzeugung von Fonts aus eigenen Vektordaten anbieten. Beispielhaft sind hierbei „IcoMoon“[113] oder „Fontello“[114] zu nennen.

Zusammenfassend lässt sich sagen, dass es sich bei der Verwendung von Vektorgrafiken wie SVG oder Icon-Fonts um zeitgemäße und vorteilhafte Lösungen handelt. Aktuell gibt es keine alternative Möglichkeit, auflösungsunabhängige Grafiken derart komfortabel und performanceschonend zu integrieren. Jedoch beschränkt sich der Einsatz auf Icons, UI-Elemente oder ähnlich minimale Grafiken, da Detailreichtum die Dateigröße massiv nach oben treiben würde.

112 W3C Working Draft. CSS Fonts Module Level 3. http://www.w3.org/TR/css3-fonts/ (Stand 19.07.2013)
113 Icon-Sammlung und Schriftgenerator IconMoo. http://icomoon.io/ (Stand 19.07.2013)
114 tIcon-Sammlung und Schriftgenerator Fontello. http://fontello.com/ (Stand 19.07.2013)

4.4.2.3. Media Queries

Wie bei so vielen Anforderungen im MFRWD können MQ auch bei der Bereitstellung von kontextabhängigen Bilder und Grafiken gewinnbringend eingesetzt werden. Durch sinnvolle Kombination ihrer Abfrageparameter kann das jeweilige Umfeld erkannt und die Pixeldichte abgefragt werden. Auf Basis dieser Informationen ist es möglich, differenziert Bilder zur Verfügung zu stellen.

Dabei können die Grafiken entweder als Hintergrund mittels *background-property* oder direkt im Markup definiert werden:

```
<div class="logo"></div>
<img class="high_res" src="img.jpg" />
<img class="norm_res" src="img_@1.5x.jpg" />
```

```
/* Definition für hochauflösende Displays */
@media (min-device-pixel-ratio: 1.5) {
    .logo {
       background: url(„logo_@1.5x.png");
    }
    .high_res {
      display: block;
    }
    .norm_res {
       display: none;
    }
}

/* Definition für normale Displays */
.logo {
    background: url(„logo.png");
}
.high_res {
    display: none;
}
.norm_res {
    display: block;
}
```

In obigem Beispiel wird mit der MQ die Pixeldichte abgefragt (In der Praxis müssten noch die Vendor-Prefixes der Browserhersteller inkludiert werden) und mit Hilfe dieser Weiche festgelegt, welches Bild zur Verfügung gestellt wird. Dies geschieht zum einen durch den Austausch der Hintergrundgrafik in der Klasse *.logo* und zum anderen durch das Aus- bzw. Einblenden der Klassen *.norm_res* oder *.high_res*.

Das Anhängen des Kürzels *@1.5x*, dem sogenannten „High-Resolution Modifier“ von Apple[115], hat sich mittlerweile zum Quasi-Standard zur Bezeichnung von hochauflösenden Grafiken entwickelt.

Durch Ausbau der MQs ist es auch möglich, kleinere Bilder für bestimmte Bildschirmgrößen zur Verfügung zu stellen, was die zu herunterzuladende Datenmenge reduzieren kann. Ein Nachteil dieser Implementierung ist jedoch das Mehr an Codezeilen und die Tatsache, dass mindestens zwei Bilder erzeugt werden müssen. Ebenso müssen des Weiteren zwei Bildquellen geladen werden, wenn die Grafiken mittels *<img/>*-tag eingebunden und anschließend versteckt werden, was zu höherem, performance-mindernden Ladeaufwand führt.

4.4.2.4. JavaScript Lösungen

Lösungen mit JS greifen die Grundidee der im vorherigen Abschnitt beschriebenen Technik mit MQs auf. Zunächst werden die Parameter des aktuellen Umfeldes analysiert und auf Basis der gewonnenen Information verschiedene Versionen der Bilder zur Verfügung gestellt. So kann beispielsweise für hochauflösende Displays eine spezielle Grafik eingebunden werden:

```
$(document).ready(function(){
   var mq = „(min-device-pixel-ratio: 1.5)“;
   if(window.matchMedia(mq).matches){
     $(„img“).each(function(index, img){
       var $img = $(img),
         path = $img.attr(„src“),
         filetype = path.substr(-4),
         name = path.substr(0, path.length-4);
       $img.attr(„src“,name+“_@1.5x“+filetype);
     });
   }
});
```

Das obige Beispiel verwendet jQuery[116] als Hilfsbibliothek. Nachdem der DOM geladen wurde, wird mittels der *matchMedia*-Funktion eine vordefinierte MQ auf Gültigkeit überprüft. Trifft es zu, dass im aktuellen Kontext mindestens eine 1.5-fache Pixeldichte vorherrscht, so werden alle <img/>-tags durchlaufen und ihre Bildquellen mit dem Pfad zur hochauflösenden Grafik ersetzt. Der Codeausschnitt ist als exemplarisch zu interpretieren und bedürfte bis zum tatsächlichen Einsatz noch einiger Optimierung.

Eine andere Möglichkeit zur Bereitstellung von Bildvarianten bestünde in der Nutzung des d*ata-tags*:

115 Devloper Guidlines von Apple. Supporting High-Resolution Screens. http://developer.apple.com/library/ios/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/SupportingHiResScreensInViews/SupportingHiResScreensInViews.html#//apple_ref/doc/uid/TP40010156-CH15-SW1 (Stand 20.07.2013)

116 JavaScript Bibliothek jQuery. http://jquery.com/ (Stand 20.07.2013)

```
<img src="img.jpg" data-highres="img_@1.5x" />
```

Ein passendes JS müsste hierbei den Originalpfad mit der innerhalb des *data-tags* definierten Alternativquelle ersetzen.

Es wird aufgezeigt, dass es mittels JS möglich ist, kontextbezogen Bilder und Grafiken bereit zu stellen. Der Vorteil gegenüber einer reinen MQ-Lösungen liegt darin begründet, dass lediglich ein zentrales JS benötigt wird. Diese Art von Lösung erfüllt den Anspruch, hochauflösende Bilder zur Verfügung zu stellen. Jedoch müssen auch hier, jeweils mindestens zwei Varianten einer Grafik vorliegen, was einen größeren Aufwand nach sich zieht. Außerdem werden in der Summe in jedem Falle zwei Bilder geladen, da erst nach der Anzeige des Originalbildes dasselbige mit der hochauflösenden Variante ersetzt wird.

Als produktionsreife Implementierungen sind beispielsweise „retina.js“[117] oder „Interchange“[118] zu nennen.

4.4.2.5. Serverseitige Lösungen

Alle bisher vorgestellten Implementierungen haben sich auf die Ausführung auf dem Client beschränkt. Jedoch ist es auch möglich serverseitig Bilder zu generieren, die dem jeweiligen Kontext gerecht werden. Der prinzipielle Ablauf ist wie folgt:

Dem Server wird mitgeteilt für welchen Kontext ein Bild benötigt wird. Dies geschieht entweder über den User-Agent-String, der mit einem http-Request gesendet wird, oder über ein Cookie, dass per JS generiert und durch den Server ausgewertet wird. Per rewrite-rule werden sämtliche Anfragen die Bilder betreffen, an ein weiterverarbeitendes PHP-Skript geschickt.

In einer Variante der verfügbaren Lösungen wird nun auf dem Server, ausgehend von einem ausreichend großen Quellbild, eine kontextoptimierte Version erstellt und diese an den Client zurückgesendet. Das generierte Bild wird auf dem Server gespeichert und kann bei einer neuen Anfrage ohne erneute Generierung und somit schneller zur Verfügung gestellt werden.

In einer anderen Art der Implementierung wird kein neues Bild generiert, sondern aus einem Pool von kontextoptimierten Bildern das passende ausgewählt und an den Client übermittelt.

Der große Vorteil der Serverlösungen ist die Tatsache, dass lediglich eine einzige Bildquelle geladen werden muss. Dies hat positive Effekte auf die Performance der Website und erspart sogar, je nach eingesetzter Variante, das manuelle Erzeugen von Bildvarianten.

---

117 JavaScript für hochauflösende Bildervarianten retina.js. https://github.com/imulus/retinajs (Stand 20.07.2013)

118 JavaScript ür hochauflösende Bildervarianten Interchange. http://zurb.com/playground/foundation-interchange (Stand 20.07.2013)

Die Funktionsweise der Servertechnik ist in nachfolgender Grafik noch einmal dargestellt:

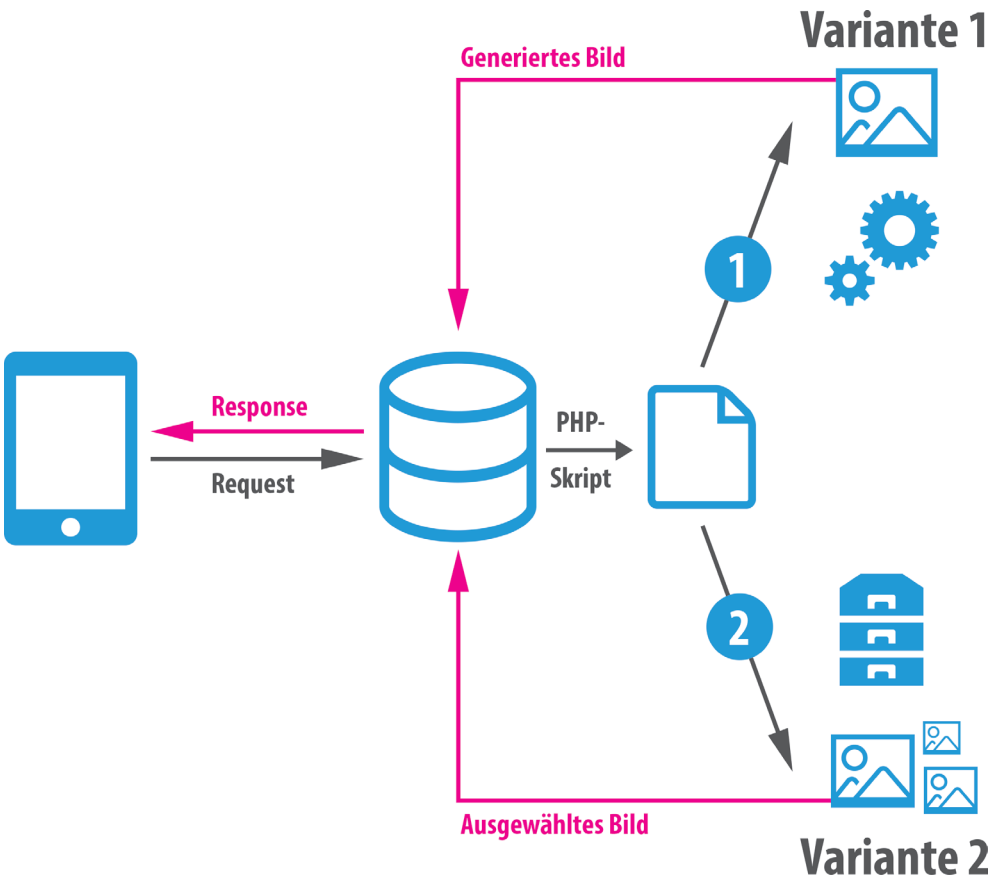


Abbildung 35 - Prinzip der serverseitigen, kontextabhängigen Bildgenerierung

Soll die Variante der dynamisch erzeugten Bilder eingesetzt werden, so empfiehlt sich der Einsatz von „Adaptive Images“[119], welche jedoch einen Apache-Webserver[120], PHP und den Einsatz der GD-Lib[121] benötigt.

Die zweite Art der Implementierung wird beispielsweise durch „Sencha.io src“[122] verwendet.

4.4.3. Probleme und Ausblick

Bereits das Vorhandensein unzähliger Lösungen und Herangehensweisen zeigt auf, dass bisher kein optimaler Weg gefunden wurde, Bilder und Grafiken für den jeweiligen Kontext optimiert zur Verfügung zu stellen, und dabei die unter *Abschnitt 4.4.1* beleuchteten Anforderungen gleichermaßen zu erfüllen.

So besteht bei vielen clientseitigen Lösungen mit MQs oder JS das Problem, dass Bilder doppelt geladen werden, was zu erheblicher Ladeverzögerung und somit zu beträchtlichen Performanceeinbußen führen kann.

119 Serverseitiges Bildbereitsellungstool Adaptive Images. http://adaptive-images.com/ (Stand 20.07.2013)
120 Apache-Webserver. http://httpd.apache.org/ (Stand 20.07.2013)
121 Serverseitige Grafikbibliothek GD-Lib. http://libgd.bitbucket.org/ (Stand 20.07.2013)
122 Serverseitiges Bildbereitstellungstool Sencha.io src http://www.sencha.com/products/io/ (Stand 20.07.2013)

Des Weiteren stellt sich die Frage, ob überhaupt hochauflösende Bilder ausgeliefert werden sollen, wenn man sich im mobilen Kontext befindet, aber temporär über keine ausreichend schnelle Verbindung verfügt. In diesem Falle wäre es wesentlich sinnvoller, die mindere Qualität von Grafiken in Kauf zu nehmen und die Website so performant wie möglich zu gestalten.

Diese Frage deckt jedoch das Problem der Unfähigkeit zur Bandbreitenabfrage auf. Momentan existieren keinerlei sinnvolle und zuverlässige Implementierungen zur Abfrage der momentanen Verbindungsgeschwindigkeit. Es gibt zwar Ansätze mit JS, beispielsweise „foresight.js“[123], die eine derartige Messung vornehmen, doch sind diese selbst wiederum der Performance nicht zuträglich: Es empfiehlt sich nicht, eine Dummy-Grafik zu Messzwecken herunterzuladen und auf Basis der Downloaddauer die Datenverbindung zu bestimmen. Außerdem kann die Netzabdeckung stark schwanken. So ist es möglich, dass der Download eines hochauflösenden Bildes mit einer HDSPA-Verbindung gestartet wurde, bei der Hälfte jedoch nur noch EDGE zur Verfügung steht. Eine derartige Implementierung müsste nativ durch den Browser geschehen, da dieser der Hardware am nächsten ist. Einen Ansatz, dessen Realisierung jedoch nicht absehbar ist, bietet die „Network Information API“[124] des W3Cs.

Ein weiteres Problem ist die uneinheitliche Art der Bereitstellung der Bilder, da bisher kein gemeinsamer Standard seitens des W3Cs verabschiedet wurde. So existieren oftmals für jedes Webprojekt verschiedene Lösungen, was ein permanentes Umdenken und Einarbeiten seitens der Entwickler erfordert.

Vielversprechende Ansätze bietet hierbei das <*picture*>-Element[125], mit welchem es möglich sein soll, verschiedene Bildquellen und korrespondierende MQ zu definieren, mit deren Hilfe dann der Browser die passende Ressource auswählt. Der Ansatz ist bereits zu diesem Zeitpunkt mit einem Polyfill[126] einzusetzen und stellt sich syntaktisch wie folgt dar:

```
<picture width="500"  height="500">
  <source media="(min-width: 45em)" src="large.jpg">
  <source media="(min-width: 18em)" src="med.jpg">
  <source src="small.jpg">
  <img src="small.jpg" alt="">
</picture>
```

Mit dieser Spezifikation könnte auch eine weitere Schwierigkeit, das sogenannte „Art Direction Problem“[127], bewältigt werden. Dieses betrifft den Bildausschnitt, der ebenfalls kontextabhän-

123 JS-Tool zur Geräteanalyse foresight.js. https://github.com/adamdbradley/foresight.js/ (Stand 21.07.2013)
124 W3C Working Draft. The Network Information Api. http://www.w3.org/TR/netinfo-api/ (Stand 21.07.2013)
125 W3C Working Draft. The picture element. http://www.w3.org/TR/html-picture-element/ (Stand 21.07.2013)
126 Polyfill für das picture-element. https://github.com/scottjehl/picturefill (Stand 21.07.2013)
127 Alexander, Sherri. Choosing A Responsive Image Solution. http://mobile.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/ (Stand 21.07.2013)

gig ausgewählt werden sollte, da sonst die Kernbotschaft eines Bilder verloren gehen könnte. Zu dessen Lösung werden jedoch verschiedene Bildvarianten benötigt. Nachfolgende Grafik illustriert die Problematik:

Abbildung 36 - Art-Direction Problem

Für Bilder, die als Hintergründe per CSS eingebunden werden, könnte das *image-set-property*[128] aus den CSS4-Spezifikationen von Interesse sein:

```
background-image: image-set(„img.png“ 1x, „img@2x.png“ 2x,
„img_print.png“ 600dpi);
```

Sollen alle Anforderungen erfüllt werden, angefangen bei kurzen Ladezeiten bis hin zur Lösungen der Bildausschnitt-Problematik, ist ein hoher Aufwand notwendig. Es wird noch einige Zeit und große Anstrengungen der Entwicklergemeinde benötigen, bis ein einheitlicher Weg gefunden wurde, um die zahlreichen Ansprüche an Bilder im MFRWD zu erfüllen.

Unabhängig davon, welche Lösung jetzt oder zu einem späteren Zeitpunkt gewählt wird, empfiehlt es sich Bilder zu komprimieren. Der Einsatz von Techniken wie der vorgestellten „Meeting the Middle“-Vorgehensweise oder von Optimierungstools wie „pngmin“[129] können dabei helfen, die Performance einer Website massiv zu steigern. Für UI-Elemente oder Icons ist der Einsatz der unter *Abschnitt 4.4.2.2* vorgestellten Vektorgrafiken lohnend.

4.5. Content

4.5.1. Anspruch im MFRWD

Wie bereits an manchen Stellen erwähnt, geht Mobile First mit Content First einher. Wie auch MFRWD nicht nur ein Ansatz für mobile sondern für alle Geräte ist, muss Gleiches auch für den Inhalt einer Website gelten. Inhalte müssen für alle Kontexte funktional sein[130]. Trotz des De-

128 W3C Editors Draft. Image-Set Notation. http://dev.w3.org/csswg/css-images/#image-set-notation (Stand 21.07.2013)
129 Tool zu Minimierung von PNGs pngmin. https://github.com/zauni/pngmin (Stand 21.07.2013)
130 McKGrane, Karen. Content Strategy for Mobile. New York: A book Apart 2012. S. 33

signs, der Benutzerfreundlichkeit und der technischen Raffinessen, die für eine gelungene UE notwendig sind, ist der Hauptgrund weshalb Websites aufgerufen werden, immer noch der Content. Diese Tatsache ist unumstößlich und wird sich auch nicht ändern. Mit der Zeit wurde dessen Wichtigkeit jedoch immer mehr in den Hintergrund gedrängt, wie nachfolgende Abbildung[131] aufzeigt:

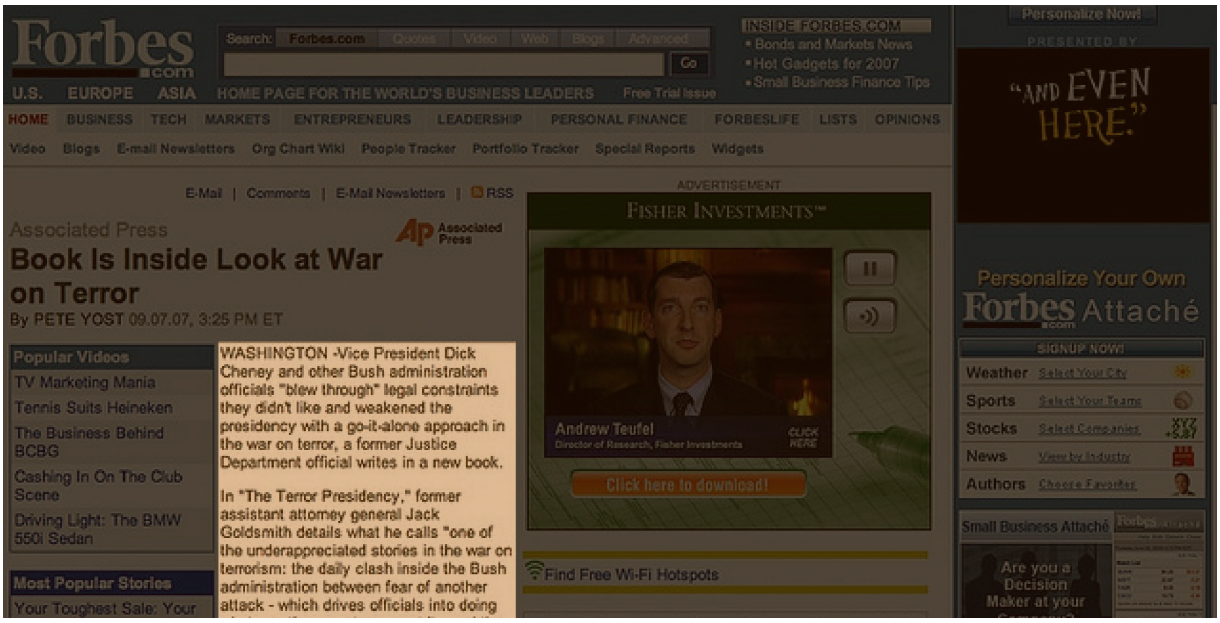


Abbildung 37 - Beispiel für Missachtung von Content First

Der Screenshot zeigt einen Artikel der Forbes-Website aus dem Jahre 2007. Dem eigentlichen Inhalt, der hervorgehobenen Nachrichtenmeldung, wird im Wirrwarr aus Werbebannern, Navigationen und anderen ablenkenden Elementen lediglich eine erschreckend kleine Spalte eingeräumt. Statt die Meldung mit einem korrespondierenden Bild aufzulockern, wird rechts daneben ein Werbevideo platziert, welches beinahe doppelt so viel Raum in Anspruch nimmt.

Natürlich handelt es sich hierbei um ein Extrembeispiel. Dennoch zeigt es die Notwendigkeit Inhalte zu priorisieren. Dabei beziehen sich die Termini Content und Inhalt nicht nur auf reinen Text, sondern auch auf Bilder, Diagramme, Tabellen – also alles, was Inhalte *transportiert*.

Wie bereits im Abschnitt *3. Gründe für MFRWD* aufgezeigt, ist es wichtig, dem Benutzer die Informationen zu bieten, die er zu einem bestimmen Zeitpunkt benötigt, also den Kontext zu berücksichtigen.

Dies stellt sich aber, anders als bei allen anderen bisher beleuchteten Teilbereichen des MFRWD, äußert kompliziert dar. Denn nur weil ein User, eine Website über das ME aufruft, bedeutet das noch nicht, dass er sich tatsächlich physikalisch mobil bewegt. Es ist durchaus möglich und sogar relativ wahrscheinlich, dass der Aufruf beispielsweise von der heimisches Couch aus erfolgt (vergleiche *Kapitel 3.1 Mobiles Wachstum*). Alle Anpassungen die Layout, Schrift etc. be-

131 Screenshot aus dem Forbes-Magazin von 2007. http://www.flickr.com/photos/merlin/1343376738/ (Stand 23.07.2013)

treffen, können ohne Weiteres getätigt werden, da diese sich auf das Gerät an sich beziehen. Eine Anpassung des Inhalts ist aber ungleich schwieriger, weil es nicht möglich ist festzustellen, in welchem Kontext sich der Nutzer „geistig" befindet. Wird die volle Informationsfülle einer Website erwartet oder reicht ein Bruchteil, beispielsweise eine Postanschrift, bereits aus?

Daher es ist unumgänglich, sich auf das Wesentliche einer Website zu konzentrieren. Es muss klar sein, welche Kernbotschaften transportiert werden sollen. Dazu müssen Texte angemessen kurz und prägnant formuliert sein. Davon profitieren mobiler und stationärer Kontext in gleichem Maße.

Content nach MFRWD muss sich daher mehr in Bezug auf Darstellung und Priorisierung anpassen lassen als inhaltlich. Die Kernaussage, der tatsächliche Inhalt, muss kontextübergreifend der Gleiche sein. Es muss also eine kontextabhängige Veränderung vorgenommen werden, jedoch ohne auf Kosten des Informationsgehalts zu gehen.

Obwohl sich die Ansprüche an den Inhalt einer Website relativ klar formulieren lassen, so ist es doch äußerst schwer ihnen zu genügen, da es nicht möglich ist, sie konkret zu „implementieren". Erschwerend kommt die Abhängigkeit vom jeweiligen Projekt und der Zielgruppe hinzu. Nachfolgend soll ein Einblick in die Kernstrategie für Content im MFRWD gegeben werden. Durch eine vollständige Betrachtung würde vom eigentlichen Thema abgekommen und der Rahmen dieser Arbeit gesprengt werden.

4.5.2. Content Strategy

Die Kernstrategie für Content in zukunftssicheren Websites lautet „Adaptive Content" (AC) und bezeichnet Inhalte, die sich in ihrer Darstellungsform kontextabhängig transformieren können, was wiederum mit den Grundsätzen des MFRWD korreliert. Es steht also die Forderung nach modularen Inhalten im Raum, die so gestaltet sind, dass sie auf multiplen Plattformen verwendbar sind. Die Texterin („Content Strategist") Karen McGrane beschreibt es wie folgt:

> *„It means getting your content into a format so you can share and distribute it to any platform you want."*[132]

Inhalte müssen also in eine Form gebracht werden, so dass sie in jeglichem Umfeld veröffentlicht werden können. Um dies zu erreichen, definiert McGrane die nachfolgend beleuchteten fünf Schlüsselelemente[133]:

- Wiederverwertbarer Content
- Strukturierter Content

132 McGrane, Karen. S. 47
133 McGrane, Karen. S. 53

- Präsentationsunabhängiger Content
- Beschreibende Metadaten
- Verwendung eines CMS

4.5.2.1. Wiederverwertbarer Content

Hierbei handelt es sich um Inhalte, die für eine optimale Wiederverwertung in unterschiedlichen Kontexten entwickelt wurden. So müssen beispielsweise Bilder im mobilen Umfeld andere Ausschnitte zeigen, als in der stationären Nutzungssituation (man denke an das „Art-Direction Problem" aus *Abschnitt 4.4.3*), da sonst die Kernbotschaft verloren gehen könnte. Gleiches gilt auch für textlichen Inhalt. Eine Headline kann auf dem Desktop optimal dargestellt werden, auf dem ME aber aufgrund ihrer Länge schlecht zu lesen sein. Daher empfiehlt es sich also, verschiedene Varianten zur Verfügung zu stellen:

„FC Bayern schlägt Barcelona in herausragender Art und Weise" | Desktop
„Furiose Bayern schlagen Barcelona" | Tablet
„Bayern 4, Barca 0" | Smartphone

„Deutsche Geheimdienste nutzen US-Schnüffelsoftware XKeyscore" | Desktop
„Geheimdienste nutzten US-Spähprogramm" | Tablet
„BND nutzt US-Software" | Smartphone

Die obigen Überschriften unterscheiden sich in ihrer Länge, jedoch nicht in ihrer Kernbotschaft und sind doch kontextspezifisch optimiert.

4.5.2.2. Strukturierter Content

Strukturierter Content bildet eine der Säulen des ACs und führt zu mehr Flexibilität hinsichtlich der Priorisierungs- und Kombinationsmöglichkeiten. Komplexe und lange Inhalte müssen in kürzere Blöcke geteilt und mit Hilfe des „Content-Modellings"[134] nach folgenden Kriterien kategorisiert werden:

- Inhaltstyp - Artikel, Bildergalerie etc.
- Attribute - Wird ein beschreibender Text oder eine Headline benötigt?
- Limitationen - Wo sind die Grenzen der Attribute? Werden spezielle Formate benötigt?
- Beziehungen - Stehen die Blöcke in Verbindung zueinander?

Dadurch, dass Inhalte strukturiert vorliegen, können sie je nach Kontext an einer anderen Stelle platziert und somit beispielsweise priorisiert werden.

134 McGrane, Karen. S. 60

4.5.2.3. Präsentationsunabhängiger Content

Content darf keine Formatierungen enthalten, weil dadurch die Fähigkeit der flexiblen und modularen Anzeige in den unterschiedlichen Umgebungen verloren geht. Um modularen Inhalt zu erzeugen, muss Form und Inhalt getrennt werden. Content muss also im Rohformat zur Verfügung gestellt werden, damit sein Aussehen vom jeweiligen Kontext bestimmt werden kann. Daher ist es insbesondere bei der Verwendung von WYSIWYG-Editoren in CMS-Systemen wichtig, eine „Verunreinigung" durch Formatierungen zu vermeiden.

4.5.2.4. Beschreibende Metadaten

Metadaten sollen den Inhalt genauer beschreiben. Zum einen können dies Schlüsselwörter, Priorisierungen oder Gültigkeitsdaten sein, zum anderen aber auch Tags, die das spätere Aussehen des Inhalts definieren können. Damit kann eine weitere Trennung von Aussehen und Inhalt erreicht werden. Aus der Kombination von Metadaten können schließlich Systeme entwickelt werden, die den Content kontextspezifisch formatieren. Man spricht von „Meaningful Metadata"[135].

4.5.2.5. Verwendung eines CMS

Um die bisher definierten Elemente in die Tat umzusetzen, wird unter anderem auch von McGrane die Verwendung eines CMS gefordert. Jedoch ist es derzeit schwierig ein System zu finden, welches den Anforderungen in Gänze gerecht werden kann, ohne dass es Anpassungen oder Kompromisse bedarf.

4.5.3. Fazit

Durch die obig beleuchteten Schlüsselelemente wird Content flexibler, besser skalierbar und einfacher zu warten[136]. Es wird möglich, Inhalte verschiedenartig zu präsentieren. So können diese beispielsweise versteckt werden, bis der User sie anfordert oder in unterschiedlicher Art und Weise angeordnet oder kombiniert sein, damit sie den Anforderungen des aktuellen Kontextes genügen, ohne das Informationen verloren gehen:

> *„Get your content ready to go anywhere because it's going to go everywhere."*[137]

Die Forderung von Brad Frost, Inhalte darauf vorzubereiten, überall angezeigt zu werden ist durchaus berechtigt. Jedoch wird erkennbar, dass AC ein äußerst komplexes Thema ist und den ganzen restlichen Entwicklungsprozess einer Website in den Schatten stellen kann. Natürlich ist es wichtig, die Devise „Content First" zu beachten, doch geht damit ein nicht unbeträchtlicher

135 McGrane, Karen. S. 73
136 Bradley, Steven. The Five Elements Of Modular And Adaptive Content. http://www.vanseodesign.com/web-design/adaptive-content/ (Stand 24.07.2013)
137 Frost, Brad. For a Future-Friendly Web. http://bradfrostweb.com/blog/web/for-a-future-friendly-web/ (Stand 24.07.2013)

technischer und konzeptioneller Zusatzaufwand einher. So müsste, soll allen obig dargelegten Schlüsselelementen Rechnung getragen werden, hinter jeder Website ein CMS stehen, was für viele Projekte schlicht und einfach überdimensioniert ist.

Die Realisierung aller Elemente des ACs bietet sich für extrem inhaltslastige Websites wie *tagesschau.de* oder *spiegel.de* an. Für kleinere bis mittlere Websites ist es im Normalfall ausreichend, Inhalte präzise und schnörkellos, auf die Zielgruppe bezogen aufzubereiten. Eine Modularisierung ermöglicht es beispielsweise, überlange Inhalte in Form eines Akkordeons zu verstecken, bis der Benutzer sie benötigt. Auch die Realisierung von kontextgerechten Headlines lässt sich beispielsweise mit MQs relativ einfach realisieren.

4.6. Workflow und Designprozess

4.6.1. Ausgangssituation und Problemstellung

Die gewachsenen Anforderungen des MFRWD zwingen Gestalter und Entwickler Projekte anders zu planen als bisher. Die Aufgaben sind wesentlich komplexer geworden, da es plötzlich nicht nur eine Bildschirmgröße und einen Kontext, sondern eine ganze Vielzahl davon zu beachten gilt. Wie verhält sich die Navigation, wenn nicht mehr ausreichend Raum für alle Links zur Verfügung steht? Was passiert mit dem Layout, wenn Breakpoints erreicht werden und wie bereitet man Bilder auf, so dass sie auf allen Geräten optimal dargestellt werden? Diese und viele weitere Fragen gilt es zu beantworten.

Der klassische Workflow, dem sogenannten Wasserfallmodell, welches den Anforderungen von MFRWD nicht gerecht wird, gestaltet sich häufig wie folgt:

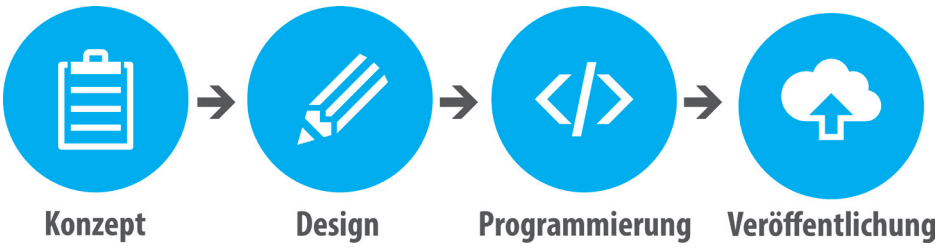


Abbildung 38 - Veraltetes Wasserfallmodell

Zunächst wird ein Konzept erarbeitet, anschließend in einem Grafikeditor (Photoshop o.ä.) ein Layout erstellt, welches nachfolgend in HTML/CSS umgesetzt und am Ende auf einem Webserver veröffentlicht wird.

Dieser Vorgehen ist für statische, nicht reaktionsfähige Websites deshalb möglich, weil keine erweiterten Anforderungen existieren: Wird ein Konzept verabschiedet und das Design vom Kunden abgenommen, kann es 1:1 umgesetzt werden. Eventuelle Fehlplanungen können in diesem Workflow erst relativ spät erkannt und nur mit großem Aufwand korrigiert werden.

Versucht man nun eine MFRWD-Website mit diesem Vorgehen zu gestalten, stößt man spätestens im Teilbereich Design auf die erste unüberwindbare Hürde: Wie gestaltet man für unzählig mögliche Auflösungen? Selbst wenn man sich im Design auf drei Breakpoints beschränkt, müssen bei beispielsweise fünf unterschiedlichen Seiten 3 x 5 = 15 Screendesigns erstellt werden, was einen schier nicht zu bewältigen und teuren Aufwand darstellt. Ergibt sich in einer dieser Seiten eine grundlegende Korrektur, so muss diese in allen anderen ebenso vorgenommen werden. Zu diesem massiven Problem im Wasserfallmodell gesellen sich noch weitere Nachteile[138]:

- Pixelperfekte Layouts wecken bei vielen Kunden falsche Erwartungen
- Flexible Layouts lassen sich nicht darstellen
- Strukturelle Änderungen haben oft Auswirkungen auf das gesamte Design
- Workflow ist langsam, teuer und unflexibel

Es bedarf daher einem Umdenken in der Art und Weise wie Websites geplant, gestaltet und technisch realisiert werden. Ein neuer Workflow wird benötigt.

4.6.2. Workflow und Designprozess im MFRWD

4.6.2.1. Grundlagen

Bereits in der Planung einer Website muss ein Umdenken stattfinden. Anstatt sich blindlings in die grafische Ausarbeitung zu stürzen, empfiehlt es sich, nach der Recherche und der Definition der Anforderungen an das Projekt, zunächst Inhalte herauszuarbeiten. Denn diese bilden wie bereits mehrfach erwähnt das Herzstück jeder Website (Content First), bei welcher es sich vorrangig um ein Informationsmedium handelt[139]. Erst im Anschluss daran, sollte man sich Gedanken über die Gestaltung machen, ganz nach der Devise *„Form Follows Function“*[140].

Bevor noch die erste Zeile Code bzw. die erste Ebene in einem Grafikeditor entsteht, lohnt es sich, Entwürfe mit Bleistift und Papier zu gestalten. Zum einen entstehen dadurch erste Layoutansätze wesentlich schneller, weil Gedanken rascher und unmittelbarer grafisch formuliert und zum anderen bereits zu dieser frühen Phase, Planungsfehler und Schwierigkeiten erkannt werden können. Zusätzlich dazu, können so auch verschiedene Bildschirmgrößen schneller simuliert werden, woraus eine massive Zeitersparnis resultiert.

Des Weiteren sollten Design und Programmierung stärker verzahnt, also parallel von Statten gehen. Im besten Falle wird bereits nach den ersten handgezeichneten Entwürfen ein interaktiver Prototyp erstellt um das Verhalten der Website in unterschiedlichen Kontexten analysieren

138 Hellwig, Jonas. Der Workflow im Responsive Webdesign. http://blog.kulturbanause.de/2013/06/workflow-responsive-web-design-prototyping/ (Stand 25.07.2013)
139 Ebenda.
140 Wikipedia-Artikel. Form Follows Function. http://de.wikipedia.org/wiki/Form_follows_function (Stand 25.07.2013)

zu können. So erhalten Kunden bereits zu diesem Zeitpunkt einen wesentlich realitätsnäheren Eindruck, wie die beauftrage Website letztendlich aussehen wird.

Grundsätzlich lässt sich sagen, dass es erstrebenswert ist, frühzeitig mit der technischen Realisierung zu beginnen und eine Website in Wechselwirkung zwischen Design und Programmierung zu entwickeln. Apples „President of Software Design", Sir Jonathan Ive formuliert es wie folgt:

> *„The process is much about designing, prototyping and making. If you seperate those, I think the final result will suffer."*[141]

Ein Workflow, der den Ansprüchen des MFRWD gerecht wird, enthält also iterative Elemente und stellt sich beispielsweise wie folgt dar:

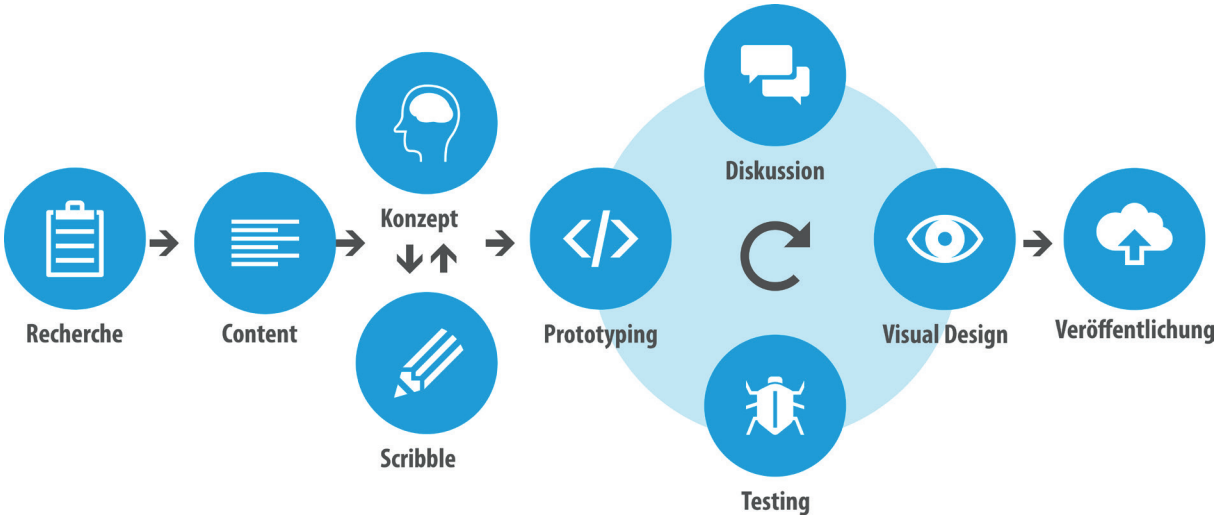


Abbildung 39 - Exemplarischer MFRWD-Workflow

Unverzichtbar ist außerdem das ausgiebige Testen einer Website. Dies sollte nicht nur mit Hilfe verschiedener Browser und simulierten Kontexten, sondern auch auf möglichst vielen realen Endgeräten geschehen.

Mit dieser neuen Herangehensweise steigen aber auch die Anforderungen an die Designer und Programmierer, ebenso an den Kunden. Eine kommunikative Zusammenarbeit zwischen diesen drei Polen ist unerlässlich.

Hinsichtlich der konkreten grafischen Gestaltung haben sich diverse grundsätzliche Vorgehensweisen entwickelt, von welchen die zwei Populärsten nachfolgend beleuchtet werden.

4.6.2.2. Reines Browserdesign

Einige Entwickler und Designer empfinden das Gestalten einer Website in einem Grafikeditor wie Photoshop zunehmend als Belastung und unnötigen Umweg[142]. Sie schlagen vor, die ge-

141 Sir Jonathan Ive. The iMan cometh. http://www.standard.co.uk/lifestyle/london-life/sir-jonathan-ive-the-iman-cometh-7562170.html (Stand 25.07.2013)

142 Responsive Webdesign in the Browser. Kill Photoshop. http://blog.teamtreehouse.com/responsive-web-design-in-the-browser-part-1-kill-photoshop (Stand 25.07.2013)

samte Design- und Entwicklungsphase mit Hilfe eines Code-Editors und des Browser zu absolvieren, um Änderungen unmittelbarer vornehmen zu können und somit Zeit zu sparen. Des Weiteren wird argumentiert, dass es einfacher ist, die Ansprüche der unterschiedlichen Kontexte frühzeitiger zu berücksichtigen.

Die in obiger Abbildung dargestellte Teilung zwischen „Visual Design" und „Prototyping" wird also zu einer einzigen Phase verschmolzen.

Obwohl die genannten Argumente zutreffend sind, gibt es dennoch einige kritische Faktoren in diesem Vorgehen. Zum einen wird vorausgesetzt, dass der Designer gleichzeitig als Entwickler fungiert, zum anderen hat ein Kunde, abgesehen von den handgezeichneten Entwürfen, keine Möglichkeit, Teile des Designs zu begutachten, bevor der Prototyp nicht einen gewissen Detailgrad erreicht hat.

Außerdem kann der permanente Wechsel zwischen Editor und Browser, um das eben Gestaltete und Programmierte zu begutachten, auf Dauer entnervend wirken, auch wenn beispielsweise mit den „Chrome DevTools"[143] die Möglichkeit besteht, Code direkt im Browser anzupassen. Das geänderte Markup muss letztendlich doch wieder in die richtige Datei übernommen werden.

4.6.2.3. Designing To The Extremes

Der Webdesigner Jeremy Girard schlägt vor[144], lediglich die „Extrempunkte" einer Website, also jeweils für den kleinsten und den größten möglichen Kontext, zu gestalten. Das bedeutet im Normalfall, dass zuerst eine Variante für das mobile und dann für das stationäre Nutzungsumfeld in einem grafischen Editor ausgearbeitet wird, nachdem ein ersten interaktiver Prototyp erstellt wurde.

Dieses Vorgehen hat den Vorteil, dass der Kunde bereits frühzeitig in den Gestaltungsprozess eingebunden werden kann. Außerdem müssen Gestalter und Entwickler nicht einer Person entsprechen.

4.6.2.4. Persönlicher Workflow des Autors

Die nachfolgend dargelegte Herangehensweise entspricht meinem persönlichen Vorgehen und hat sich in zahlreichen Projekten bewährt, was jedoch nicht bedeutet, dass dieser Workflow optimal und für Jedermann passend ist.

Mein Gestaltungsprozess beginnt nach der konzeptionellen Phase aufgrund der obig dargelegten Gründe ebenfalls auf Papier. Hierbei empfiehlt sich die Verwendung von vorgefertigten

143 Chrome DevTools. https://developers.google.com/chrome-developer-tools/?hl=de (Stand 25.07.2013)
144 Girard, Jeremy. Building A Better Responsive Website. http://mobile.smashingmagazine.com/2013/03/05/building-a-better-responsive-website/ (Stand 25.07.2013)

„Sketch-Sheets“[145], die beispielsweise bereits über ein Browserfenster oder ein Grid verfügen.

Da ich ein Anhänger des Ansatzes „Designing To The Extremes“ bin, erarbeite ich jeweils ein Scribble für den mobilen und den stationären Kontext. Zumeist skizziere ich dabei das Layout einer Startseite und einer typischen Inhaltsseite. Nur wenn sich die Einzelseiten einer Website stark voneinander unterscheiden und individuellen Charakter besitzen, beziehe ich sie in die Zeichenphase mit ein.

Ist ein Scribble fertiggestellt wird ein detaillierterer Wireframe – also ein grafischer Prototyp – mit Hilfe eines Editors, in meinem Falle Photoshop, erstellt. Wird in Teams gearbeitet, ist es den Entwicklern bereits jetzt möglich, mit Hilfe des Wireframes einen interaktiven Prototypen anzufertigen, während der Gestalter den Entwurf weiter ausarbeiten kann. Die Verwendung eines Wireframes lohnt sich deshalb, weil in diesem die Proportionen der einzelnen Gestaltungselemente näher am tatsächlichen Endprodukt liegen, als in der gezeichneten Version, da beispielsweise bereits hier mit dem später verwendeten Grid gearbeitet wird.

Die Ausarbeitung des Entwurfs beginnt zunächst mit der Erstellung eines „Styletiles“[146]. Dabei handelt es sich um ein einzelnes Photoshop-Dokument, welche alle benötigen Farben, Schriften, Logos usw. enthält. Durch dieses kann in Verbindung mit den Wireframes ein erster visueller Eindruck gewonnen werden. Der Kunde kann bereits zu diesem frühen Zeitpunkt eine relativ gute Vorstellung entwickeln, wie sich die endgültige Website darstellen wird. Außerdem können Fehler und Korrekturwünsche rasch erkannt und bearbeitet werden.

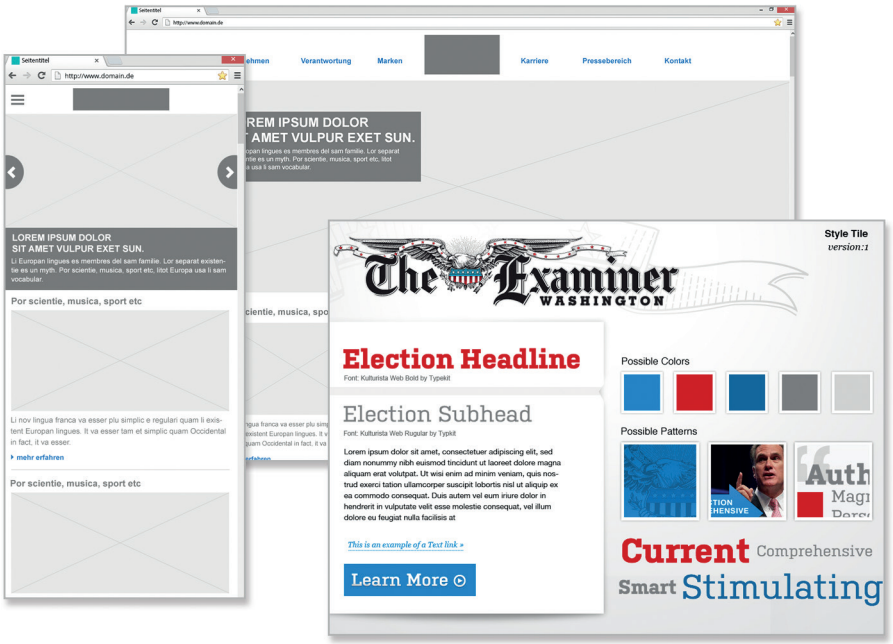


Abbildung 40 - Beispiele für Wireframe und Styletile

Wurde allen Ansprüchen und Änderungswünsche in dieser ersten Abstimmungsrunde Rechnung getragen, werden Wireframes in konkrete und detailreiche Photoshop-Layouts überführt.

145 Browser Sketch-Sheets. http://sneakpeekit.com/ (Stand 25.07.2013)
146 Hilfsmittel zur Gestaltung Styletile. http://styletil.es/ (Stand 25.07.2013)

Währenddessen kann die Codeentwicklung weiter fortschreiten und der Kunde erhält die ausgearbeiteten Entwürfe zur Ansicht.

Alle weiteren Seiten werden auf Basis der entwickelten Styletiles, Wireframes und Feinlayouts entwickelt und adaptiert. Der Zwischenschritt der Ausgestaltung in Photoshop ist an sich nicht zwingend notwendig, hilft aber dabei, grafische Finessen auszuarbeiten. Außerdem ist die Vorstellungskraft vieler Kunden begrenzt, daher wird dieser Schritt oftmals benötigt.

Bei großen und umfangreichen Projekten ist es hilfreich, Designsysteme zu entwickeln[147]. Dabei handelt es sich im weitesten Sinne um erweiterte Styletiles, die alle wichtigen UI-Elemente wie Buttons, Formularfelder oder Icons enthalten.

Der weitere Workflow gestaltet sich ähnlich, wie in obiger Abbildung dargestellt.

4.6.3. Fazit

Es wurde ersichtlich, dass ein veränderter Workflow benötigt wird, um den Anforderungen des MFRWD gerecht zu werden. Dieser erscheint auf den ersten Blick um ein Vielfaches aufwendiger als das bisher oftmals eingesetzte Wasserfallmodell. In der Tat erfordert der neue Prozess ein Mehr an Kommunikation und Koordination, gerade wegen dem iterativen Anteil. In der Summe werden jedoch durch die intensive Planungsphase und die Verzahnung von Design und Entwicklung unnötige und frustrierende Korrekturschleifen vermieden, was in einem besseren Endprodukt resultiert.

4.7. Tools & Frameworks

4.7.1. Tools

Im Lauf der Zeit haben sich einige äußerst sinnvolle Tools für die Entwicklung von Websites entwickelt, die gerade bei der Umsetzung von MFRWD äußerst hilfreich sein können. Nachfolgend vorgestellte Tools stellen lediglich eine kleine Auswahl aus den relevanten Teilbereichen dar.

4.7.1.1. Designtools

Wie bereits im *Abschnitt 4.6 Workflow* dargelegt, empfinden viele Designer die Verwendung von Photoshop oder ähnlichen Grafikeditoren aufgrund ihrer Unfähigkeit flexible Layouts darzustellen, als überholt. Auf diesen Umstand hat auch der Softwarehersteller Adobe reagiert und mit „Edge Reflow“[148] ein Programm entwickelt, mit dem responsive Layouts erstellt werden können. Es ist Teil von „Adobe Edge“[149], einer Sammlung aus Programmen und Services, auf welche mit

147 Zillgens, Christoph. S. 165
148 Adobe Edge Reflow. http://html.adobe.com/edge/reflow/ (Stand 26.07.2013)
149 Adobe Edge. http://html.adobe.com/edge/ (Stand 26.07.2013)

einer kostenlosen „Creative-Cloud“[150]-Mitgliedschaft zugegriffen werden kann.

Edge Reflow stellt eine Art Hybrid aus Fireworks, Photoshop und Dreamweaver dar. In der Oberfläche des Programms können zahlreiche Einstellungen wie Breakpoints oder MQs vorgenommen, sowie Text und andere Inhalte platziert werden. Das Layout wird also mit Hilfe des GUIs erstellt, im Hintergrund generiert „Edge Reflow“ jedoch HTML/CSS-Code.

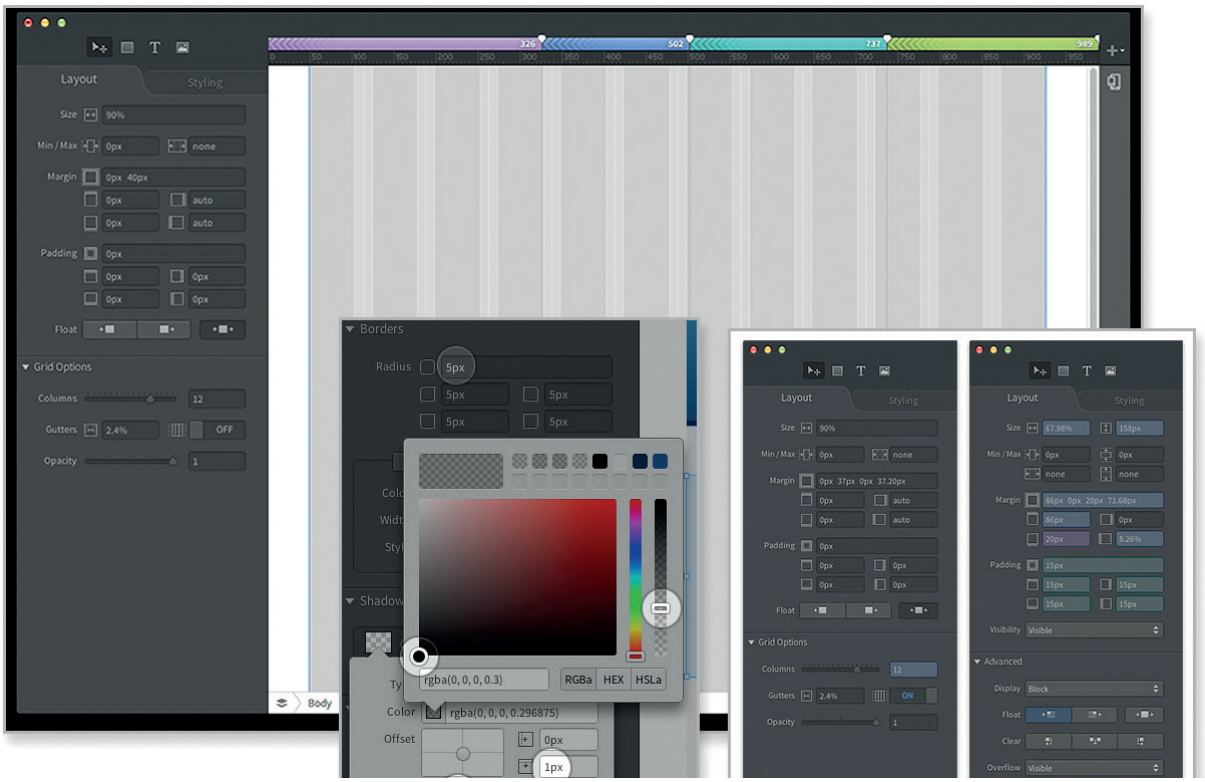


Abbildung 41 - Screenshot von Adobes Edge Reflow

Laut Adobe[151] soll „Edge Reflow“ Programme wie Photoshop nicht ersetzten, sondern lediglich ein Tool zur Erstellung von reaktionsfähigen Prototypen sein. In jedem Falle stellt es aber einen interessanten Ansatz dar.

Als ähnliches Tool ist das browserbasierte „Webflow“[152] zu nennen, welches sich jedoch noch in der Beta-Phase befindet.

4.7.1.2. Vorschautools

Um während der Entwicklung möglichst viele Nutzungssituationen und Kontexte simulieren zu können, empfiehlt sich die Verwendung von Hilfsprogrammen zur Vorschau von Websites nach MFRWD.

Hierbei lohnt sich ein Blick auf das von Brad Frost entwickelte „Am I Responsive“[153], einer einfa-

150 Adobe Cloud. https://creative.adobe.com/ (Stand 26.07.2013)
151 Griffith, Chris. Introducing Edge Reflow Preview. http://www.adobe.com/devnet/edge-reflow/articles/introducing-edge-reflow.html (Stand 26.07.2013)
152 Browsertool zur Erstellung von responsiven Layouts Webflow. http://www.webflow.com/ (Stand 26.07.2013)
153 Tool zur Vorschau von Websites. http://ami.responsivedesign.is/ (Stand 26.07.2013)

chen browserbasierten Lösung. Über ein Textfeld kann die URL der zu betrachtenden Website eingetragen werden (auch http://localhost/ ist möglich), welche dann zur Vorschau in unterschiedliche Geräte geladen wird.

Auch über die bereits erwähnten „Chrome DevTools“ kann eine Simulation der Website für unterschiedliche Kontexte erfolgen:

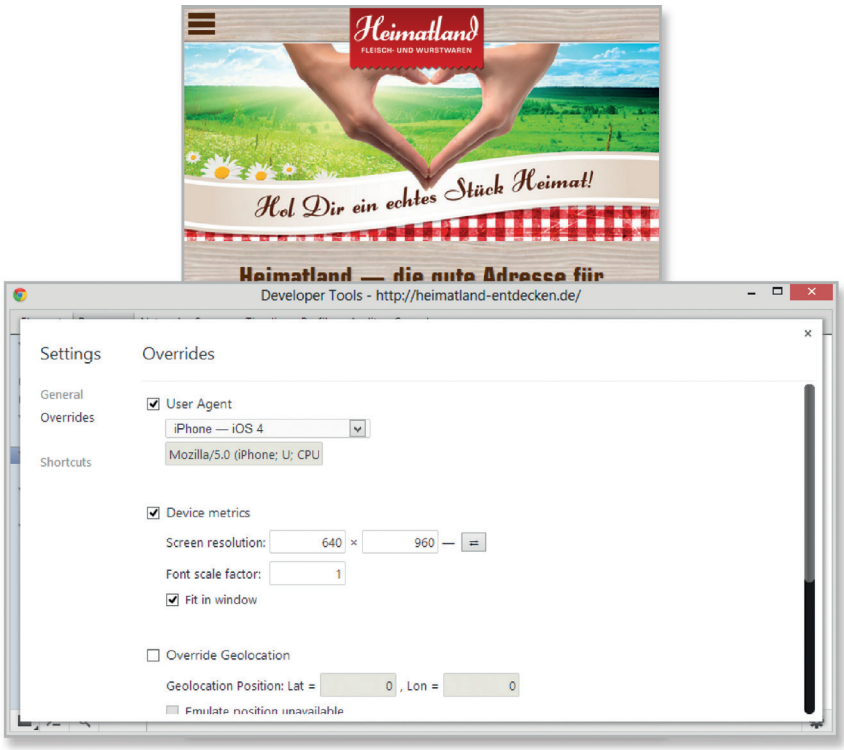


Abbildung 42 - Kontextspezifische Vorschau mittels ChromeDev Tools

Vorgestellte Tools können das Testen mit realen Endgeräten natürlich nicht ersetzen, bieten aber gerade in der Entwicklungsphase wertvolle Informationen.

4.7.1.3. Performancetools

Performance ist gerade auf ME eine äußerst knappe Ressource, daher ist es vor der Veröffentlichung einer Website unverzichtbar, diese auf etwaige Leistungsschwächen zu untersuchen. Dazu empfiehlt sich die Verwendung von „YSlow“[154], einem von Yahoo entwickelten Performancetool. Dieses analysiert eine Website entweder anhand von vor- oder frei definierbaren Parametern. Es überprüft beispielsweise ob Ressourcen minimiert vorliegen oder die Anzahl von http-Request in einem vertretbaren Rahmen stattfinden. Es ist entweder als Browserplugin oder als Kommandozeilentool erhältlich.

Mit Hilfe der „Chrome DevTools“ kann die Performance einer Website ebenfalls überprüft werden, indem beispielsweise eine Analyse des Speicherverbrauchs mittels der integrierten „Timeline“[155] erfolgt.

154 Performancetool YSlow. http://developer.yahoo.com/yslow/ (Stand 26.07.2013)
155 Chrome DevTools. Performance profiling with the timeline. https://developers.google.com/chrome-developer-tools/docs/timeline?hl=de (Stand 26.07.2013)

4.7.2. Frameworks

Um schneller und effizienter Entwickeln zu können, bietet sich der Einsatz eines Frameworks an. Dabei handelt es sich im Grunde um Sammlungen von vordefinierten Modulen, Grids und Best-Practices, die abhängig vom jeweiligen Projekt kombiniert und angepasst werden können.

Ob man auf die Hilfe eines Frameworks zurückgreifen soll, lässt sich nicht pauschal beantworten. Einerseits ist es durch ihre Verwendung möglich, schneller zu entwickeln, da beispielsweise keine eigenen Grids definiert oder UI-Elemente lediglich angepasst werden müssen. Außerdem sind die inkludierten Lösungen von einer Vielzahl von Entwicklern verwendet und erprobt worden, was für eine hohe Qualität der Implementierungen spricht. Anderseits limitiert man sich unter Umständen in gestalterischer sowie technischer Hinsicht und muss in Kauf nehmen, dass die vordefinierten Rahmenbedingungen einzuhalten sind. Des Weiteren besteht die Gefahr, dass viele Funktionalitäten eines Frameworks nicht benutzt werden, aber dennoch im Code der Website inkludiert sind, was diesen unnötig aufblähen kann.

Es existieren unzählige Frameworks, welche die Realisierung von Websites nach MFRWD erleichtern sollen. Zu den beiden Größten und Populärsten zählen „Bootstrap“[156], welches von Twitter entwickelt wurde und „Foundation“[157], das aus dem Hause Zurb stammt. Nachfolgende Tabelle gibt einen Überblick über beide Frameworks:

	Bootstrap	**Foundation**
Version	2.3.2	4
Grid	Multiple Breakpoint, responsive	Ein einziger vordefinierter Breakpoint, responsive
CSS-Komponenten	Buttons, Modals, Media Object, Popovers. Ingesamt äußerst modularer Aufbau	Buttons, Modals, Media Object, Popovers. Ingesamt äußerst modularer Aufbau
JavaScript	jQuery	Zepto
CSS Preprocessor	LESS	SASS
BrowserSupport	>= Internet Explorer 8	> Internet Explorer 8
MobileFirst	Durch minimale Anpassung	Standardmäßig

Tabelle 3 - Vergleich Bootstrap / Foundation

Zusammenfassend lässt sich sagen, dass die Unterschiede im Detail liegen und die Auswahl des richtigen Frameworks auch auf Basis der persönlichen Vorlieben getroffen werden muss. Für „Foundation“ spricht, dass es standardmäßig mit den Prinzipien von MF entwickelt wurde. Es benötigt jedoch nur wenige Zeilen Code um auch „Bootstrap“ das gleiche Verhalten an den Tag legen zu lassen. Die anstehende Veröffentlichung der Version 3.0 von Bootstrap verspricht die vollständige Fokussierung auf den Ansatz MF.

156 Framework Twitter Bootstrap. http://twitter.github.io/bootstrap/index.html (Stand 26.07.2013)
157 Framework Foundation. http://foundation.zurb.com/ (Stand 26.07.2013)

5. Fazit

Das Web ist nicht mehr auf dem Weg mobil zu werden, sondern ist bereits dort angekommen. Die vorangegangenen Kapitel zeigen auf, dass es sich bei MFRWD um einen lohnenden und unumgänglichen Ansatz zur Gestaltung von zukunftssicheren Websites handelt, will man für das was kommen mag gerüstet sein. Wer weiß, welche Geräte, Formate oder Plattformen die Zukunft bereithält?

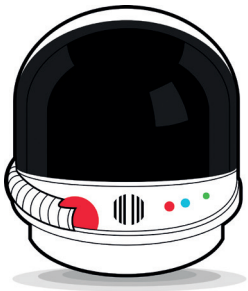
Nach der Lektüre dieser Arbeit sollte ein grundsätzliches Verständnis der hier vorgestellten Techniken vorhanden und eine Bereitschaft geschaffen worden sein, bisherige Projekte kritisch zu betrachten und zukünftig auf den hier vorgestellten Ansatz zu setzen.

Die meisten der analysierten Teilbereiche verfügen bereits jetzt über sinnvolle und erprobte Implementierungen. Natürlich existiert noch massiver Handlungsbedarf in Sachen Standardisierung. Doch ist es mit den vorgestellten Lösungen bereits jetzt möglich, Mobile First Responsive Webdesign in seiner Vollständigkeit zu betreiben.

Die Entwicklergemeinde wird weiterhin gemeinsam an Lösungen und neuen zielführenden Implementierung arbeiten. Es gibt keinen goldenen Weg – keine Methoden, Technologien oder Workflows – die für alle Projekte gültig wären. Es gilt immer, die jeweils passenden Schritte zu wählen um dem Benutzer eine optimale Web-Experience zu ermöglichen.

Die Kritik[158], Design nach MFRWD führe zu langweiligen Websites, da durch den Einsatz von Grids und wiederverwertbaren Modulen eine Art Einheitsmasse entstehe, sei an dieser Stelle nachdrücklichst zurückgewiesen. Das Gegenteil beweisen die unzähligen fantastischen Beispiele, die das Web zu bieten hat, in technischer sowie gestalterischer Hinsicht.

Es liegt an den Gestaltern und Entwicklern die Philosophie, die in Mobile First Responsive Webdesign verankert ist, weiterzutragen, in ihre tägliche Arbeit einfließen zu lassen und Websites zu erstellen, die „future friendly“[159] sind.



future friendly

158 Longnecker, Jonathan. Responsive Webdesign is boring. http://www.netmagazine.com/opinions/responsive-web-design-boring (Stand 27.07.2013)

159 Initiative für zukunftssichere Websites. http://futurefriend.ly/index.html (Stand 27.07.2013)

Anhänge

Auf beiliegendem Datenträger befinden sich viele der hier vorgestellten Codebeispiele und Strategien zur Umsetzung von „Mobile First Responsive Webdesign“. Zudem kann diese Thesis darauf als PDF gefunden werden.

Literaturverzeichnis

10 Jahre UMTS-Versteigerung. http://www.bitkom.org/de/themen/54894_64598.aspx
(Stand 01.06.2013)

Motorola Z3 WAP & JAVA Internet browsing. http://www.youtube.com/watch?v=8aaOtVJQcg0
(Stand 01.06.2013)

Spezifikationen des Apple iPhones Original. http://de.wikipedia.org/wiki/Apple_iPhone_(original)
(Stand 01.06.2013)

Maurice, Florence. Mobile Websiten. Strategien, Techniken, Dos und Don'ts für Webentwickler. München: Carl Hanser Verlag 2012.

Remick, Jarel. What is a Web App? Here is our definition. http://web.appstorm.net/general/opinion/what-is-a-web-app-heres-our-definition/ (Stand 02.06.2013)

Marcotte, Ethan. Responsive Webdesign. http://alistapart.com/article/responsive-web-design
(Stand 02.06.2013)

Zilligens, Christoph. Responsive Webdesign. München: Carl Hanser Verlag 2013.

Hellwig, Jonas. Adaptive Website vs. Responsive Website. http://blog.kulturbanause.de/2012/11/ad apti-ve-website-vs-responsive-website/ (Stand 02.06.2013)

Rieger, Bryan. Website zur Demonstration von Mobile First Responsive Webdesign. http://yiibu.com/about/site/index.html (Stand 03.06.2013)

Gustafson, Aaron. Understanding Progressive Enhancement. http://alistapart.com/article/understandingpro-gressiveenhancement (Stand 03.06.2013)

Borowska, Paula. Graceful Degration vs. Progressive Enhancement. http://designsuperstars.net/graceful-de-gradation-versus-progressive-enhancement-in-web-design-who-will-win/ (Stand 03.06.2013)

Wroblewski, Luke. Google CEO Eric Schmidt on Mobile First. http://www.lukew.com/ff/entry.asp?1270
(Stand 05.06.2013)

Our Mobile Planet. Smartphone Studie von Google. Endbericht 2012. http://services.google.com/fh/files/blogs/our_mobile_planet_germany_de.pdf

Mobile phones will overtake PCs as the most common Web access device worldwide [Gartner]. http://www.nextbigwhat.com/mobile-phones-will-overtake-pcs-as-the-most-common-web-access-device-worldwide-gartner-297/ (Stand 05.06.2013)

Wroblewsky, Luke. Designing for today's web. http://static.lukew.com/TodaysWeb_09302010.pdf
(Stand 05.06.2013)

New Study Reveals the Mobile Web Disappoints Global Consumers. http://www.compuware.com/d/relea-se/592528/new-study-reveals-the-mobile-web-disappoints-global-consumers (Stand 26.02.2013)

Perez Sarah. Facebook's Mobile Monthly Active Users Grew 21% Over Past Four Months. http://techcrunch.com/2012/02/01/facebook-has-425-million-mobile-monthly-active-users-up-from-350-million-in-september/
(Stand 05.06.2013)

Costine, Josh. Facebook VP: We Pivoted To Create The Right Mobile Experience First, The Desktop Can Catch Up Later. http://techcrunch.com/2012/10/19/facebook-mobile-first/ (Stand 05.06.2013)

Preisentwicklung bei verschiedenen Smartphone-Modellen in Deutschland von 2010 bis März 2011 (in Euro). http://de.statista.com/statistik/daten/studie/182567/umfrage/preisentwicklung-bei-smartpho-nes-in-deutschland/ (Stand 05.06.2013)

Wroblewski, Luke. Mobile First. New York: A book Apart 2011

Datenübertragung im Mobilfunk. http://www.elektronik-kompendium.de/sites/kom/0910141.htm (Stand. 12.06.2013)

Is mobile affecting when we read? http://readitlaterlist.com/blog/2011/01/is-mobile-affecting-when-we-read/ (Stand 12.06.2013)

Wellman, Stephen. Google lays out it's Mobile User Experience Strategy. http://www.informationweek.com/mobility/business/google-lays-out-its-mobile-user-experien/229216268 (Stand 12.06.2013)

W3C Working Draft. Geolocation API Specification. http://dev.w3.org/geo/api/spec-source.html (Stand 16.06.2013)

Wroblewski, Luke. An Event Apart: Content First. http://www.lukew.com/ff/entry.asp?1598 (Stand 05.07.2013)

Frost, Brad. Mobile First Responsive Webdesign. http://bradfrostweb.com/blog/web/mobile-first-responsive-web-design/ (Stand 06.07.2013)

W3C Recommendation. Media Queries. http://www.w3.org/TR/css3-mediaqueries/ (Stand 05.07.2013)

Browserunterstützung von Media Queries. http://caniuse.com/#feat=css-mediaqueries (Stand 05.07.2013)

Grigsby, Jason. CSS Media Query for Mobile is Fool's Gold. http://blog.cloudfour.com/css-media-query-for-mobile-is-fools-gold/

Yahoo! Developer Network. Best Practises for Speeding Up Your Websites. http://developer.yahoo.com/performance/rules.html (Stand 05.07.2013)

Fastest Processor. Cell Phone Top List. http://www.phonegg.com/Top/Fastest-Processor-Cell-Phones.html (Stand 06.07.2013)

Matanich, Tyson. Media Queries are not the answer. http://coding.smashingmagazine.com/2013/06/25/media-queries-are-not-the-answer-element-query-polyfill/ (Stand 05.07.2013)

Matanich, Tyson. Element Query Polyfill. https://github.com/tysonmatanich/elementQuery (Stand 05.07.2013)

Marcotte, Ethan. Responsive Webdesign. New York: A book Apart 2011

Wroblewsky, Luke. Multi Device Layout Pattern. http://www.lukew.com/ff/entry.asp?1514 (Stand 06.07.2013)

W3C Candidate Recomendation. CSS Values and Units Module Level 3. http://www.w3.org/TR/css3-values/#-font-relative-lengths (Stand 07.07.2013)

W3C Candiate Recommendation. CSS Flexible Box Layout Module. http://www.w3.org/TR/css3-flexbox/ (Stand 09.07.2013)

Browserunterstützung von Flexbox. http://caniuse.com/#feat=flexbox (Stand 10.07.2013)

W3C Working Draft. CSS Template Layout Module. http://www.w3.org/TR/2009/WD-css3-layout-20090402/ (Stand 10.07.2013)

W3C Candidate Recommendation. CSS Multi-column Layout Module. http://www.w3.org/TR/css3-multicol/ (Stand 10.07.2013)

W3C Working Draft. CSS Grid Layout. http://www.w3.org/TR/css3-grid-layout/ (Stand 10.07.2013)

iOS Human Interface Guidlines. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html (Stand 14.07.2013)

Windows Phone UI Design and Interaction Guide. http://go.microsoft.com/?linkid=9713252 (Stand 14.07.2013)

Finger-Friendly-Design. http://uxdesign.smashingmagazine.com/2012/02/21/finger-friendly-design-ideal-mobile-touchscreen-target-sizes/ (Stand 14.07.2013)

Hedmann, Falk. Webdesign Trends. 10 Dinge, die uns 2013 erwarten. http://t3n.de/news/webdesign-trends-2013-10-dinge-424796/ (Stand 17.07.2013)

Qualitätsstudie der Bundesnetzagentur zur Datenübertragung. http://www.bundesnetzagentur.de/cln_1912/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Breitband/Dienstequalitaet/qualitaetsstudie/qualitaetsstudie-node.html;jsessionid=A26A459969C2C61B7A1604F39A1B3801 (Stand 17.07.2013)

Internetstatistik zu Dateigrößen. http://httparchive.org/interesting.php (Stand 17.07.2013)

Work, Sean. How Loading Time Affects Your Bottom Line. http://blog.kissmetrics.com/loading-time/ (Stand 17.07.2013)

Rupert, Dave. Ughck. Images. http://daverupert.com/2013/06/ughck-images/ (Stand 17.07.2013)

Browserunterstützung von background-size. http://caniuse.com/#feat=background-img-opts (Stand 17.07.2013)

CSS Image Sprite. http://www.w3schools.com/css/css_image_sprites.asp (Stand 18.07.2013)

Jobsis, Daan. Retina Revolution. http://blog.netvlies.nl/design-interactie/retina-revolution/ (Stand 18.07.2013)

Matanich, Tyson. Images in a responsive Web. http://www.matanich.com/2012/11/06/picture-polyfill/ (Stand 18.07.2013)

Robson, Ann. Progressive jpegs: a new best practise. http://calendar.perfplanet.com/2012/progressive-jpegs-a-new-best-practice/ (Stand 18.07.2013)

W3C Recommendation. Scalable Vector Graphics (SVG) 1.1 Specification. http://www.w3.org/TR/2003/ REC-SVG11-20030114/ (Stand 19.07.2013)

Browserunterstützung von SVG. http://caniuse.com/#feat=svg (Stand 19.07.2013)

CSS3-Info. Web-Fonts with @font-face. http://www.css3.info/preview/web-fonts-with-font-face/ (Stand 19.07.2013)

W3C Working Draft. CSS Fonts Module Level 3. http://www.w3.org/TR/css3-fonts/ (Stand 19.07.2013)

Devloper Guidlines von Apple. Supporting High-Resolution Screens. http://developer.apple.com/library/ios/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/SupportingHiResScreensInViews/SupportingHiResScreensInViews.html#//apple_ref/doc/uid/TP40010156-CH15-SW1 (Stand 20.07.2013)

W3C Working Draft. The Network Information Api. http://www.w3.org/TR/netinfo-api/ (Stand 21.07.2013)

W3C Working Draft. The picture element. http://www.w3.org/TR/html-picture-element/ (Stand 21.07.2013)

Alexander, Sherri. Choosing A Responsive Image Solution. http://mobile.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/ (Stand 21.07.2013)

W3C Editors Draft. Image-Set Notation. http://dev.w3.org/csswg/css-images/#image-set-notation (Stand 21.07.2013)

McKGrane, Karen. Content Strategy for Mobile. New York: A book Apart 2012.

Bradley, Steven. The Five Elements Of Modular And Adaptive Content. http://www.vanseodesign.com/web-design/adaptive-content/ (Stand 24.07.2013)

Frost, Brad. For a Future-Friendly Web. http://bradfrostweb.com/blog/web/for-a-future-friendly-web/ (Stand 24.07.2013)

Hellwig, Jonas. Der Workflow im Responsive Webdesign. http://blog.kulturbanause.de/2013/06/workflow-responsive-web-design-prototyping/ (Stand 25.07.2013)

Wikipedia-Artikel. Form Follows Function. http://de.wikipedia.org/wiki/Form_follows_function (Stand 25.07.2013)

Sir Jonathan Ive. The iMan cometh. http://www.standard.co.uk/lifestyle/london-life/sir-jonathan-ive-the-iman-cometh-7562170.html (Stand 25.07.2013)

Responsive Webdesign in the Browser. Kill Photoshop. http://blog.teamtreehouse.com/responsive-web-design-in-the-browser-part-1-kill-photoshop (Stand 25.07.2013)

Girard, Jeremy. Building A Better Responsive Website. http://mobile.smashingmagazine.com/2013/03/05/building-a-better-responsive-website/ (Stand 25.07.2013)

Griffith, Chris. Introducing Edge Reflow Preview. http://www.adobe.com/devnet/edge-reflow/articles/introducing-edge-reflow.html (Stand 26.07.2013)

Chrome DevTools. Performance profiling with the timeline. https://developers.google.com/chrome-developer-tools/docs/timeline?hl=de (Stand 26.07.2013)

Longnecker, Jonathan. Responsive Webdesign is boring. http://www.netmagazine.com/opinions/responsive-web-design-boring (Stand 27.07.2013)