Fachbereich Informationswissenschaften

Fachhochschule Potsdam

Responsive Webdesign

Eine Untersuchung zu Ansätzen, Methoden und technischen Möglichkeiten in der Umsetzung von anpassungsfähigen Websites

Bachelorarbeit

Patrick Posner

Matrikel-Nummer 11163

Betreuer	BA. Anja Baltrusch
Erstprüfer	Prof. Dr. Angela Schreyer
Zweitprüfer	BA. Anja Baltrusch

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.0.1 Überblick

«Today, anything that's fixed and unresponsive isn't web design, it's something else. If you don't embrace the inherent fluidity of the web, you're not a web designer, you're something else. »(Cla11)

Das Zitat von Andrew Clarke zeigt die Bedeutung von Responsive Webdesign im Kontext der Entwicklung von modernen Websites. Die Möglichkeit der Reaktion von Websites auf unterschiedliche Zustände wie z.B. Bildschirmgrößen oder Displayauflösungen ist zum elementaren Bestandteil des Entstehungsprozesses einer Website geworden.

Ziel dieser Arbeit soll eine fundierte Untersuchung der Konzepte, Methoden und Lösungsansätze im Responsive Webdesign sein, um eine qualitative Einschätzung zur flächendeckenden Verwendung der neuen Möglichkeiten im Bereich des Webdesigns geben zu können. Dabei fließen neben den Meinungen unterschiedlicher Fachspezialisten und Meinungsträgern auch die praktischen Erfahrungen des Autors mit ein.

Responsive Webdesign wird in dieser Arbeit ganzheitlich betrachtet. Aus diesem Grund wird im ersten Kapitel mit der Begriffserklärung und der Arbeitsorganisation (nachfolgend Workflow genannt) zur Entwicklung von reaktionsfähigen Websites begonnen. Die drei zentralen Workflow-Modelle Graceful Degradation, Progressive Enhancement und Mobil First werden thematisiert.

Im zweiten Kapitel werden alle grundlegenden Elemente einer Website vorgestellt und auf Möglichkeiten zur Umsetzung im Responsive Webdesign untersucht. Jedes Unterkapitel liefert dabei unterschiedliche Ansätze zur Umsetzung, Auswertung der Vor- und Nachteile und eine abschließende Zusammenfassung des Themenbereichs. Es werden alle elementaren Bestandteile zur Umsetzung von reaktionsfähigen Websites untersucht, so z. B. der Einsatz von Rastern, die Implementation und Funktionsweise von Mediaqueries, der Umgang mit Bildern, Möglichkeiten von Navigationstypen und der Umgang mit Typographie im Responsive Webdesign.

1.0.2 Methodik

Die Methodik dieser Arbeit bildet sich aus einer Analyse der gängigen Fachlektüre und der aktiven Auseinandersetzung mit dieser durch den Autor. Alle Konzepte, Methoden und Lösungsansätze werden durch den Autor kontextabhängig geprüft und im Sinne der optimalen Anwendung bewertet. Grundlage der Analyse stellt zum einen die gängige englische und deutsche Fachlektüre zum Thema, zum anderen wird sie ergänzt durch diverse Online- und Multimediaquellen, um eine ganzheitliche Betrachtung der Themenkomplexe zu ermöglichen.

1.0.3 Abgrenzung der Arbeit und Definition der Zielgruppe

Diese Arbeit besteht auf eine ganzheitliche Betrachtung des Responsive Webdesigns. Es wird versucht dem Leser eine qualitative Auswahl an Möglichkeiten zur Umsetzung einer reaktionsfähigen Website aufzuzeigen. Explizit ausgeschlossen sind in dieser Bachelorarbeit angrenzende Themenbereiche wie z. B. die Entwicklung von nativen Applikationen und der Einsatz von gesonderten mobilen Websites. Die Zielgruppe sollte über ein Grundverständnis in den Beschreibungssprachen HTML und CSS, sowie den Prozess der Entwicklung einer Website haben, um einen praktischen Nutzen aus dieser Arbeit ziehen zu können. Da sich das Thema Responsive Webdesign in einem ständigen Wandel befindet, erhebt diese Arbeit keinerlei Anspruch auf Vollständigkeit.

1.0.4 Motivation zur Arbeit

Mein Studium an der Fachhochschule Potsdam vermittelte mir fundierte Grundlagen über die Konzeption und Anforderungen an moderne Informationssysteme. Außerdem wurden Kenntnisse in Beschreibungssprachen wie z. B. HTML, CSS und XML, sowie Programmiersprachen wie z. B. PHP und der Einsatz von Datenbanken vermittelt. Neben meinem Studium war ich als Werkstudent in den unterschiedlichsten Unternehmen im Bereich Internetwirtschaft, Webdesign und Programmierung tätig und erlangte die notwendigen praktischen Fähigkeiten für die Umsetzung von modernen Websites. Im Juni 2012 machte ich mich als freiberuflicher Webdesigner und Webentwickler neben dem Studium selbständig und eignete mir in dieser Zeit ein umfangreiches Wissen an Ansätzen, Methoden und Lösungsmöglichkeiten zur Umsetzung von responsiven Websites an. Mit Beginn des Jahres 2014 schloss ich mich der imwebsein Online Marketing Agentur an und bin fort an für die Umsetzung von responsiven Websites für kleine und mittelständische Unternehmen zuständig. In der praktischen Arbeit bleibt allerdings selten die Zeit für eine theoretische Auseinandersetzung mit einem komplexen Thema wie Responsive Webdesign. Diese Möglichkeit möchte ich im Rahmen meiner Bachelorarbeit nachholen.

2 Responsive Webdesign

2.1 Was ist Responsive Webdesign?

Der Begriff Responsive Webdesign umschließt alle Konzepte, Methoden und Lösungsansätze zur Umsetzung von reaktionsfähigen Websites. Er wurde geprägt durch den amerikanischen Webdesigner Ethan Marcotte, der bis heute als Begründer des Responsive Webdesigns gilt. In seinem Artikel(Mar10) vom 25.10.2010 im Blog von alistpart.com nahm er Bezug auf eine neuartige Disziplin der Architektur mit dem Namen Responsive Architecture. In diesem Bereich der Architektur versucht man Gebäude zu entwerfen, die auf unterschiedliche Umwelteinflüsse reagieren können. Diese Idee versuchte Marcotte auf die Gegebenheiten des moderenen Webdesigns anzuwenden und legte damit den Grundstein für unser heutiges Verständis von Responsive Webdesign. Technologisch betrachtet handelt es sich bei Responsive Webdesign nicht um einen aktuellen Trend des Webdesigns, sondern viel mehr eine Rückbesinnung auf die Flexibilität von HTML-Dokumenten. Die Übernahme der gestalterischen Gesetze aus den Grafik-und Printbereichen führten erst dazu, dass man versuchte eine künstliche Leinwand[1] zu erzeugen und die klassischen gestalterischen Grundlagen aus diesen Bereichen zu übernehmen. Es wurde versucht dem Designer die Kontrolle zu geben, die er braucht, um eine Leinwand zu gestalten, ohne dabei die Flexibilität des Internets zu berücksichtigen. Diese Methode der Entwicklung von Websites wurde über Jahre praktiziert. Erst die steigende Anzahl an mobilen Internetnutzern (vgl. dazu(AO13)) und die Masse an unterschiedlichen, mobilen Endgeräten führte zu einem Paradigmenwechsel im Schaffensprozess einer Website. Die fehlende Standardisierung der Hersteller machte es unmöglich eine spezielle Website für jeden Gerätetyp anzufertigen und so trat die Notwendigkeit der Entwicklung im Sinne des Responsive Webdesigns ins Bewusstsein der Webdesigner und Webentwickler.

1 Eine Künstliche Leinwand meint eine gestalterische Fläche mit klaren Abmessungen in Länge und Breite

2.2 Der Responsive-Webdesign-Worklow

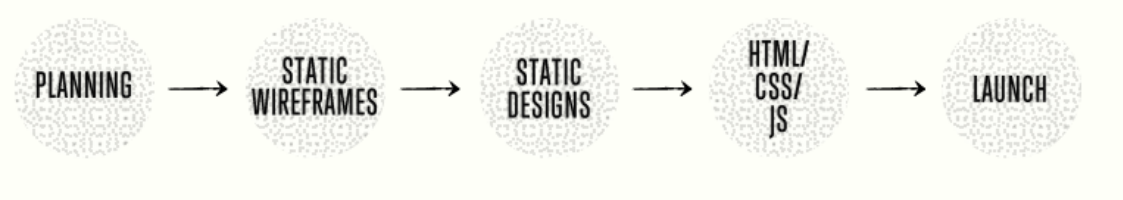

Abbildung 2.1: Der bisherige Workflow im Webdesign

Die folgende Grafik zeigt den bisherigen Workflow zur Konzeption, Gestaltung und Entwicklung von Websites. Jedes Projekt begann mit einer Planungsphase. Diese enthielt Arbeitsschritte zur Strukturierung der Informationen und Überlegungen zu Navigations-und Inhaltskonzepten. Hierfür werden bis heute Sitemaps[2] und Strukturmodelle benutzt, um die Inhalte logisch und verständlich präsentieren zu können. Im zweiten Schritt wurden einfache Entwürfe, sogenannte Mock-Ups[3], angefertigt, um einen grundlegenden Aufbau der Website zu visualisieren. Anschließend erfolgte die Umsetzung in einem statischen Design. Dieses statische Design diente zur Präsentation und vermittelte ein nahezu vollständiges Bild der zukünftigen Website. Waren alle Änderungen und Ausarbeitungen am statischem Design abgeschlossen, wurde es mit Hilfe der gängigen Frontend-Technologien wie HTML, CSS und JavaScript für den Browser umgesetzt. Auch die Programmierung von serverseitigen Backends[4] fand innerhalb dieses Prozessschrittes statt. Abschließend wurde eine Fehlerkontrolle vollzogen und das fertige Projekt auf den Server des Auftraggebers übertragen.

Abbildung 2.2 zeigt einen möglichen responsiven Workflow. Der erste Schritt ist auch hier die inhaltiche Planung des Projektes. Im nächsten Schritt wird auf die Visualisierung als Mock-Up und die Umsetzung in ein statisches Design verzichtet, stattdessen wird ein interaktiver HTML-Prototyp enwickelt (vgl. (Pet14)). In diesem Prozessabschnitt testet man den Prototypen in unterschiedlichen Auflösungen und auf verschiedenen Displaygrößen und passt ihn entsprechend an. Basierend auf dem Prototypen startet nun ein iterativer Prozess zwischen Design und Entwicklung der Website.

2 Sitemaps dienen der einfachen Visualisierung von Navigationspfaden in einer Website
3 Mock-Ups sind einfache Skizzen die den grundlegenden Aufbau der Website zeigen. Sie werden primär dafür genutzt um Inhaltsbereiche festzulegen
4 Serverseitige Backends geben die Möglichkeit Inhaltseingaben in einem einfachen Editor umzusetzen, ohne dabei auf Webtechnologien zurückgreifen zu müssen

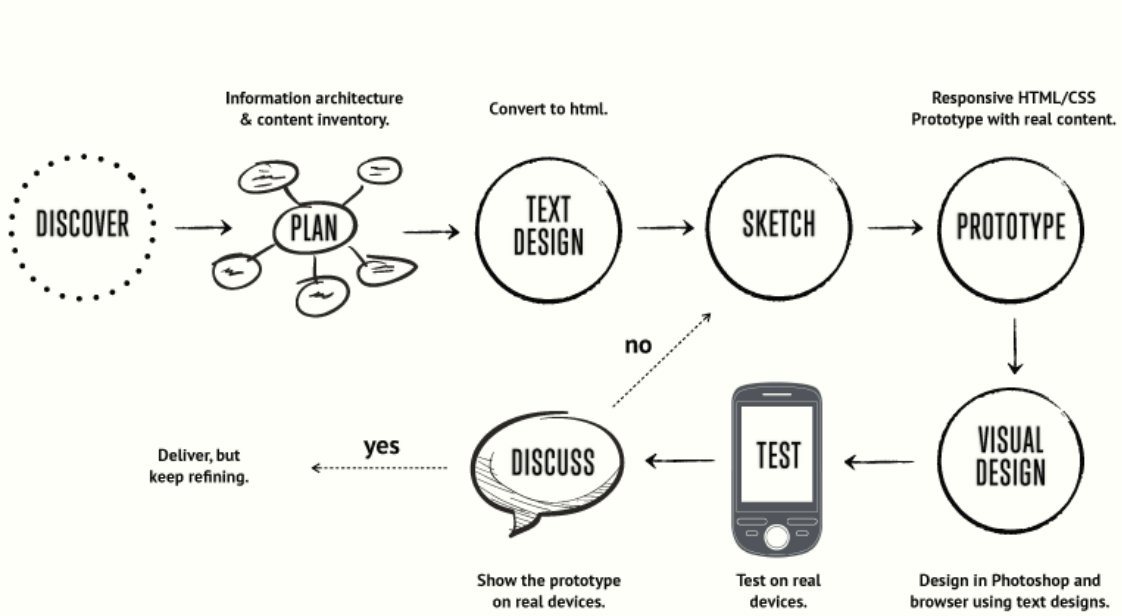

Abbildung 2.2: Der responsive Workflow im Webdesign

Man gliedert die Bestandteile einer Website in Module, so wird z. B. die Navigation vom Designer gestaltet, gleich durch den Entwickler technisch umgesetzt und im interaktiven Prototypen[5] getestet.

So besteht die Möglichkeit einer schnellen Korrektur, falls Elemente nicht wie geplant ins Design der Website passen. Dieser Prozess wird solange wiederholt, bis ein Modul abgeschlossen ist und mit dem Nächsten begonnen werden kann. Dieser Prozess ist durch seinen iterativen Charakter äußerst robust und optimal für den neuen Workflow des Responsive Designs geeignet.

5 Interaktive Prototypen sind die Weiterentwicklung eines Mock-Ups. Sie können sich dynamisch auf unterschiedliche Displaygrößen anpassen und so getestet werden

2.3 Graceful Degradation, Progressive Enhancements und Mobil First

Im letzten Abschnitt ging es um die Unterschiede des klassischen und des responsiven Workflows. Dabei wurden alle Prozessschritte von der Konzeption bis zur Gestaltung und Entwicklung der Website betrachtet. In diesem Abschnitt soll es nun um drei wesentliche technische Ansätze zur Entwicklung von responsiven Websites gehen.

Die beiden Ansätze Graceful Degradation und Progressive Enhancement im Kontext des Webdesigns gelten als Grundlage der browserkompatiblen Entwicklung von Websites. Sie unterscheiden sich primär in ihrer Betrachtungsweise der Entwicklung. Graceful Degradation betrachtet eine Website aus Sicht von modernen Geräten wie z. B. neuen Desktop-Computern mit den aktuellsten Browserversionen. Progressive Enhancement hingegen betrachtet eine Website aus Sicht von älteren Geräten wie z. B. älteren Notebooks mit Windows XP und einem Internet Explorer der Version 7.

2.3.1 Graceful Degradation

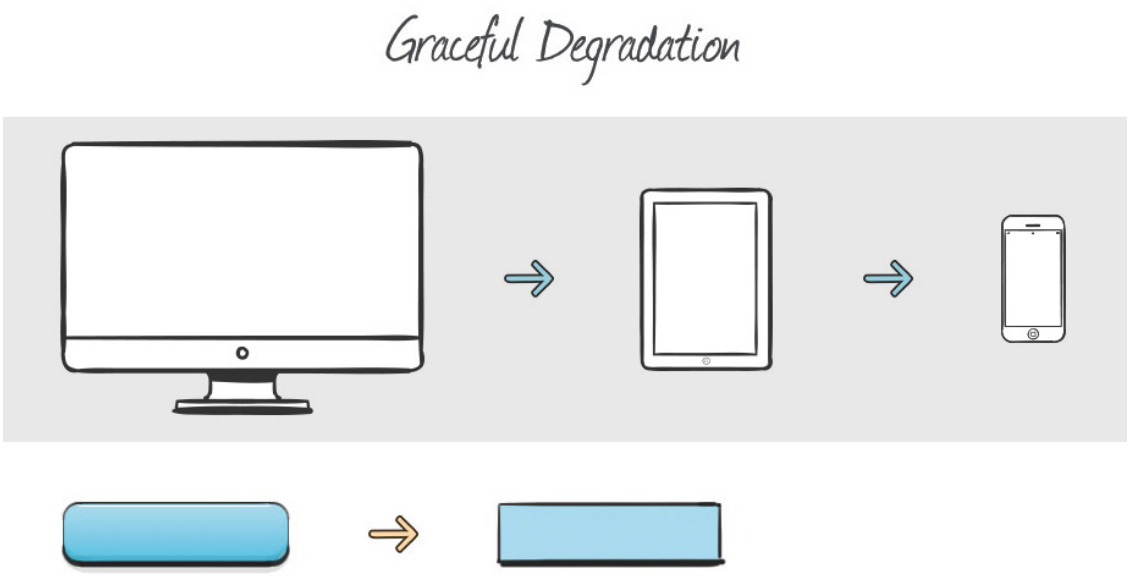

Abbildung 2.3: Graceful Degradation nach Jonas Hellwig

Der Ansatz von Graceful Degradation entspricht der bisherigen Arbeitsweise vieler Agenturen und Unternehmen. Man arbeitet mit modernsten Computern und den aktuellsten Browserversionen und entwickelt die Website passend für seine eigenen technischen Möglichkeiten. Ist die Website

auf modernen Geräten fertig gestellt, versucht man über entsprechende Fallback-Lösungen[6] die Kompatibilität für ältere Browser und Geräte zu ermöglichen. Gängige Praxis ist z. B. der Einsatz von JavaScript. Ist kein JavaScript aktiviert, wird eine vereinfachte CSS-Version des jeweiligen Elements geladen. Diese Herangehensweise entspricht im Wesentlichen dem klassischen Workflow im Webdesign. Die Website wird inhaltlich geplant, in einem statischen Design festgehalten und anschließend programmiert.

Die Wahrung der Kompatibilität für ältere Browser und Systeme wird bei dieser Herangehensweise zunehmend zum Problem. Die Menge an Fallback-Lösungen erfordert ein Vielfaches mehr an Entwicklungszeit und sorgt, durch die Steigerung der Dateigröße und der steigenden Menge der Anfragen an den Server, für deutliche Probleme bei der Performance der Website (vgl. dazu(Hel14a)) Auch die Positionierung der Inhalte auf kleinen Endgeräten gestaltet sich oftmals schwierig. Durch die inhaltliche Planung auf Basis von großen Bildschirmen ist es nahezu unmöglich alle Informationen sinnvoll auch auf kleineren Displaygrößen zu bereitzustellen. Die Frage, ob es Sinn macht, Inhalte für Besucher mit einem kleineren Display zu kürzen, klärt Karen McGrane in ihrem Artikel: Responsive Design Won't Fix Your Content Problem auf alistapart.com (McG13). Dennoch hat der Ansatz des Graceful Degradation durchaus seine Vorteile. Er entspricht dem klassischen Arbeitsablauf der Entwicklung von Webseiten. Gerade in größeren Agenturen ist es fast unmöglich in kürzerer Zeit die kompletten Arbeitsabläufe stark umzustrukturieren.

Der Fokus auf starke Systeme führt außerdem häufig dazu, die gegebenen Möglichkeiten auch auszunutzen und so auf neueste Technologien zu setzen. Ein letzter wichtiger Aspekt, der für Graceful Degradation spricht, ist das Re-Design von bereits bestehenden Websites mit begrenztem Budget. Man wandelt die bisherige statische Website modular in eine responsive Website um. Die Inhalte werden entsprechend umstrukturiert und bestimmte Elemente bei kleineren Displaygrößen werden ausgeblendet (vgl. dazu(ZU)). Das größte Problem ist die unzureichende Unterstützung von schwachen Systemen. Die Gewährleistung der Performance, Browserkompatibilität und Barrierefreiheit kann nur mit einem deutlichem Zuwachs an Entwicklungszeit bewerkstelligt werden. Die Erweiterbarkeit der Website ist durch die aufwendigen Testmaßnahmen ebenfalls deutlich eingeschränkt (vgl. dazu(Hel14a)).

6 Fallback-Lösungen sind meistens über JavaScript realisierte Nachrüstungen von Funktionalitäten in älteren Browsern

2.3.2 Progressive Enhancement

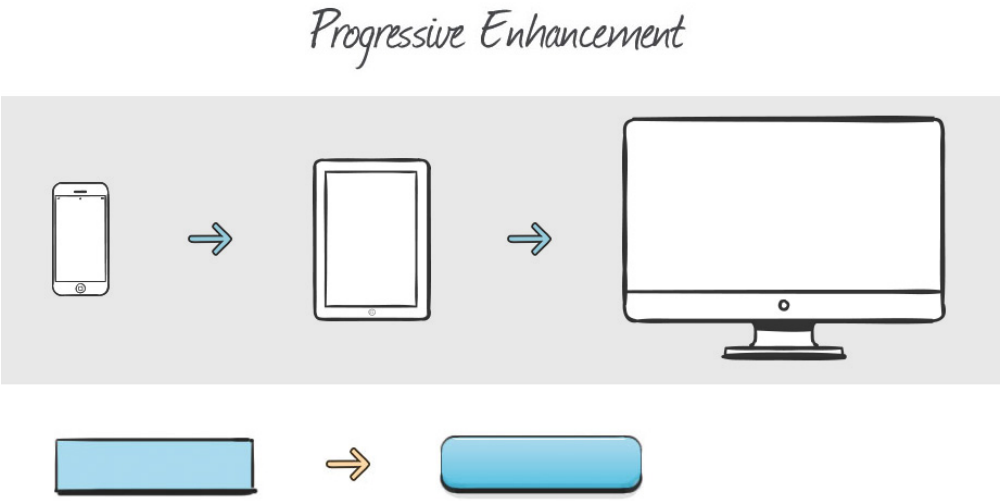

Abbildung 2.4: Progressive Enhancement nach Jonas Hellwig

Im Gegensatz zu Graceful Degradation bezieht sich die Sichtweise der Entwicklung bei Progressive Enhancement auf die schwachen Systeme. Diese Herangehensweise stellt Agenturen und Unternehmen vor die Herausforderung des Umdenkens. Den Ausgangspunkt stellt das schwächste zu unterstützende, System. Sobald die Basis-Version der Website fertig gestellt wurde, widmet man sich dem stärkeren System (vgl. dazu(Hel14a)). Die Menge der zu unterstützenden Systeme variiert in der Umsetzung je nach Kostenumfang und Zeitbudget. Eine klassische Technik in diesem Zusammenhang nennt sich Feature Detection[7]. Dabei wird versucht mit Hilfe von Anfragen an den Browser die technischen Spezifikationen des Geräts zu ermitteln. Mit Hilfe des Ansatzes Progressive Enhancement ist es möglich eine Website zu entwickeln die den Anforderungen an Performance, Browserkompatiblität und Barrierefreiheit gerecht wird. Auch die Erweiterbarkeit ist durch Abwärtskompatibilität gewährleistet. Erweiterungen müssen nur im schwächsten möglichen System getestet werden und sind dann für alle kommenden Versionen kompatibel. Dieser Ansatz scheint deutlich besser auf die Anforderungen eines Responsive Webdesign Workflows reagieren zu können. Der Ansatz des Progressive Enhancements wurde durch weitere Aspekte der Betrachtung z. B. unterschiedliche Bildschirmgrößen oder Auflösungen, erweitert. Diesen Ansatz nennt man Mobil-First-Progressive Enhancement (vgl. dazu(Hel14a)).

7 Feature Detection ist eine über Cookies realisierte Abfrage der technischen Möglichkeiten des Sytems mit dem der Besucher sich auf der Website befindet.

2.3.3 Mobil First

Abbildung 2.5: Mobil First nach Jonas Hellwig

Mobil First konzentriert sich, im Gegensatz zu Graceful Degradation und Progressive Enhancement, auf mobile Geräte. Im Vordergrund stehen die Inhalte einer Website aus Sicht eines mobilen Endgerätes (vgl. dazu(Wro13a)). Die Gestaltung der Website folgt der optimalen Präsentation der Inhalte auf kleinen Displaygrößen und bringt die Notwendigkeit einer inhaltlichen Fokussierung mit sich. Die Konzentration der Inhalte und der Gestaltung sorgen für eine saubere Informationsarchitektur sowohl auf mobilen Geräten, als auch auf Desktop-Computern. Der Prozess der Erstellung orientiert sich an der Herangehensweise des Ansatzes Progressive Enhancement. Inhaltskonzeption, modulare Gestaltung, Feature-Detection sind elementare Bestandteile beider Ansätze. Mobil First erweitert das Testing um das sogenannte Unit-Testing, wie es mittlerweile auch in allen modernen Entwicklungsprozessen, Standard ist. Man testet die Website auf physischen, mobilen Geräten. Dabei kann es sich um ein iPhone mit Safari-Mobile-Browser, um ein Smartphone mit Google-Chrome-Browser oder ein älteres Feature-Phone mit einem Opera-Mini-Browser handeln. Mit Hilfe dieser Herangehensweise erreicht man eine maximale Kompatibilität und eine optimale Darstellung auf unterschiedlichen Endgeräten. Ein weiterer Vorteil der Fokussierung nach Mobil-First ist der zusätzliche Performance-Gewinn. Während man im Progressive Enhancement damit begonnen hat, mit Hilfe von Feature-Detection Fähigkeiten der Systeme zu erkennen, versucht man im Ansatz des Mobil Firsts auch native Funktionen der Smartphones zu nutzen. So kann man z. B. durch den Einsatz von HTML5-Elementen für Formulare die Gestaltung dem jeweiligen Betriebssystem überlassen und spart CSS-Code (vgl. dazu(Wro13b)). Der Nachteil liegt, wie beim Ansatz des Progressive Enhancements, im iterativen Prozess der Entwicklung. Außerdem sorgt das Unit-Testing für einen Kostenfaktor, da unterschiedliche physische Endgeräte eingekauft werden müssten. Durch dieses Problem haben sich mittlerweile einige Spezialdienstleister auf dem Markt etabiliert, die eine umfangreiche Testing-Suite gegen Gebühr zur Verfügung stellen.

3 Umsetzung einer responsiven Website

Responsive Webdesign hat Einfluss auf alle wesentlichen Elemente einer Website. In diesem Kapitel sollen alle zentralen Bestandteile betrachtet und Lösungsansätze für den Umgang im Responsive Webdesign vorgestellt werden.

3.1 Einsatz von responsiven Grids

Der Einsatz von Rastern, nachfolgend Grids genannt, ist seit Jahren ein bewährtes Werkzeug zur Entwicklung von professionellen Websites. Marcotte spricht in seinem Buch (Mar11) allerdings von flexiblen und anpassungsfähigen Grids. Diese flexiblen Grids bieten uns die Möglichkeit, auf unterschiedliche Bildformate in der Darstellung reagieren zu können.

3.1.1 Aufbau

Ein Grid (oder auch Grid-System) besteht aus einer bestimmten Anzahl von Spalten und Zeilen. Die Spalten nehmen dabei einen prozentualen Anteil der Gesamtbreite eines Bildschirms ein. Die Zeilen umschließen die Spalten und haben immer eine Breite von 100 Prozent. Das ermöglicht die saubere Positionierung der Elemente untereinander. Ein passendes Beispiel für den früheren Einsatz von Grids ist das pixelbasierte 960-Gridsystem (Smi). Dieses zwölf-spaltige Grid diente als Grundlage für eine Vielzahl von Websites. Seine geringe Gesamtbreite von 960 Pixeln ermöglichte eine problemlose Darstellung der Website auf allen gängigen Bildschirmauflösungen von Desktops und Notebooks.

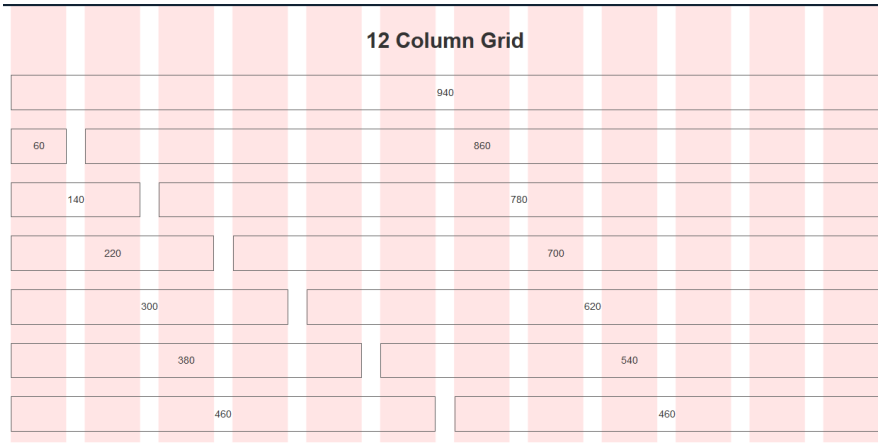

Abbildung 3.1: Das pixelbasierte 960-Grid

3.1.2 Entwicklung

Die Planung, Konzeption und Umsetzung von anpassungsfähigen Grids ist zu einer Grundqualifikation für Webdesigner geworden. Sie basiert auf einer soliden Inhalts-und Strukturplanung der gesamten Website und muss alle Eventualitäten des späteren Designs der Website abbilden können. Im Regelfall gilt, je feiner die gestalterische Ausarbeitung des Designs und die Positionierung der Elemente, desto mehr Spalten werden benötigt. Zu Beginn der Entwicklung eines Grids steht die Entscheidung der Maximalbreite unter Berücksichtigung der Zielgruppe. So kann z. B. ein 1140 Pixel breites Grid für die Darstellung auf kleineren Bildschirmen besser geeignet sein als ein 1600 Pixel breites Grid für eine umfangreiche multimediale Website. Ist die Maximalbreite des Grids erreicht, wird die Website auf dem Bildschirm zentriert und die Außenabstände werden vergrößert. Durch die Berücksichtigung der Innen-und Außenabstände der Spalten zueinander verkleinert sich außerdem die Breite der einzelnen Spalten.

3.1.3 Berechnung

Als Grundlage für die Berechnung eines responsiven Grids soll das pixel-basierte 960-Grid dienen. Die Maximalbreite, die das Grid einnehmen kann, wird demnach auf 960 Pixel festgelegt. Bei einem zwölfspaltigen Grid ergibt sich eine Spaltenlänge von 8,333333333333333 Prozent (80*100/960). Nun müssen noch die Innen-und Außenabstände in die Planung des Grids mit aufgenommen werden. Die Breite der Spalten wird mit dem Innenabstand jeder Spalte und dem Außenabstand des gesamten Grids addiert. Das Ergebnis sollte 960 Pixel sein. Bei einem 960 Pixel breitem Grid bietet sich ein Innen-und Außenabstand von 24 Pixeln an, das entspricht 2,5 Prozent (24*100/960). Abschließend müssen die 2,5 Prozent von den 8,333333333333333 Prozent der Spaltenlänge subtrahiert werden und wir erhalten einen Wert von 5,833333333333333 Prozent.

Zusammenfasst haben wir eine Spaltenbreite von 5,833333333333333 Prozent, die addiert mit den Innen-und Außenabständen, eine Gesamtbreite von 8,333333333333333 Prozent pro Spalte auf dem Bildschirm einnimmt. Diese bilden ein 100 Prozent breites, zwölfspaltiges Grid (8,333333333333333*12 =100 Prozent), welches für die meisten Anforderungen in der Umsetzung einer responsiven Website ausreichen sollte. Mit dem, von Jonas Hellwig in seiner Video-Schulung Responsive Webdesign (Hel13a) vorgestelltem, Tool gridcalculator (Gri), lassen sich Pixel-basierte Grids nahezu automatisiert berechnen und verwenden.

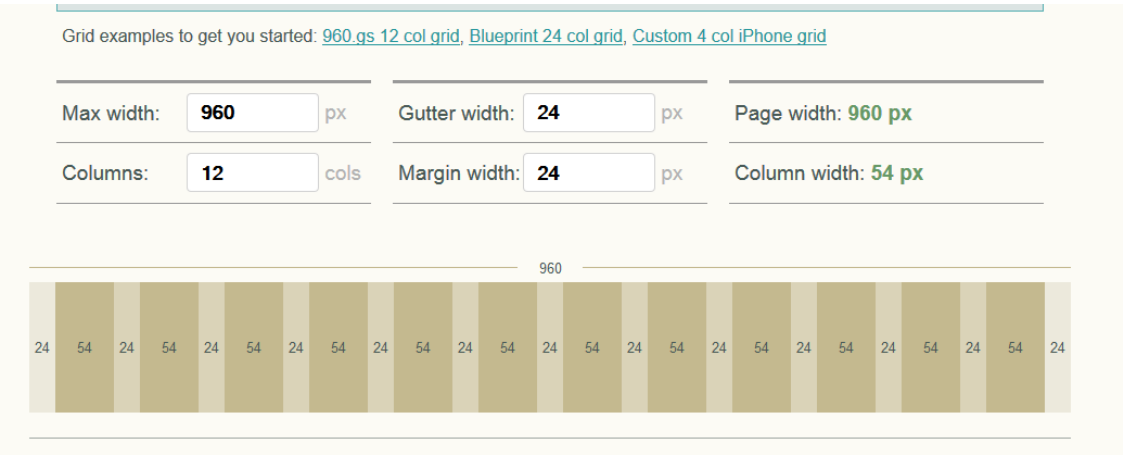

Abbildung 3.2: Das relative 960-Grid

Eine ausführlichere Umrechnung eines pixelbasierten Grids in ein responsives Grid beschreibt Christoph Zillgens in seinem Buch. Er liefert eine schrittweise Anleitung zur Umrechnung der einzelnen Elemente einer Website (Zil13a).

Border-Box-Modell

Die Planung eines responsiven Grids, unter Berücksichtigung der Innen-und Außenabstände der Spalten zueinander, kann ohne Zuhilfenahme von Tools zur automatisierten Berechnung einen beachtlichen Zeitaufwand erfordern. Durch die Spezifikation des Border-Box-Modells in CSS3 durch das W3C

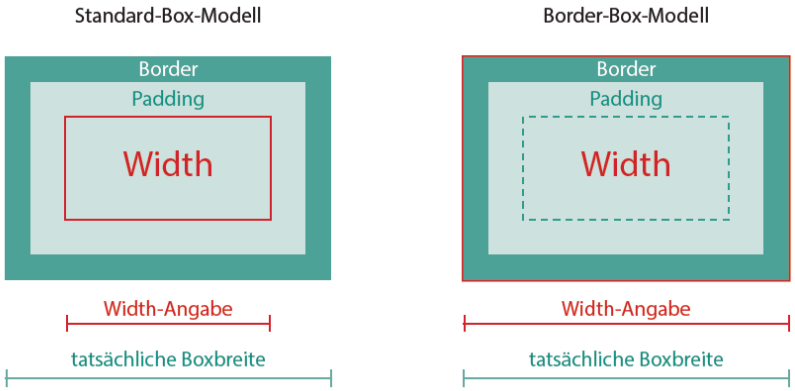

Abbildung 3.3: Das Border-Box-Modell

Das bisherige Box-Modell addierte die Innen-und Außenabstände zur ursprünglichen Größe des Elements und sorgte für eine vergrößerte Darstellung des selbigen Elements. Das Border-Box-Modell berechnet die Abstände innerhalb der eigentlichen Breite des Elements. Diese Anpassung vereinfacht die Arbeit mit Abständen im Kontext des responsive Webdesigns deutlich.

3.2 Der Einsatz von Mediaqueries

Durch die Spezifikation von CSS3 durch das W3C (W3C14) wurden die bekannten Medientypen um den Einsatz von Mediaqueries erweitert. Durch den Einsatz von Mediaqueries wird dem Webdesigner die Möglichkeit der Definition von Breakpoints gegeben. Breakpoints sind Umbruchpunkte innerhalb des Layouts einer Website die eine Neustrukturierung der bisherigen Elemente, unter Berücksichtigung der definierten Eigenschaften, ermöglichen. Mit Hilfe von Mediaqueries können z. B. Breakpoints für bestimmte Displaygrößen definiert werden. Außerdem besteht die Möglichkeit der Abfrage der Bildschirmauflösung. Diese Abfrage hat sich vor allem im Bezug der Bildoptimierung für Retina-Bildschirme bewährt.

3.2.1 Enstehung

Die Mediaqueries gehen aus den Medientypen, welche schon in der Spezifikation von CSS2 definiert wurden, hervor. Damals nutze man die Medientypen z. B. für die Bereitstellung einer gesonderten Darstellung für Druck oder den Einsatz auf Beamern. Die fehlende Erkennung des Medientyps „handheld“ durch moderne Smartphones machte die Bereitstellung einer mobilgerechten Version einer Website unmöglich. Aus diesem Grund entschloss sich das W3C zur Einführung von Unterabfragen des Medientyps „screen“.

3.2.2 Definition

Mediaqueries können sowohl als gesonderte CSS-Datei, als auch inline im CSS-Dokument deklariert werden. Beispiele für die Definition in einer gesonderten CSS-Datei und den Inline-Einsatz mit Hilfe von @media liefert Zillgens in seinem Buch (Zil13b). Beide Möglichkeiten sollen hier kurz aufgezeigt werden:

```
1  <link rel="stylesheet" media="screen and (min-width: 800px)" href="screen.css"/>
```

Abbildung 3.4: Externe Einbindung von Mediaqueries

```
1  @media screen and (min-width: 800px) {
2  */ Hier folgen die CSS-Angaben für den Viewport /*
3  }
```

Abbildung 3.5: Inline-Einbindung von Mediaqueries

Mit Hilfe von Mediaqueries können eine Vielzahl von weiteren Eigenschaften abgefragt werden, so z. B. die Orientierung des Seitenformats (Querformat und Hochformat), oder die durch das Gerät darstellbaren Farben (16-Bit oder 256-Bit). In der Praxis werden die meisten Eigenschaften allerdings nur selten benutzt. Eine Auflistung der möglichen Abfragen durch Mediaqueries liefert selfhtml.org (sel14).

Abfrage des Viewports

Die Apple Inc. benutzte für die Darstellung von Websites auf dem iPhone der ersten Generation ein spezielles Rendering. Responsive Websites galten zu diesem Zeitpunkt als Ausnahmefall, weshalb das iPhone eine automatische 100-Prozent-Skalierung der Website auf die Bildschirmbreite des Gerätes vornahm. Diese Technik der Skalierung wurde von anderen Geräteherstellern adaptiert und führte zu dem Problem der falschen Darstellung von responsiven Websites auf mobilen Endgeräten. Im Zuge der Verbreitung von Responsive Webdesign lieferte die Apple Inc. eine Lösung des Problems mit Hilfe der Metaangabe des Viewports.

Die Meta-Angabe wird wie folgt im Head-Bereich der Website definiert:

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

Abbildung 3.6: Viewport-Metaangabe

Die Angabe des Viewports löste das Problem der Skalierung und gilt bis heute als Standard-Eigenschaft für responsive Websites. Einen visuellen Vergleich zeigt Zillgens (Zil13c).

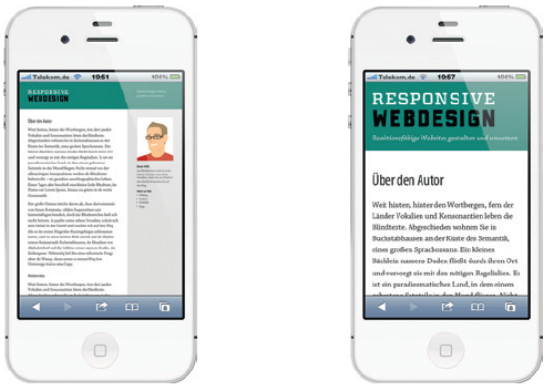

Abbildung 3.7: Viewport-Vergleich

3.3 Responsive Bilder

Bilder sind eine der größten Herausforderungen in der Umsetzung von responsiven Websites. Die Reaktionsfähigkeit von pixelbasierten und statischen Bildquellen erfordert unterschiedliche Lösungsansätze für eine qualitative Umsetzung im Responsive Webdesign. Die einfachste Form für eine grundlegende Anpassung von Bildern nach dem Responsive Webdesign ist die prozentuale Skalierung. Diese wird innerhalb einer CSS-Datei folgendermaßen definiert:

```
1  img {
2  Height:auto;
3  Max-width:100%;
4  }
5
```

Abbildung 3.8: Einbindung von responsiven Bildern

Die Angabe zur maximalen Breite überschreibt die statische Breitenabmessung der Bilder. Stattdessen orientiert sich die Breite an den Abmessungen des umschließenden Elements. Die Angabe der automatischen Höhe sorgt für eine Skalierung des Bildes. Die Höhenangabe sorgt dafür die Proportionen von Bildern zu erhalten um einer Verzerrung vorzubeugen.

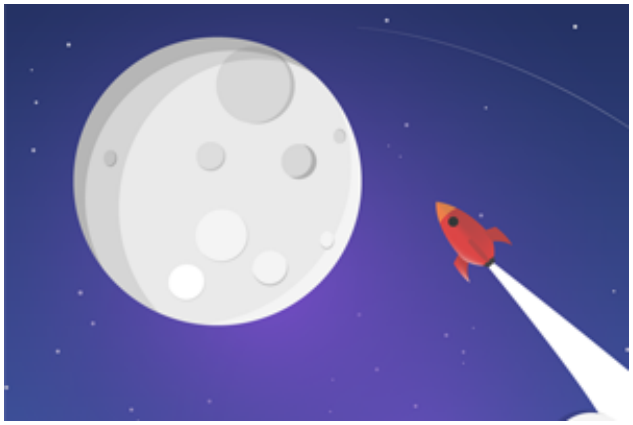

Abbildung 3.9: Einbindung von responsiven Bildern mit height:auto

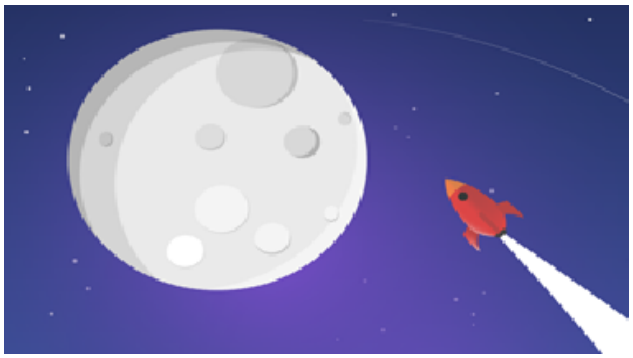

Abbildung 3.10: Einbindung von responsiven Bildern ohne height:auto

Diese Angabe führte zur grundlegenden Möglichkeit der Verwendung von Bildern im Responsive Webdesign. Diese Angabe ist für die Vielzahl an anderen Problemen mit Bildern im reaktionsfähigen Kontext allerdings nicht ausreichend. Die wesentlichen Probleme lassen sich in drei Kategorien einteilen: Datenmenge und Datenrate, Berücksichtigung der Bildaussage und die Bereitstellung von hochauflösenden Bildern. Diese Probleme werden in Kurzform in Zillgens Buch (Zil13d) geschildert und sollen als Grundlage der Strukturierung dieses Themas dienen.

3.3.1 Datenmenge-und Datenrate

Die Datenmenge und Datenrate spielt eine große Rolle bei der Bereitstellung von Bildern im Responsive Webdesign. Geringe Übertragungsraten und ein hohes Aufkommen an Datenmengen sorgen für ein Problem der performance-optimierten Bereitstellung auf mobilen Endgeräten. Je größer die Menge der bereitzustellenden Daten, desto länger muss ein Besucher auf die angeforderten Informationen warten. Einen Großteil der Datenmenge einer Website stellen die Bilder. Im Folgenden sollen nun mögliche Lösungsansätze für eine ladezeiten-freundliche Bereitstellung von Bildern in einer reaktionsfähigen Website vorgestellt werden.

Adaptive Images

Matt Willcox ist der Entwickler der server-und clientseitigen Lösung «Adaptive Images»(Ima14). Dieses Script ermöglicht eine automatisierte Verkleinerung der Bilder in Abhängigkeit der Displaygröße des betrachtenden Gerätes. Das clientseitige[1] Javascript liest mit Hilfe von Cookies[2] die Geräteeigenschaften aus und das serverseitige PHP-Script sorgt für eine Umrechnung und die Speicherung auf dem Server. Sobald die Website mit einem neuen Endgerät aufgerufen

1 clientseitig = Aus Sicht des Browsers
2 Cookies sind Textdateien die zur Speicherung von Systeminformationen auf dem Computer des Benutzers gespeichert werden.

wird, werden die Geräteeigenschaften ausgelesen, das Bild rechnerisch verkleinert und physisch auf dem Server abgelegt. Wenn nun ein anderer Besucher mit den gleichen Geräteeigenschaften die Website besucht, stellt das Script die bereits verkleinerte Version des Bildes zur Verfügung.

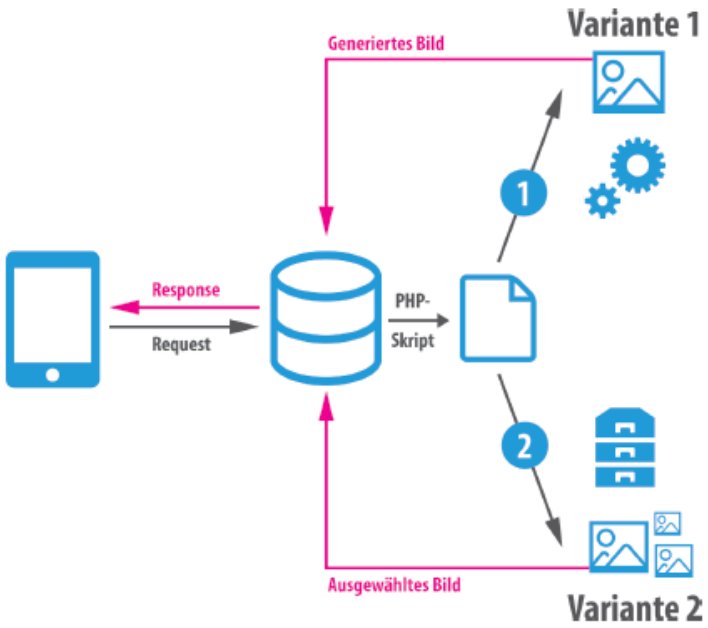

Abbildung 3.11: Adaptive Images

Vorteile:

Ein wesentlicher Vorteil ist die dauerhafte Bereitstellung der komprimierten Bilder auf dem Server. Dadurch können komprimierte Bilder, ohne weitere Anpassung, für eine Vielzahl von Geräten bereitgestellt werden. Außerdem läuft dieses Verfahren vollständig automatisiert und kann auch nachträglich in bestehende Websites implementiert werden, da keine Anpassungen innerhalb des Quellcodes notwendig sind.

Nachteile:

Die Notwendigkeit von aktivierten Cookies macht dieses Verfahren anfällig für Browser ohne Javascript-Unterstützung. Die Bilder werden dann in ihrer ursprünglichen Größe ohne Berücksichtigung auf Displaygröße ausgeliefert. Die Automatisierung führt außerdem zu Problemen bei der Berücksichtigung von Bildausschnitten (siehe dazu Unterkapitel: Bildaussage). Eine ausführliche Anleitung zur Installation und Inbetriebnahme von Adaptive Images findet sich auf der gleichnamigen Website (Ima14).

Das srcset-Attribut

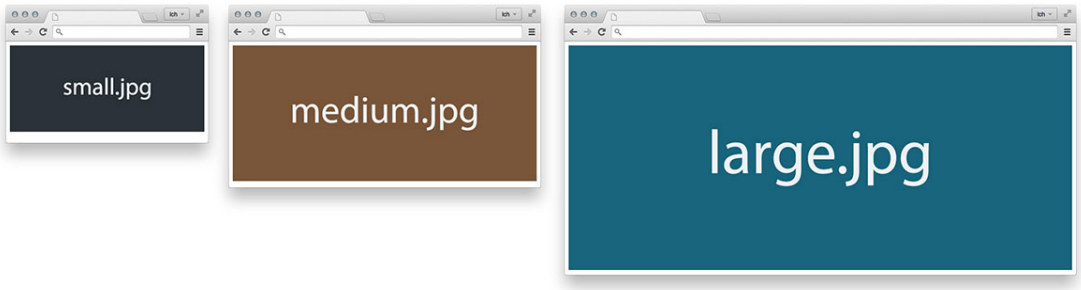

Abbildung 3.12: Src-Set Visualisierung - Jonas Hellwig

Das HTML-Element «img»lässt sich mit Hilfe des Attributs «srcset»besser für den Einsatz im Responsive Webdesign nutzen. Innerhalb des srcset-Attributs ist es möglich, dem Browser Empfehlungen für eine optimierte Version zum aktuellen Gerätetyp anzubieten. Diese Empfehlungen werden durch den Browser mit den Informationen zu den Geräteeigenschaften (aus Cookies) verglichen und bei Übereinstimmung das angepasste Bild geladen.

Definition des srcset-Attributs

```
1  <img src="beispiel.jpg" srcset="beispiel.jpg 320w, beispiel-medium.jpg 600w, beispiel-large.jpg 900w"
   alt="">
```

Abbildung 3.13: Definition des src-Attributs

Innerhalb des «img»Elements wird zunächst die Bildquelle mit Hilfe des Attributs «src»eingebunden. Im «srcset»Attribut werden nun angepasste Versionen des Bildes für bestimmte Bildschirmgrößen hinterlegt. Die Einheit «w»beschreibt die physikalische Größe des Bildes und ist notwendig um dem Browser die Möglichkeit des Abgleiches zu geben (vgl dazu:(Hel14b)).

Vorteile:

Das srcset-Attribut ermöglicht eine Bereitstellung von Bildern für unterschiedliche Displaygrößen ohne den Einsatz von Javascript. Durch die Angabe der Displaybreite mittels der Einheit «w»lassen sich optimierte Bilder für eine Vielzahl von Bildschirmauflösungen bereitstellen.

Nachteile:

Obwohl der Browser über die notwendigen Informationen zur Anpassung der Bilder an den entsprechenden Viewport besitzt, kann er diese nur langsam interpretieren und verarbeiten. Hier liegt auch der Nachteil des srcset-Attributs. Wird z. B. ein Bild mit einer Breite von 320w für kleinere Bildschirme bereitgestellt, der Nutzer verfügt allerdings nur über eine Darstellungsbreite von 300 Pixeln, kann der Browser die Grafik nicht schnell genug skalieren.

Eine Lösung für dieses Problem liefert die Angabe von Breiten innerhalb der CSS-Datei mittels der Einheit «vw». Diese Eigenschaft ermöglicht eine relative Angabe des Bild-Elements zur aktuellen Viewport-Breite (vgl. (Web14)).

CSS-Sprites

Neben der Dateigröße von Bildern sind die sogenannten «HTTP-Requests»ein bedeutender Faktor in Bezug auf die Ladegeschwindigkeit einer Website. HTTP-Requests sind Anfragen durch den Browser an den Server. Hierbei werden alle notwendigen Dateien für die Darstellung der Website geladen. Je mehr Anfragen ein Browser an den Server stellen muss, desto länger die Ladezeit der gesamten Website. CSS-Sprites dienen dem Zusammenfassen von kleineren Bildern innerhalb einer großen Datei. Mit Hilfe eines beliebigen Bearbeitungsprogramms, wie z. B. Photoshop, werden diese Bilder mit einem gewissen Abstand zueinander in einer Datei gebündelt.

Sind die Grafiken zusammengefügt, werde sie mit der CSS-Eigenschaft «background-image»eingebunden und mit «background-position»positioniert.

Ein praktisches Beispiel liefert Jonas Hellwig in seinem Video-Training «Responsive Webdesign»(Hel13b):

Abbildung 3.14: Die erstelle Sprite-Grafik in Photoshop

```
.twitter {background-position:0 0;}
.googleplus {background-position:-87px 0;}
.facebook {background-position:-173px 0;}
```

Abbildung 3.15: Der CSS-Code für die Positionierung der Elemente

Für die Ansprache mit Hilfe von CSS wurde jedem Link eine entsprechende Klasse zugewiesen. Die Sprite-Grafik wird auf dem Link-Element platziert, so dass alle drei Klassen das gleiche Hintergrundbild bekommen. Mit Hilfe der Positionierung über die CSS-Eigenschaft «background-position »werden dann die entsprechenden Bildausschnitte dargestellt.

Abbildung 3.16: Das Ergebnis im Browser

Zusammenfassung

Die performance-optimierte Bereitstellung von Bildern im Responsive Webdesign ist die Basis für eine qualitative moderne Website. Die vorgestellten Ansätze haben alle ihre Vor- und Nachteile und bisher gibt es keine allgemeingültige beste Option. Je nach Ausgangspunkt kann die Entscheidung für Adaptive Images oder das srcset-Attribut die bessere Lösung für die Optimierung der Bilder sein. Der Einsatz von CSS-Sprites wird mittlerweile teilweise durch die Kodierung mit Data-URIs in Base64[3] ersetzt. Die massive Erhöhung des CSS-Quellcodes führt allerdings nur selten zu einer wirklichen Verbesserung der Ladezeitengeschwindigkeit, weshalb dieser Ansatz hier nicht weiter thematisiert werden soll. Einen ausführlichen Vergleich beider Ansätze liefert Peter McLachIan in seinem Artikel «CSS Sprites vs. Data URIs: Which is Faster on Mobile?»(McL13). Die Performance-Optimierung von Bildern in reaktionsfähigen Websites steht in direktem Zusammenhang mit den anderen Problemen in der Bereitstellung von Bildern. Im folgenden Abschnitt wird daher die Problematik des Bildausschnittes ausführlich thematisiert.

3 Base64 ist ein Verfahren zur Kodierung von 8-Bit-Binärdaten

3.3.2 Die Bildaussage

Im Webdesign werden große Header-Grafiken oder Hintergrundbilder zur Vermittlung von Botschaften oder bestimmter Stimmungen genutzt. Diese Großformat-Bilder verlieren mit Skalierung im Responsive Webdesign ihre Wirkung. Diese Problematik sollen mit gezielten Bildausschnitten für unterschiedliche Gerätegrößen gelöst werden.

Picture-Element

Das picture-Element orientiert sich an den HTML-5-Elementen zur Einbindung von Video-und Audiodateien und ist ein Umsetzungsvorschlag der Responsive Images Community Group(Gro). Es benutzt das srcset-Attribut für die Bereitstellung angepasster Bilder und kombiniert diese mit der Definition von Mediaqueries. So können angepasste Versionen der Bilder für spezielle Breakpoints definiert werden.

Definition des picture-Elements

Das Picture-Element wird als umfassender Container für die Einbindung des Bildes genutzt. Mit Hilfe des Attributes «source media»wird die Quelle des Bildes für den aktuellen Breakpoint angegeben. Die Angabe einer Klasse ist dabei optional. Im Screenshot ist die Veränderung des Bildpfades, je nach Veränderung des Breakpoints, zu sehen. Als Fallback-Lösung für ältere Browser wird am Ende die eigentliche Bildquelle ohne Breakpoint angegeben.

```
<picture>
  <source media="(min-width: 70em)"
    srcset="http://beispiel.de/1120/560"
    class="flexible">
  <source media="(min-width: 55em)"
    srcset="http://beispiel.de/880/440"
    class="flexible">
  <source media="(min-width: 40em)"
    srcset="http://beispiel.de/640/320"
    class="flexible">
  <source
    srcset="http://beispiel.de/360/180">
  <img src="http://beispiel.de/400x200&text=Fallback"
       class="flexible"
       alt="Beschreibungstext">
</picture>
```

Abbildung 3.17: Implementation des picture-Elements

Vorteile:

Der Einsatz der bekannten Syntax von Mediaqueries und die Kombination mit dem src-Attribut sorgen für ein leichtes Verständnis und einen schnellen Einsatz des picture-Elements. Die Definition von Minimal- und Maximalbreiten zur Bereitstellung von Bildern ist der reinen Einbindung mit Hilfe des src-Attributes vorzuziehen, da Bilder besser für unterschiedliche Displaygrößen bereitgestellt werden können. Außerdem bedient sich dieser Ansatz der bereites bekannten HTML5-Syntax zur Einbindung von Multimediadateien.

Nachteile:

Die umfangreiche Syntax macht den schnellen Umstieg auf diesen Lösungsansatz unmöglich. Die bis heute durchschnittliche Browserkompatibilität sorgt für die Notwendigkeit eines javascript-basierten Polyfills[4] (vgl. z. B.(Jeh14)). Insgesamt betrachtet bietet diese Lösung großes Potenzial für die Aufbereitung und Bereitstellung von Bildern im Kontext reaktionsfähiger Websites, aber die fehlende Kompatibilität und die umfangreiche Syntax sorgen für eine langsame Verbreitung dieses Ansatzes.

4 Polyfills bezeichnen JavaScript basierte Fallback-Lösungen.

Fokales Cropping mit FokusPoint

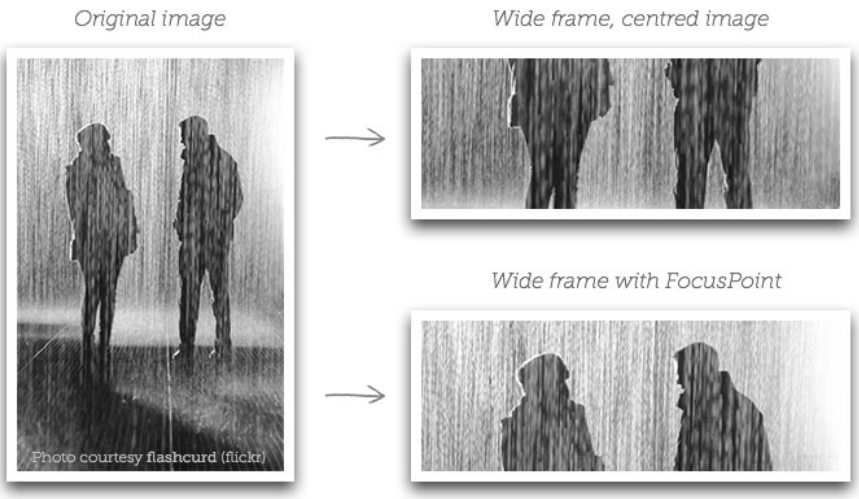

Abbildung 3.18: Fokales Cropping mit FokusPoint

Das Fokale Cropping beschreibt eine Technologie, bei der festgelegte Elemente des Bildes als repräsentativer Ersatz des Gesamtbildes für kleinere Bildschirmgrößen bereitgestellt werden. Eine praktische Anwendung ist das jQuery-Plugin «FocalPoint», welches mit Hilfe eines Koordinatensystems die Angabe repräsentativer Bildausschnitte ermöglicht.

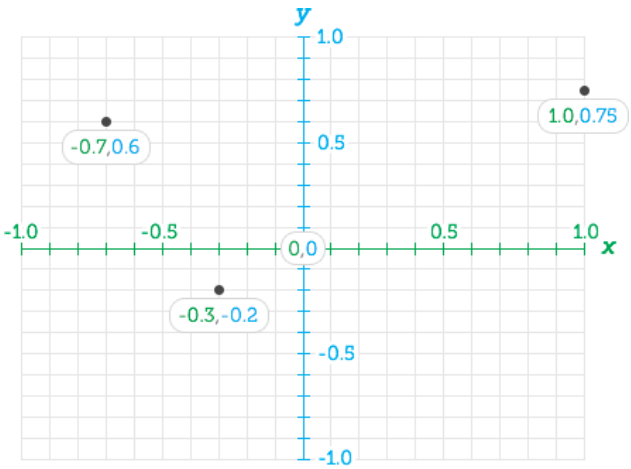

Abbildung 3.19: Koordinatensystem von FokusPoint

Das Plugin berechnet auf Grundlage der Gesamtgröße des Bildes ein Koordinatensystem und ermöglicht die Auswahl der entsprechenden Punkte auf der X- und Y-Achse. Die Koordinaten werden innerhalb von data-Attributen direkt im HTML-Markup angegeben (vgl. dazu auch (Fokb)).

```
1  <div class="focuspoint"
2  data-focus-x="0.331"
3  data-focus-y="-0.224"
4  data-image-w="400"
5  data-image-h="300">
6      <img src="image.jpg" alt="" />
7  </div>
```

Abbildung 3.20: Einbindung von FokusPoint

Die Klasse «focuspoint»wird dann mit Hilfe von jQuery innerhalb des HTML-Dokumentes selektiert und die Methode «.focusPoint()»angewendet.

```
1  <script>
2  $('.focuspoint').focusPoint();
3  </script>
```

Abbildung 3.21: Einbindung von FokusPoint

Ein praktisches Beispiel liefert die Demo von FocusPoint (Foka).

3.3.3 Hochauflösende Bilder

Die kontinuierliche Weiterentwicklung von mobilen Endgeräten brachte die Einführung hochauflösender Displays, wie das Retina-Display von Apple, mit sich. Bei hochauflösenden Bildschirmen wird die Anzahl der Pixel pro Zoll um das 1,5-fache erhöht. Das menschliche Auge ist nicht mehr in der Lage Pixelunterschiede zu erkennen. Der Einbau von hochauflösenden Displays ist allerdings den hochpreisigen Segmenten des Marktes vorenthalten, weshalb hier nicht von einem Standard für alle Geräte gesprochen werden kann. Das Problem ist die Bereitstellung von Bildern in unterschiedlichen Auflösungen und die Abfrage der Geräteeigenschaften durch den Browser.

SVG

Das SVG (Scalable Vector Graphics)-Format ist das einzige vektorbasierte Bildformat für den Einsatz im Web. Im Gegensatz zu pixelbasierten Grafiken ergibt sich durch die Skalierung von SVG-Grafiken kein Qualitätsverlust. Mit Ausnahme des Internet Explorers (Unterstützung ab Version 10) ist eine flächendeckende Browserkompatibilität gewährleistet. Einen anschaulichen Qualitätsvergleich liefert Ashish Bogawat in seinem Artikel im Smashing Magazine (Bog13).

Abbildung 3.22: Vergleich von Vektor-und Pixelgrafiken

In diesem Bild wird der Qualitätsunterschied deutlich, im vektorbasiertem Format sind die Konturen scharf und deutlich zu erkennen, während sich beim pixelbasiertem Format unscharfe Konturen bilden. Dieses Phänomen lässt sich vor allem bei der Vergrößerung über die physische Größe hinaus erkennen.

Vorteile:

Der Einsatz des SVG-Formats löst das Problem der gesonderten Bereitstellung von hochauflösenden Bildern. Durch die verlustfreie Skalierung ist es möglich, ein Bild für unterschiedliche Auflösungsformate zu nutzen.

Nachteile:

Die fehlende Browserkompatibilität für den Internet Explorer bringt die Notwendigkeit einer Fallbacklösung, z. B. mit einer alternativen PNG-Grafik, mit sich. Außerdem ist das SVG-Format nur für Darstellungen von einfachen Konturen und Elementen gedacht. Bei Einsatz für z. B. Fotos oder Zeichnungen lässt sich ein exponentieller Anstieg der Dateigröße erkennen, der den Einsatz dieses Formates nicht mehr erlaubt. Der Umgang mit SVG-Grafiken im Responsive Webdesign ist ein sehr umfangreiches Thema, weshalb im Rahmen dieser Arbeit keine ausführlichere Darstellung angestrebt wird.

Iconfonts

Die Symbolschriftart „Windings“ aus dem Jahre 1997 ist wohl die erste Anwendung von Iconfonts. Diese klassische Systemschriftart konvertiert Buchstaben in Symbole, wobei jeder Buchstabe für ein anderes Symbol steht. Schriften haben im Responsive Webdesign den Vorteil verlustfrei skalierbar und über CSS-Anweisungen anpassbar zu sein (vgl. dazu (Web12)). Diese Fähigkeiten führen zu einem großflächigen Einsatz dieser Technik für die Darstellung von Icons, Symbolen und anderen Elementen. Eines der bekanntesten Icon-Fonts ist Font Awesome (Awe). Dieses Icon-Set enthält 479 unterschiedliche Icons für die Verwendung auf Websites.

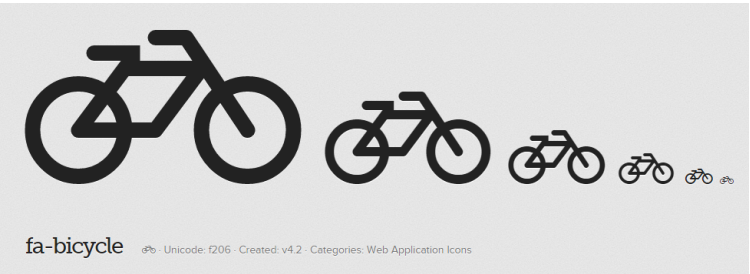

Abbildung 3.23: Ausschnitt des Iconsets von FontAwesome

Abbildung 3.23 zeigt die verlustfreie Skalierung von Iconfonts am Beispiel des Bicycle-Icons aus dem Font Awesome Icon-Set.

Vorteile:

Die verlustfreie Skalierung und die Anpassung mit Hilfe von CSS-Anweisungen machen den Einsatz von Icontfonts äußerst flexibel. Die Einbindung als Schrift hat außerdem einen deutlichen Performancevorteil gegenüber klassischen Bildern. Iconfonts erzeugen nur einen weiteren HTTP-Request durch die Einbindung der Schrift, im Gegensatz zu z. B. vereinzelten Bildern für die Darstellung von Icons.

Nachteile:

Der Einsatz von Iconfonts eignet sich nur für Elemente die keine semantische Funktion im Quellcode haben. Im Gegensatz zu Bildern verfügen Iconfonts nicht über die Angabe eines Titels oder «ALT»-Attributs und sind für die semantische Interpretation durch Suchmaschinen oder Screenreader ungeeignet. Dem Vorteil der Einsparung von HTTP-Requests steht die erhöhte Dauer der Bereitstellung und des Renderings der Schrift durch den Browser gegenüber. Der Performance-Gewinn ist nur bei umfangreicher Nutzung des Icon-Sets, im Gegensatz zu Einzelgrafiken oder CSS-Sprites, zu erreichen.

Zusammenfassung:

Die Bereitstellung von hochauflösenden Bildern für entsprechende Endgeräte ist immer noch ein Problem. Die fehlende Browserkompatibilität von SVG und der Verzicht auf Semantik bei Iconfonts sorgen nur für eine unzureichende Lösung der Bereitstellung. Außerdem eignen sich beide Formate nicht für die Darstellung komplexer Grafiken wie z. B. Fotos, Zeichnungen oder anderer vielschichtiger Illustrationen.

3.4 Navigationstypen

Die Wahl des Navigationstypen steht im direkten Zusammenhang mit der Inhaltsplanung der Website. Im Responsive Webdesign muss eine Navigation auf eine Vielzahl von unterschiedlichen Displaygrößen reagieren können. Mittlerweile haben sich einige Navigationslösungen für den Einsatz in reaktionsfähigen Websites etabliert, von denen hier exemplarisch vier vorgestellt werden sollen.

3.4.1 Top-Navigation

Abbildung 3.24: Beispiel für eine Top-Navigation

Die Top-Navigation ist die erste und einfachste Möglichkeit der Anpassung einer Navigation im Responsive Webdesign. Im Bild ist die Website des kanadischen Webdesigners Sean Halpin zu sehen (Hal14). Bei diesem Navigationstyp werden die Abstände der Navigationspunkte so lange verkleinert, bis sie auch auf kleineren Endgeräten darstellbar sind.

Vorteile:

Dieser Navigationstyp erfordert nur sehr geringe Anpassungen am bestehenden Quellcode. Er benötigt keinen Einsatz von Javascript oder umfangreichen CSS-Anpassungen (vgl. dazu (Zil13e)).

Nachteile:

Dieser Navigationstyp funktioniert nur bei einer geringen Menge an Navigationspunkten, die Anwendung eines Drop-Down-Menüs z. B. ist nicht möglich. Dieser Navigationstyp wird meistens nur auf Portfolio-Websites der Kreativbranche eingesetzt.

3.4.2 Footer-Navigation

Abbildung 3.25: Beispiel für eine Footer-Navigation

Ein mittlerweile ebenfalls nicht mehr gängiger Navigationstyp ist die Footer-Navigation. Hierbei werden alle Navigationselemente in den Footerbereich der Website verlagert. Im Kopfbereich der Website wird nur ein Verweis auf die Navigation gesetzt. Dies kann in Form eines Links oder Buttons geschehen, der mit einer Sprungmarke belegt ist.

Vorteile:

Dieser Navigationstyp bietet alle Vorteile der Top-Navigation, ist allerdings unendlich erweiterbar. Die Platzierung der Navigationselemente im Footer-Bereich der Website ermöglicht die Platzierung einer umfangreichen Navigation. Auch der Einsatz von komplexeren Navigationen, z. B. mit Dropdown-Funktionalität, ist möglich.

Nachteile:

Der wesentliche Nachteil ist die im ersten Moment fehlende Navigationsmöglichkeit für den User und die untypische Reaktion des Scrollens der Website. Aus Sicht der Usability wäre eine Verlangsamung der Scroll-Animation für ein besseres Verstehen des Prozesses notwendig. Das führt allerdings zu einer erhöhten Komplexität bei der Umsetzung dieser Methode und sorgt für die Notwendigkeit von JavaScript.

3.4.3 Toggle-Navigation

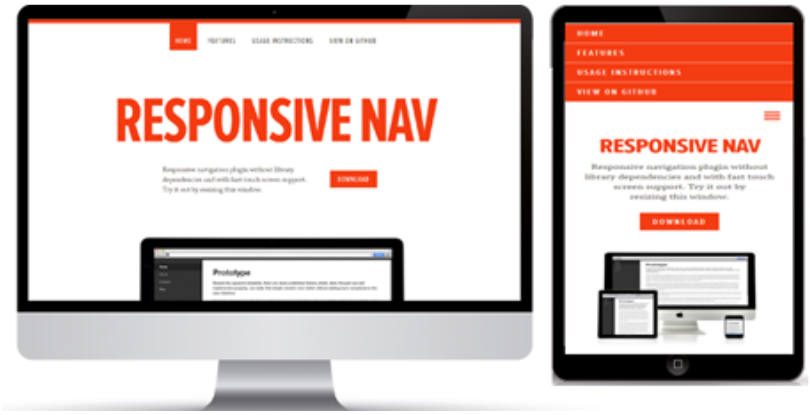

Abbildung 3.26: Beispiel für eine Toggle-Navigation

Die Toggle-Navigation gilt als die gängigste Lösung zur Umsetzung von Navigationen im Responsive Webdesign. Auf großen Displaygrößen wie z. B. Desktops wird auf eine klassische Top-Navigation gesetzt. Ist eine bestimmte Viewport-Breite erreicht, werden die Navigationselemente hinter einem Icon oder Button versteckt und bei Interaktion untereinander aufgelistet. Im Zuge der steigenden Entwicklung von Web-Applikationen ist allerdings auch eine vollständige Toggle-Navigation häufiger anzutreffen.

Vorteile:

Die Toggle-Navigation ermöglicht eine robuste Herangehensweise der Umsetzung von reaktionsfähigen Navigationen. Der Navigationstyp lässt sich unbegrenzt erweitern und durch komplexere Navigationselemente anreichern. Die Umsetzung ist mit Hilfe von Javascript oder CSS3 möglich und kann so performance-orientiert eingesetzt werden.

Nachteile:

Der einzige Nachteil dieser Lösung besteht im Einsatz eines Buttons. Obwohl sich der «Hamburger Button»[5] mittlerweile als effiziente Darstellung für die Navigation erwiesen hat, kann ein Verständnis nicht bei jedem User vorausgesetzt werden. Eine ausführlichere Erläuterung dieses Problems liefert der Artikel «HAMBURGER-MENÜ – JA ODER NEIN?»von Creative Construction (HER14).

5 Der Hamburger Button erhielt seinen Namen durch den Vergleich mit einem Hamburger (Nahrungsmittel)

3.4.4 Off-Canvas-Navigation

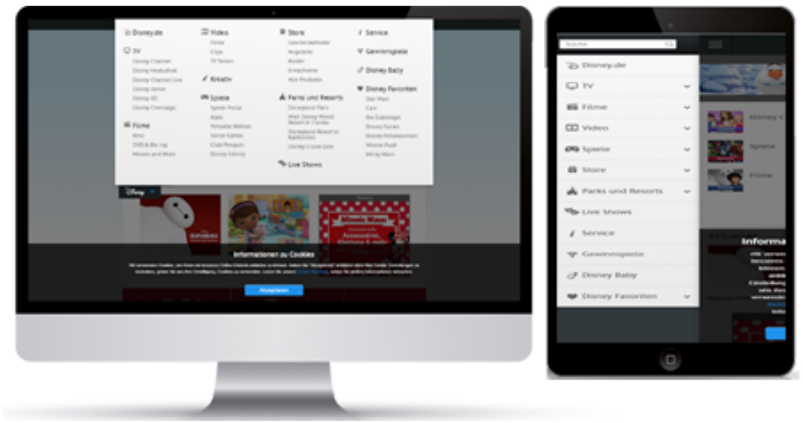

Abbildung 3.27: Beispiel für eine Off-Canvas-Navigation

Die Off-Canvas-Navigation nutzt eine negative Positionierung der Elemente zur Platzierung der Navigationselemente. Hierbei werden alle Navigationselemente absolut z. B. mit einem Wert von -9999 positioniert. Die Interaktion mit dem Button verschiebt die Position der Navigationselemente in den sichtbaren Bereich der Website, wobei ein Teil der eigentlichen Website sichtbar bleibt. Eine gute Illustration liefert der Interaktive Designer Jason Weaver auf seiner Website (Wea14).

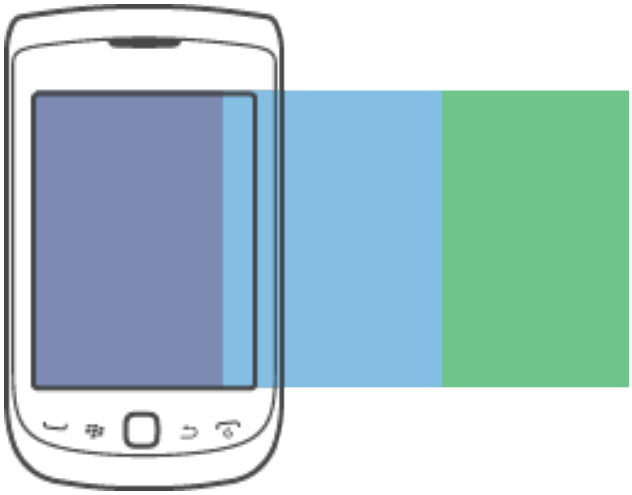

Abbildung 3.28: Der Off-Canvas-Bereich

Der magenta-farbene Bereich kennzeichnet den Bereich der Navigation nach der Interaktion. Ein Teil des eigentlichen Inhaltes bleibt sichtbar, dieser wird hier mit der hellblauen Farbe gekennzeichnet. Der grüne Bereich steht stellvertretend für den bisherigen Sidebar-Inhalt der Website und wird vollständig außerhalb des Sichtbereiches positioniert.

Vorteile:

Dieser Navigationstyp bietet uneingeschränkte Möglichkeiten der Navigations-Struktur. Der Einsatz von modernsten Technologien macht ihn zum animationsreichsten Navigationstyp.

Nachteile:

Die aufwendigen Animationseffekte die bei Off-Canvas-Navigationen eingesetzt werden, führen gerade bei älteren Geräten zu Performance-Problemen. Die Unterstützung von Javascript ist absolut notwendig, so dass eine fundierte Fallback-Lösung bedacht werden muss.

Zusammenfassung

Die Möglichkeiten in der Umsetzung von responsiven Navigationen sind umfangreich. In erster Linie entscheiden die inhaltliche Struktur und die gegebenen Anforderungen, welcher Navigationstyp am besten geeignet ist. Die Umsetzung von anspruchsvollen Toggle- oder Off-Canvas-Navigationen erfordert einen deutlichen Mehraufwand in der Entwicklung, welcher zwingende Berücksichtigung bei der Kalkulation von Webprojekten finden sollte. Mit Ausnahme der Top-Navigation sind aktuell, bei allen Navigationstypen, entsprechende JavaScript- und Fallbacklösungen notwendig um eine browser- und gerätekompatible Darstellung zu ermöglichen.

3.5 Responsive Typografie

Die Typographie einer Website hat großen Einfluss auf die Gesamterscheinung. Sie beeinflusst die wahrgenommene Größe von allen Elementen und ist durch ihre unterschiedlichen Eigenschaften wie z. B. Schriftstil, Kontrast und Zeilenabstand ein wichtiges Kriterium für eine gute Usability. Im Kontext des Responsive Webdesigns müssen diese Eigenschaften auf unterschiedliche Geräteeigenschaften angepasst werden. Die wesentlichen Elemente der Typographie und deren Einsatz in reaktionsfähigen Websites sind Bestandteil dieses Unterkapitels.

3.5.1 Schriftgrößen

Die bisher klassischen Einheiten für den Einsatz von Schriften waren «px»(Pixel) und «em». Pixel ist eine starre Einheit, die nicht auf Veränderungen des Kontextes reagiert. Aus diesem Grund war es die einfachste und beliebteste Anwendung für Größen von Schriften. Im Kontext des Responsive Webdesigns bestand die Notwendigkeit von anpassungsfähigen Schriften, weshalb eine zunehmende Nutzung von em zu erkennen ist. Einen wichtigen Aspekt für die richtige Schriftgröße liefert Zillgens in seinem Buch (Zil13f). Er bezieht die Betrachtungsentfernung in die Anpassung der Schriftgröße mit ein, dass bedeutet, die Schrift wird z. B. auf einem Smartphone kleiner skaliert, da ein geringerer Betrachtungsabstand herrscht.

3.5.2 typografische Tonleiter

Das Verhältnis der Schriftgrößen zueinander ist ein wichtiges Kriterium für eine optimale Lesbarkeit und Struktur des Inhaltes. Die typografische Tonleiter ist eine seit Jahrhunderten genutzte Skala zur harmonischen Skalierung von Schriftgrößen. Das Prinzip folgt den gleichen Gesetzen wie die einer musikalischen Tonleiter (Zil13f).

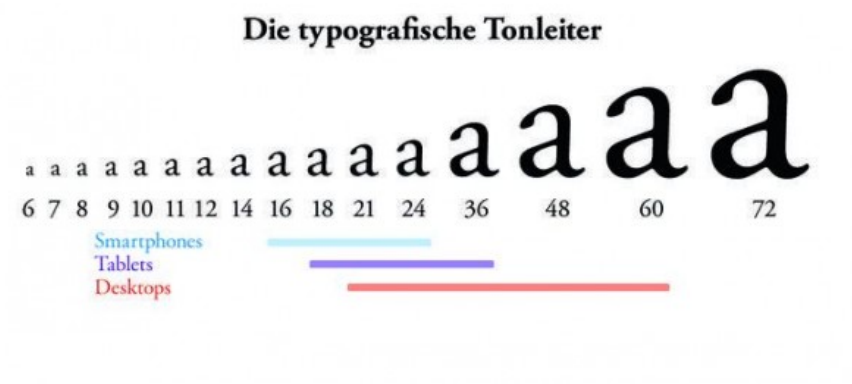

Abbildung 3.29: Die typographische Tonleiter

Abbildung 3.29 zeigt den Aufbau einer klassischen typographischen Tonleiter. Diese wurde mit entsprechenden Anmerkungen zu den Bereichen der klassischen Bildschirmgrößen im Webdesign ergänzt und liefert einen guten Überblick der Verteilung von Schriftgrößen im Responsive Webdesign. Allerdings sind die folgenden Angaben pixelbasiert und müssten entsprechend in ein anpassungsfähiges Format wie z. B. «em»umgerechnet werden.

3.5.3 Relative Einheiten für Schriften

Anpassungsfähige Schriften werden mit Prozent oder «em»angegeben. Diese Einheiten ermöglichen uns eine automatische Anpassung der Größe an unterschiedliche Gerätegrößen. Allerdings sind beide Einheiten nicht optimal für den Einsatz von Schriften im Responsive Webdesign geeignet. Stattdessen haben sich im Laufe der Zeit einige neue Einheiten für den Einsatz etabliert, welche im Verlauf dieses Kapitels genauer besprochen werden sollen.

Das Problem mit em im Kontext von Responsive Webdesign

Die Einheit «em»passt die Schriftgröße in Bezug zum Elternelement an. Hier liegt auch das eigentliche Problem dieser Einheit, der Bezug zum Elternelement.

```
1  body {
2      font-size:1.5em;
3  }
4  .container{
5      font-size:2em
6  }
7  .container .untercontainer {
8      font-size:1.5em
9  }
```

Abbildung 3.30: Abhängigkeit von em

Das Hauptelement «body»besitzt eine Schriftgröße von 1,5 em. Das «container»-Element liegt direkt innerhalb des «body»-Elements und erhält eine Schriftgröße von 2 em. Innerhalb des «container»-Elements gibt es nun das Element Untercontainer mit einer Schriftgröße von 1 em. Durch die Abhängigkeit vom Elternelement variieren die Schriftgrößen entsprechend. Eine ausführliche Thematisierung dieser Problematik findet sich im Buch «Responsive Typography »von Jason Pamental (Pam12).

Um das Ergebnis zu verdeutlichen wurde eine kurze HTML-Datei wie folgt angelegt:

```
1  <style>
2  body {
3      font-size:1.5em;
4  }
5  .container{
6      font-size:2em
7  }
8  .container .untercontainer {
9      font-size:1.5em
10 }
11 </style>
12 <body>
13     <p>Hier steht ein Beispieltext.</p>
14     <div class="container">
15         <p>Hier steht ein Beispieltext.</p>
16         <div class="untercontainer">
17             <p>Hier steht ein Beispieltext.</p>
18         </div>
19 </div>
```

Abbildung 3.31: Abhängigkeit von em

Das Ergebnis im Browser verdeutlicht das Problem der variierenden Schriftgrößen. Das Element «body»und das Element «untercontainer»haben die gleiche Schriftgröße, allerdings variiert die Darstellung aufgrund des Elternelements Container mit einer abweichenden Schriftgröße.

Das Ergebnis im Browser zeigt sich wie folgt:

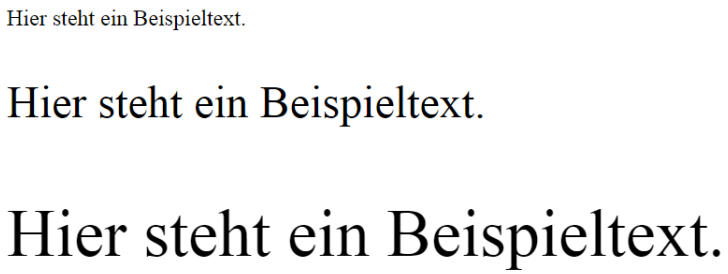

Abbildung 3.32: Abhängigkeit von em

Die Einheit rem

«Rem»bedeutet Root-em. Im Gegensatz zur Einheit «em», bezieht sich «rem»immer auf das Wurzelelement zur Skalierung der Schriftgröße. Wenn also im Body-Element eine Schriftgröße von 1.5 em angegeben worden ist, stellt diese Angabe den Bezug für alle folgenden «rem»-Angaben.

Zur Veranschaulichung wurde das em-Beispiel mit Hilfe der neuen Einheit «rem»angepasst:

```
1  <style>
2  body {
3      font-size:1.5rem;
4  }
5  .container{
6      font-size:2rem
7  }
8  .container .untercontainer {
9      font-size:1.5rem
10 }
11 </style>
12 <body>
13     <p>Hier steht ein Beispieltext.</p>
14     <div class="container">
15         <p>Hier steht ein Beispieltext.</p>
16         <div class="untercontainer">
17             <p>Hier steht ein Beispieltext.</p>
18         </div>
19 </div>
```

Abbildung 3.33: Anpassung nach rem

Die Ausgabe im Browser:

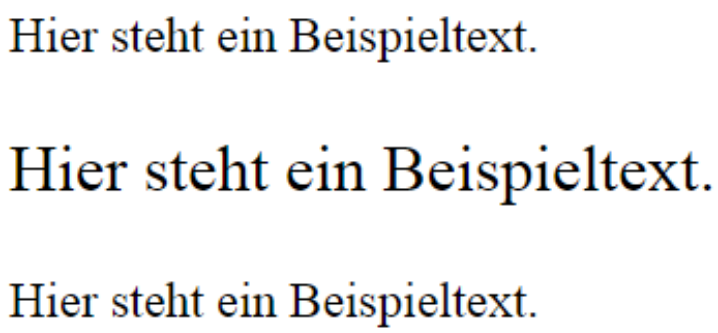

Abbildung 3.34: Anpassung nach rem

Die Gestaltung des Schriftbildes mit Hilfe der Einheit rem gestaltet sich deutlich einfacher durch den Bezug zum Wurzelelement einer Website. Der Einsatz von starren Größenangaben wie z. B. Pixel ist nicht notwendig. Eine weitere Eigenschaft für die Verwendung von relativen Angaben der Typografie ist die Einheit «vw», dieses wird allerdings aufgrund der unzureichenden Browserunterstützung nicht näher in dieser Arbeit betrachtet.

3.5.4 Zeilenlänge

Die optimale Zeilenlänge ist maßgeblich für die Lesefreundlichkeit der Inhalte einer Website verantwortlich. In statischen Websites war es möglich, Texte exakt auf eine optimale Zeilenbreite zu optimieren. Im Kontext des Responsive Webdesigns stellt uns dieses Kriterium vor neue Herausforderungen. Robert Bringhurst beschreibt in seinem Buch «The Elements of Typografic Style»(Bri13) eine optimale Zeichenanzahl von 45 - 75 Zeichen pro Zeile.

Mit Hilfe von Mediaqueries haben wir die Möglichkeit, die Zeilenlängen für unterschiedliche Displaygrößen zu verändern und somit eine optimale Lesbarkeit zu ermöglichen. Trend Walton beschreibt in seinem Artikel «Fluid Type“»(Wal12) eine Methode, welche die Optimierung der Zeilenlängen mit Hilfe von Breakpoints vereinfacht. Er positioniert Sternchen nach dem 45. und dem 75. Zeichen einer etwa 100 Zeichen langen Zeile. Die Zeichenlänge ist zu lang, falls beide Sternchen in die neue Zeile umbrechen, zu kurz wenn beide Sternchen in der gleichen Zeile bleiben und optimal, wenn nur das erste Sternchen in der Zeile bleibt und das zweite Sternchen umbricht.

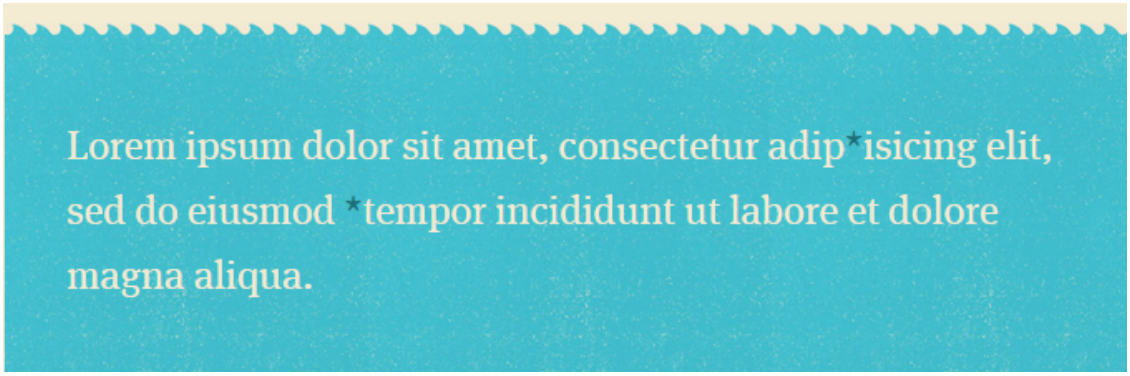

Abbildung 3.35: Optimale Zeilenlänge nach T.Walton

Zusammenfassung

Der richtige Einsatz von Schriften im Responsive Webdesign ist durch den Einsatz von neuen Einheiten vereinfacht worden. Dennoch befinden sich ein Großteil dieser Einheiten noch in der genauen Ausarbeitung, oder es fehlt eine entsprechende Browserunterstützung. Der Einsatz von Webfonts, die automatische Aufteilung in unterschiedliche Spalten, oder die Verwendung von jQuery zur Optimierung von Überschriften sind weitere Aspekte der Typographie im Kontext des Responsive Webdesigns, die allerdings nicht Bestandteil der Arbeit sind.

4 Fazit

4.1 Fazit

Ziel dieser Arbeit war eine qualitative Einschätzung der Methoden, Konzepte und Lösungsansätze für den flächendeckenden Einsatz der neuen Möglichkeiten durch Responsive Webdesign. Die Thematisierung der Workflow-Modelle sorgte bereits für erste Bedenken für eine einfache Adaptierung der neuen Technologien. Der Aufbau einer Website erfordert ein Umdenken auf konzeptioneller und technischer Sicht. Die Entwicklung von Websites nach den Workflow-Modellen Progressive Enhancement oder Mobil First steht im Gegensatz zum klassischen Webdesign-Workflow und macht einen iterativen Erstellungsprozess notwendig.

Auch wenn die meisten vorgestellten Elemente einer Website über Lösungskonzepte für die Umsetzung im Sinne des Responsive Webdesigns verfügen, sollte klar geworden sein, dass Einheitslösungen im Responsive Webdesign nur selten existieren. Eine genaue Anforderungsanalyse und ein test-getriebener Entwicklungsprozess sind notwendig geworden um gute Ergebnisse erzielen zu können.

Die fehlende Browserkompatibilität vieler Lösungsansätze macht die Berücksichtigung von Fallback-Lösungen notwendig und sollte als fester Bestandteil der professionellen Entwicklung von Websites gelten.

Die Entwicklung von neuen Endgeräten wie z. B. Smartwatches und Google Glass liefert neue Herausforderungen für die Umsetzung von reaktionsfähigen Websites. Abzuwarten bleibt auch die Reaktion des W3C in Bezug auf die Einbindung von Bildern im responsiven Kontext. Je nach Entscheidung könnten hier umfangreiche Anpassungen z. B. bei Content-Management-Systemen notwendig werden.

Es lohnt sich die Weiterentwicklung von Responsive Webdesign mit einem gewissen Interesse zu verfolgen, um auch in Zukunft den wachsenden Anforderungen gerecht werden zu können.

Literaturverzeichnis

[AO13] ARD/ZDF-ONLINESTUDIEN: Mobile Internetnutzung 2009 bis 2013. (2009-2013). http://www.ard-zdf-onlinestudie.de/index.php?id=426. – Website: http://www.ard-zdf-onlinestudie.de/index.php?id=426 Letzter Zugriff: 13.12.2014

[Awe] AWESOME, Font: Font Awesome Iconfonts. http://fortawesome.github.io/Font-Awesome/. – Website: http://fortawesome.github.io/Font-Awesome/ Letzter Zugriff: 6.1.2015

[Bog13] BOGAWAT, Ashish: Optimizing The Design Workflow With Fireworks Extensions, Part 3. In: *Smashing Magazin* (2013). http://www.smashingmagazine.com/2013/11/06/even-more-fireworks-extensions-optimized-design-workflow/. – Website: http://www.smashingmagazine.com/2013/11/06/even-more-fireworks-extensions-optimized-design-workflow/ Letzter Zugriff: 06.01.2015

[Bri13] BRINGHURST, Robert: *The Elements of Typographic Style: Version 4.0*. Hartley & Marks Publishers, 2013 http://webtypography.net/toc/

[Cla11] CLARKE, Andrew: I don't care about Responsive Web Design. (2011). http://stuffandnonsense.co.uk/blog/about/i_dont_care_about_responsive_web_design/. – Website: http://stuffandnonsense.co.uk/blog/about/i_dont_care_about_responsive_web_design Letzter Zugriff: 13.12.2014

[Foka] FOKUSPOINT: Beispiel Fokuspoint. http://jonom.github.io/jquery-focuspoint/demos/grid/lizard.html. – Website: http://jonom.github.io/jquery-focuspoint/demos/grid/lizard.html Letzter Zugriff: 28.12.2014

[Fokb] FOKUSPOINT: FokusPoint - jQuery-Plugin fr fokales Cropping. https://github.com/jonom/jquery-focuspoint. –

Website: https://github.com/jonom/jquery-focuspoint
Letzter Zugriff: 28.12.2014

[Gri] GRIDCALCULATOR: GridCalculator. , 2014. http://gridcalculator.dk/. –
Website: http://gridcalculator.dk/
Letzter Zugriff: 16.12.2014

[Gro] GROUP, Responsive Images C.: Responsive Images Community Group. http://responsiveimages.org/. –
Website: http://responsiveimages.org/
Letzter Zugriff: 28.12.2014

[Hal14] HALPIN, Sean: (2014). http://seanhalpin.io/. –
Website: http://seanhalpin.io/
Letzter Zugriff: 6.1.2015

[Hel13a] HELLWIG, Jonas: *Responsive Webdesign - Das umfassende Praxistraining*. Galilio Computing, 2013

[Hel13b] In: HELLWIG, Jonas: *Responsive Webdesign - Das umfassende Praxistraining*. Galilio Computing, 2013, S. 6.6

[Hel14a] HELLWIG, Jonas: Mobil First + Progressive Enhancement. (2014). http://blog.kulturbanause.de/2013/08/mobile-first-progressive-enhancement/. –
Website: http://blog.kulturbanause.de/2013/08/mobile-first-progressive-enhancement/
Letzter Zugriff: 13.12.2014

[Hel14b] HELLWIG, Jonas: Responsive Images <picture>, srcset, sizes & Co. (2014). http://blog.kulturbanause.de/2014/09/responsive-images-srcset-sizes-adaptive/. –
Website: http://blog.kulturbanause.de/2014/09/responsive-images-srcset-sizes-adaptive/l
Letzter Zugriff: 22.12.2014

[HER14] HEROES, CREATIVE C.: Hamburger-Menu Ja oder Nein? (2014). http://www.creativeconstruction.de/blog/hamburger-menue-ja-oder-nein/. –
Website: http://www.creativeconstruction.de/blog/hamburger-menue-ja-oder-nein/
Letzter Zugriff: 6.1.2015

[Ima14] IMAGES, Adaptive: Adaptive Images. (2014). http://adaptive-images.com/. –
Website: http://adaptive-images.com/
Letzter Zugriff: 22.12.2014

[Jeh14] JEHL, Scott: Picturefill - A responsive image polyfill. (2014). http://scottjehl.github.io/picturefill/. – Website: http://scottjehl.github.io/picturefill/ Letzter Zugriff: 6.1.2015

[Mar10] MARCOTTE, Ethan: Responsive Web Design. (2010). http://alistapart.com/article/responsive-web-design. – Website: http://alistapart.com/article/responsive-web-design Letzter Zugriff: 13.12.2014

[Mar11] MARCOTTE, Ethan ; BROWN, Mandy (Hrsg.): *Responsive Web Design*. Jeffrey Zeldman, 2011

[McG13] MCGRANE, Karen: Responsive Design Wont Fix Your Content Problem. (2013). http://alistapart.com/column/responsive-design-wont-fix-your-content-problem. – Website: http://alistapart.com/column/ responsive-design-wont-fix-your-content-problem Letzter Zugriff: 13.12.2014

[McL13] MCLACHIAN, Peter: CSS Sprites vs. Data URIs: Which is Faster on Mobile? (2013). http://www.mobify.com/blog/css-sprites-vs-data-uris-which-is-faster-on-mobile/. – Website: http://www.mobify.com/blog/ css-sprites-vs-data-uris-which-is-faster-on-mobile/ Letzter Zugriff: 6.1.2015

[Pam12] *Kapitel* Kapitel 6. In: PAMENTAL, Jason: *Responsive Typography*. O'Reilly, 2012

[Pet14] PETTIT, Nick: How I Rapidly Prototype Websites. (2014). http://blog.teamtreehouse.com/rapidly-prototype-websites. – Website: http://blog.teamtreehouse.com/rapidly-prototype-websites Letzter Zugriff: 10.1.2015

[sel14] SELFHTML.ORG: Abfragen durch Mediaqueries auf selfhtml.org. (2014). http://wiki.selfhtml.org/wiki/CSS/Media_Queries. – Website: http://wiki.selfhtml.org/wiki/CSS/Media_Queries Letzter Zugriff: 16.12.2014

[Smi] SMITH, Nathan: 960-Grid-System. http://960.gs/. – Website: http://960.gs/ Letzter Zugriff: 16.12.2014

[W3C14] W3C: Mediaqueries Spezifikation durch das W3C. (2014). http://www.w3.org/TR/css3-mediaqueries/. – Website: http://www.w3.org/TR/css3-mediaqueries/ Letzter Zugriff: 16.12.2014

[Wal12] WALTON, Trent: Fluid Type. (2012). http://trentwalton.com/2012/06/19/fluid-type/. – Website: http://trentwalton.com/2012/06/19/fluid-type/ Letzter Zugriff: 10.1.2015

[Wea14] WEAVER, Jason: A Multi-Device Web Layout Pattern. (2014). http://jasonweaver.name/lab/offcanvas. – Website: http://jasonweaver.name/lab/offcanvas Letzter Zugriff: 6.1.2015

[Web12] WEBKRAUTS: Visuelle Semantik neu entdecken. (2012). http://webkrauts.de/artikel/2012/icon-fonts-visuelle-semantik-neu-entdecken. – Website: http://webkrauts.de/artikel/2012/ icon-fonts-visuelle-semantik-neu-entdecken Letzter Zugriff: 28.12.2014

[Web14] WEBKRAUTS: Der neue Standard fr responsive Bilder. (2014). http://web-krauts.de/artikel/2014/der-neue-standard-fuer-responsive-bilder. – Website: http://web-krauts.de/artikel/2014/ der-neue-standard-fuer-responsive-bilder Letzter Zugriff: 22.12.2014

[Wro13a] In: WROBLEWSKI, Luke: *MOBIL FIRST*. Jeffrey Zeldman, 2013, S. 52

[Wro13b] In: WROBLEWSKI, Luke: *MOBIL FIRST*. Jeffrey Zeldman, 2013, S. 28–29

[Zil13a] In: ZILLGENS, Christoph: *Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen*. Hanser Verlag, 2013, S. 16–25

[Zil13b] In: ZILLGENS, Christoph: *Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen*. Hanser Verlag, 2013, S. 38

[Zil13c] In: ZILLGENS, Christoph: *Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen*. Hanser Verlag, 2013, S. 47–48

[Zil13d] In: ZILLGENS, Christoph: *Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen*. Hanser Verlag, 2013, S. 195–196

[Zil13e] In: ZILLGENS, Christoph: *Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen*. Hanser Verlag, 2013, S. 237

[Zil13f] In: ZILLGENS, Christoph: *Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen*. Hanser Verlag, 2013, S. 181

[ZU] ZURB-UNIVERSITY: Graceful Degradation - Adapting websites to less-than-ideal situations is a short-term strategy. http://zurb.com/word/graceful-degradation. – Website: http://zurb.com/word/graceful-degradation
Letzter Zugriff: 13.12.2014

4.2 Bildnachweise

2.1 Der bisherige Workflow im Webdesign

Quelle:http://viljamis.com/blog/2012/responsive-workflow/

2.2 Der responsive Workflow im Webdesign

Quelle:http://viljamis.com/blog/2012/responsive-workflow/

2.3 Graceful Degradation nach Jonas Hellwig

Quelle:http://blog.kulturbanause.de/2013/08/mobile-first-progressive-enhancement/

2.4 Progressive Enhancement nach Jonas Hellwig

Quelle:http://blog.kulturbanause.de/2013/08/mobile-first-progressive-enhancement/

2.5 Mobil First nach Jonas Hellwig

Quelle:http://blog.kulturbanause.de/2013/08/mobile-first-progressive-enhancement/

3.1 Das pixelbasierte 960-Grid

Quelle:http://960.gs/

3.2 Das relative 960-Grid Quelle:http://gridcalculator.dk/

3.3 Das Border-Box-Modell

Quelle:ZILLGENS, CHRISTOPH: Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen, Seite 42. Hanser Verlag, 2013.

3.4 Externe Einbindung von Mediaqueries

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.5 Inline-Einbindung von Mediaqueries

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.6 Viewport-Metaangabe

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.7 Viewport-Vergleich

Quelle: ZILLGENS, CHRISTOPH: Responsive Webdesign Reaktionsfaehige Websites gestalten und umsetzen, Seite 48. Hanser Verlag, 2013.

3.8 Einbindung von responsiven Bildern

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.9 Einbindung von responsiven Bildern mit height:auto

Quelle: https://creativemarket.com/lilsquid/5766-Lil-Landscapes (bearbeitet)

3.10 Einbindung von responsiven Bildern ohne height:auto

Quelle: https://creativemarket.com/lilsquid/5766-Lil-Landscapes (bearbeitet)

3.11 Adaptive Images

Quelle: Speckmaier, Florarian: Mobil First Responsive Webdesign - Eine Untersuchung zu Grundlagen, der gestalterischen und technischen Umsetzung sowie eine Analyse des Standes der Technik.2013. Hochschule der Medien Stuttgart.

3.12 Src-Set Visualisierung - Jonas Hellwig

Quelle:http://blog.kulturbanause.de/2014/09/responsive-images-srcset-sizes-adaptive/

3.13 Definition des src-Attributs

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.14 Die erstelle Sprite-Grafik in Photoshop

Quelle:HELLWIG, JONAS: Responsive Webdesign - Das umfassende Praxistraining, Seite 6.6. Galilio Computing,2013.

3.15 Der CSS-Code für die Positionierung der Elemente

Quelle: HELLWIG, JONAS: ResponsiveWebdesign - Das umfassende Praxistraining, Seite 6.6. Galilio Computing,2013.

3.16 Das Ergebnis im Browser

Quelle: HELLWIG, JONAS: ResponsiveWebdesign - Das umfassende Praxistraining, Seite 6.6. Galilio Computing,2013.

3.17 Implementation des picture-Elements

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.18 Fokales Cropping mit FokusPoint

Quelle: https://github.com/jonom/jquery-focuspoint

3.19 Koordinatensystem von FokusPoint

Quelle: https://github.com/jonom/jquery-focuspoint

3.20 Einbindung von FokusPoint

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.21 Einbindung von FokusPoint

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.22 Vergleich von Vektor-und Pixelgrafiken

Quelle: http://www.smashingmagazine.com/2013/11/06/even-more-fireworks-extensions-optimized-design-workflow/

3.23 Ausschnitt des Iconsets von FontAwesome

Quelle: http://fortawesome.github.io/Font-Awesome/

3.24 Einbindung von Iconfonts

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.25 Beispiel für eine Top-Navigation

Quelle: Screenshot von http://seanhalpin.io/

3.26 Beispiel für eine Footer-Navigation

Quelle: Screenshot von http://futurefriendlyweb.com/

3.27 Beispiel für eine Toggle-Navigation

Quelle: Screenshot von http://responsive-nav.com/

3.28 Beispiel für eine Off-Canvas-Navigation

Quelle: http://disney.de/

3.29 Der Off-Canvas-Bereich

Quelle: http://jasonweaver.name/lab/offcanvas

3.30 Die typographische Tonleiter

Quelle: http://t3n.de/magazin/responsive-webdesign-reaktionsfahige-typografie-230315/2/

3.31 Abhängigkeit von em

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.32 Abhängigkeit von em

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.33 Abhängigkeit von em

Quelle: Eigener Screenshot mit Google Chrome.

3.34 Anpassung nach rem

Quelle: Eigener Screenshot aus dem Open Source Editor Brackets.

3.35 Anpassung nach rem

Quelle: Eigener Screenshot mit Google Chrome.

3.36 Optimale Zeilenlänge nach T.Walton

Quelle: http://trentwalton.com/2012/06/19/fluid-type/

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift