HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN MÜNCHEN
FAKULTÄT FÜR INFORMATIK UND MATHEMATIK

SEMINARARBEIT

Arbeitstitel: Responsive Web Design - eine App für alle Devices

Sascha Siemens
Matrikel-Nr. 29014415

Fach:
Aktuelle Technologien zur Entwicklung verteilter Java-Anwendungen

Betreuer:
M. Theis

19. Mai 2017

Eigenständigkeitserklärung

Der Verfasser erklärt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Der Verfasser stimmt der Veröffentlichung des Dokuments auf der Webseite des Betreuers zu. Für die elektronische Veröffentlichung ist die Zustimmung auch ohne Unterschrift gültig.

München, 19. Mai 2017 Unterschrift:

Inhaltsverzeichnis

1 Einleitung

Seit der immer fortschreitenden Entwicklung sowie Nutzung von mobilen Geräten im privaten als auch im industriellen Umfeld, hat die Bedeutung von optimierten Webseiten für Geräte mit kleineren Auflösungen eine immer wichtiger werdende Rolle zugesprochen bekommen. Webseiten, welche für die Auflösung eines normalen PC's erstellt wurden, werden auf mobilen Geräten mit einer kleinen Displaygröße nur schwer leserlich dargestellt. Oft muss man mittels zwei Finger in den Bereichen der Website hineinzoomen, um die Inhalte lesen zu können. Darüber hinaus gestaltet sich die Navigation der Website äußerst umständlich, da mittels der gezoomten Ansicht innerhalb der Website zur Navigationsleiste gescrollt werden muss. Zusätzlich lassen sich Webformulare oder andere Eingaben nur schwer bzw. umständlich bedienen. Im privaten Bereich mag dies vorerst nur einen negativen Aspekt auf die Benutzerfreundlichkeit werfen. Im industriellen Umfeld hingegen, bei dem ein Mitarbeiter mit einem mobilen Gerät auf die Webanwendung angewiesen ist, stellen diese Umstände keine praktikable und somit akzeptable Lösung dar. Die Entwicklung einer eigenen mobilen Version der Website, welchen nur für Geräte mit geringer Auflösung geeignet ist, stellt ebenfalls keinen effizienten Weg dar, da Inhalte und Änderungen somit an zwei verschiedenen Stellen im HTML-Code durchgeführt werden müssten. In dieser Ausarbeitung wird der Ansatz des „responsive Web-Designs“ behandelt und beschreibt, wie Webauftritte sowohl für Desktop-PC's als auch für mobile Geräte komfortabel und einfach erstellt werden können und eine hohe Benutzerfreundlichkeit aufweisen. Dabei wird am Anfang des Dokuments auf die technischen grundlegenden Prinzipien eingegangen und anschließend auf die Umsetzung anhand des meist verbreiteten responsive Web-Framework's "Bootstrap"ausführlich dargestellt.

Hinweis: Dieses Dokument setzt grundlegende Kenntnisse in HTML5 und CSS3 voraus. Enthaltene Code-Beispiele werden nur hinsichtlich des responsive Web-Designs erläutert.

2 Einführung zum Responsive Web-Design

Als responsive Web-Design wird der Ansatz in der Web-Entwicklung beschrieben, bei dem anhand gerätespezifischen Bedingungen, wie Auflösung und Geräteausrichtung (Hoch- oder Querformat), eine optimale Darstellung der Webseite erreicht wird, ohne explizit verschiedene Versionen der Webseite anzufertigen. „Responsive“ in diesem Kontext bedeutet, dass die Webseite automatisch anhand der Auflösung des Endgeräts angepasst wird.

Zusammenfassend lässt sich responsive Web-Design mittels folgender Punkte beschreiben[1]:

- Die Webseite sollte einfach zu nutzen sein
- Die Darstellung sollte gut aussehen
- Unabhängig von Bildschirmgröße, Browser und Gerät ansprechend und automatisch skalieren

2.1 Merkmale und Beispiel eines responsive Webauftritts

Da mobile Geräte nicht die gleiche Menge an Daten auf dem Bildschirm anzeigen können, wie Laptops oder Desktop-Systeme, gibt es einige Charakteristika, welche bei mobilen Webseiten angewendet werden:

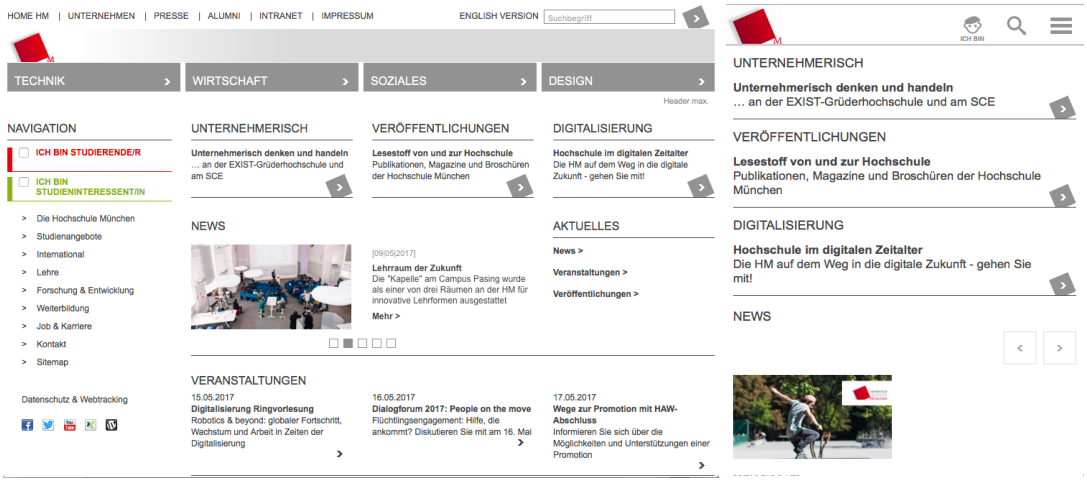
- Logos, Banner oder Header minimiert bzw. minimalisiert anzeigen (oder komplett ausblenden)
- Suchleisten, Login-Formulare oder andere statische Elemente durch Symbole ersetzen (z.B. Lupen-Symbol für Suchleiste oder Schlüssel-Symbol für Loginformular verwenden)
- Navigationsteil der Webseite durch aufklappbare Elemente ersetzen
- Beiträge oder Inhalte, welche auf der Desktop-Seite nebeneinander dargestellt werden, sollten untereinander angeordnet werden.
- Meta-Informationen ausblenden (z.B. Veröffentlichungsdatum von Beiträgen oder Namen von Autoren)

[1] [W3S17a]

- Beiläufige Angaben, welche nicht unmittelbar zur Information der jeweiligen Seite benötigt werden, sollten in der Regel ausgeblendet oder können durch dynamische Elemente aufgeklappt werden.

Die oben beschriebenen Techniken werden im Abschnitt 3 mit anschaulichen Beispielen unter Verwendung des responsive Web-Frameworks „Bootstrap“ verdeutlicht.

Abbildung 1 zeigt eine responsive Webseite der Hochschule München und stellt die Desktop-Version der mobilen Variante gegenüber. Dabei sind die oben genannten Charakteristika klar zu erkennen.



(a) Desktop-Version (b) Mobile Version

Abbildung 1: Demonstration der Charakteristika anhand des Webauftritts der Hochschule München

2.2 Technische Umsetzung des responsive Web-Design

Technisch wird ein ansprechender Webauftritt mittels den aktuellen Elementen von HTML5 und CSS3 realisiert. HTML5 und CSS3 gelten somit auch als Voraussetzung für ein responsive Web-Design. Beim Abrufen der Webseite muss dem Endgerät mitgeteilt werden, welche Standardskalierung (Viewport) verwendet werden soll. Außerdem muss festgelegt werden, welche Größen die statischen Elemente, wie zum Beispiel Bilder oder Tabellen, annehmen sollen (Stichwort: relative Größenangaben). Darüber hinaus werden mittels „Media Queries“ festgelegt, wie die Webseite im Querformat oder bei Benutzung einer bestimmten Auflösung auszusehen hat und wie dabei entsprechende Elemente im Browser angeordnet werden sollen. Soeben wurden einige Fachbegriffe verwendet, welche auf den ersten Blick nicht eindeutig erscheinen, daher werden diese Themen in den nächsten Abschnitten im Detail behandelt.

JavaScript ist für die Erstellung einer skalierbaren Webseite nicht notwendig und bleibt in diesem Kapitel unbeachtet . Es sei dennoch darauf hingewiesen, dass JavaScript zunehmend an Beliebtheit gewonnen hat, um Aktions-, Navigations- oder Übergangselemente fließend zu gestalten und somit die Benutzerfreundlichkeit zu erhöhen. Diese Thematik wird im Kapitel 3 bei der Benutzung des responsive Web-Frameworks „Bootstrap“ noch einmal aufgegriffen und im Kontext von Navigationsmenüs behandelt.

2.2.1 Die Viewport-Einstellung

Vor der Zeit der mobilen Endgeräte, wie Smartphones und Tablets, wurden die Webseiten standardmäßig für Desktop-Systeme entwickelt. Diese Webseiten hatten zumeist ein statisches Design und fixe Größenangaben von Bildern, Tabellen oder anderen Elementen. Wurden diese Webseiten von mobilen Endgeräten aufgerufen, so musste der Benutzer in die Anwendung zoomen, um diese benutzen zu können. Benutzer von Webapplikationen sind es in der Regel gewohnt, vertikal und nicht horizontal zu scrollen. Die Ansicht der Webseite horizontal zu bewegen gilt unter Web-Entwicklern als schlechter Stil und sollte, auch auf Geräten mit kleiner Auflösung, vermieden werden[2]. Zur Abhilfe des Problems wurde mit HTML5 die „Viewport“-Einstellung eingeführt, welche dem Webbrowser Informationen hinsichtlich der Skalierung sowie Dimensionierung übermittelt. Ist zum Beispiel eine Webanwendung auf eine Desktop-Breite von mindestens 1024 Pixeln ausgelegt, so kann diese von einem Gerät mit einer Auflösung von 320 x 480 Pixeln nicht vollständig angezeigt werden. Mittels dem Viewport-Tag kann somit dem Endgerät mitgeteilt werden, welche „virtuelle“ Auflösung von dem Gerät angezeigt werden soll. Das Viewport-Tag kann mittels folgendem HTML-Header definiert werden.

```
1 <meta name="viewport" content="width=1024">
```

In diesem Fall zeigt das mobile Endgerät trotz seiner niedrigeren Auflösung einen virtuellen Bereich von 1024 Pixel im Webbrowser an und der User kann den gesamten Bereich der Homepage anstelle des nur kleinen Ausschnitts der Bildschirmauflösung betrachten. In diesem Beispiel wird dem Benutzer die gesamte Webseite präsentiert, allerdings ist dadurch nicht das Problem gelöst, dass nun alle Elemente (insbesondere Texte) auf dem Bildschirm sehr klein erscheinen und nicht intuitiv lesbar sind. Mittels dieser „Lösung“ wurde nur sichergestellt, dass beim ersten Aufruf der Webseite der User den gesamten Webseiteninhalt präsentiert bekommt. Enthält eine Webseite kein Viewport-Tag im HTML-Header, so wird im mobilen Webbrowser ein Standardwert zur Anzeige der Webseite angenommen. Dieser Standardwert variiert von

[2] [W3S17c]

Gerät zu Gerät und ist meist vom Betriebssystem abhängig.

Hinsichtlich des responsive Web-Designs wird die Verwendung folgendes Viewport-Tags auf jeder Webseite empfohlen:

```
1 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

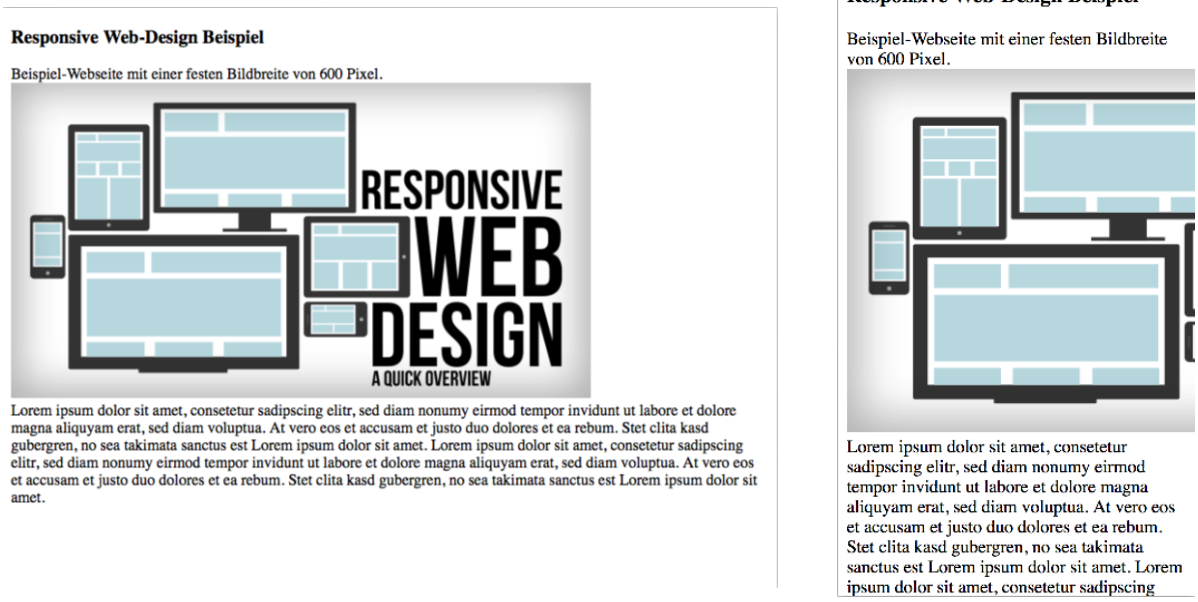
Dabei entspricht die Anzeigebreite die Pixel-Breite des jeweiligen aufrufenden Geräts (genauer: die Größe des Anzeigebereichs im Browser-Fenster) und ist somit nicht mehr fest definiert. Der *initial-scale*-Wert beschreibt dabei den anfänglichen Zoom-Faktor beim ersten Abruf der Webseite. Der Zusammenhang des empfohlenem Viewport-Tags hängt auch stark von der Benutzung der „Media Queries“ ab, welche im Kapitel „Einsatz von Media Queries“ genauer erläutert werden.

2.2.2 Einsatz von relativen Größenangaben

Neben der Verwendung des empfohlenem Viewport-Tags, spielt die Dimensionierung von Grafik-Elementen, Tabellen oder anderen größeren Objekten eine wichtige Rolle. Folgender HTML-Code zeigt ein Minimal-Beispiel mit einem Bild der festen Breite von 600 Pixel (siehe Zeile 11):

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta name="viewport" content="width=device-width, initial-scale=1.0">
5  <meta charset="UTF-8">
6  <title>Responsive Web-Design Beispiel</title>
7  </head>
8  <body>
9  <h3>Responsive Web-Design Beispiel</h3>
10 Beispiel-Webseite mit einer Bildbreite von 600 Pixeln.<br>
11 <img src="example_rwd.gif" alt="Example Picture" width="600px"/><br>
12 Lorem ipsum [... Text gekürzt ...] Lorem ipsum dolor sit amet.
13 </body>
14 </html>
```

Diese Webseite mag für die Anzeige auf einem Desktop-System kein Problem darstellen. Allerdings zeigt die Ansicht der gleichen Webseite von einem mobilen Gerät mit der Auflösung von 320 x 568 Pixel, dass das Bild nicht vollständig angezeigt wird.



Desktop-Version mit einer Auflösung von 800 x 600 Pixel

Mobile Version mit einer Auflösung von 320 x 568 Pixel

Abbildung 2: Anzeige einer festen Bildbreite von 600 Pixel

Abbildung 2 zeigt die Gegenüberstellung der beiden Ansichten, sowie die Problematik, dass bei der mobilen Version nur ein Teil des Bildes dargestellt wird. Es wird daher empfohlen, alle Elemente relativ zur Bildschirmgröße zu skalieren[3]. Im obigen Beispiel sollte die Breite des Bildes nicht in Pixel sondern prozentual zur Bildschirmgröße angegeben werden. Das Image-Tag in Zeile 11 des HTML-Beispiels wird deswegen wie folgt abgeändert:

```
1 <img src="example_rwd.gif" alt="Example Picture" width="100%"/><br>
```

Abbildung 3 zeigt die Darstellung der mobilen Ansicht mit der dynamischen Anpassung von 100 Prozent. Dieses Prinzip sollte auf allen Webseite eingehalten werden. Anstatt statische Pixelwerte sollten dynamische Angaben verwendet werden.

[3] [WDW12]

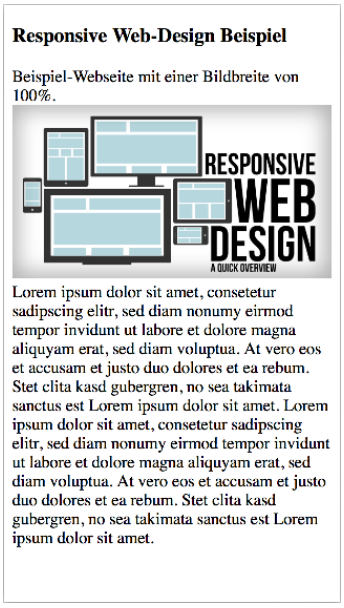


Abbildung 3: Dynamische Anzeige des Bildes

2.2.3 Einsatz von Media Queries

Mit der Einführung von CSS3 wurden auch die Media Queries im CSS-Standard eingeführt. Media Queries beschreiben dabei, wie einzelne HTML-Elemente bei verschiedenen Bildschirmauflösung darzustellen bzw. anzuordnen sind. In anderen Worten werden mittels Media Queries mehrere verschiedene CSS-Templates für einzelne Bildschirmauflösungen definiert. Voraussetzung dafür ist das bereits angesprochene Viewport-Tag im HTML-Header, um anhand der Auflösung des jeweiligen Geräts die richtige Darstellung zu garantieren.[4] Um diesen Sachverhalt zu verdeutlichen, wurde aus dem Beispiel des vorhergehenden Kapitels ein CSS3-Stylesheet in einer dedizierten Datei zugewiesen. (siehe Zeile 4). Anbei der Auszug des Headers aus der HTML-Datei:

```
1 <head>
2   <meta name="viewport" content="width=device-width, initial-scale=1.0">
3   <meta charset="UTF-8">
4   <link rel="stylesheet" href="style.css">
5   <title>Responsive Web-Design Beispiel</title>
6 </head>
```

Die dazugehörige CSS-Datei *style.css* soll folgendes Verhalten der verschiedenen Ansichten aufweisen:

[4] [W3S17b]

- Ist die Auflösungsbreite des Geräts/Browserfensters größer als 600 Pixel, so wird der Body-Hintergrund grün gefärbt, sowie alle Bilder mit einer Größe von 40 Prozent der Bildschirmbreite dargestellt
- Wird das Gerät im Querformat genutzt und hat das Gerät eine geringere Auflösungsbreite als 600 Pixel, so wird der Body-Hintegrund gelb gefärbt sowie alle Bilder ausgeblendet.
- Trifft keiner dieser Bedingungen zu, so wird das Standard-Template verwendet. (Blauer Hintergrund, Bilder werden zu 100 Prozent der Bildschirmbreite angezeigt)

Inhalt der CSS-Datei *style.css*:

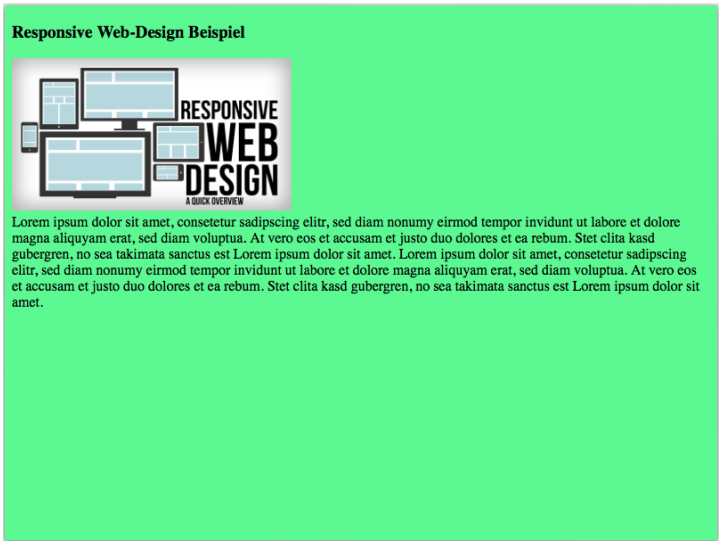
```
1  /* Standard CSS-Layout, sofern die unten definierten Bedingungen nicht
       zutreffen. */
2  body {
3  background-color: lightblue;
4  }
5
6  img {
7  width: 100%;
8  }
9
10 /* Dieses CSS-Layout wird angewendet, wenn die Auflösung der Breite
      größer als 600 Pixel ist.*/
11 @media screen and (min-width: 600px) {
12 body {
13 background-color: lightgreen;
14 }
15
16 img {
17 width: 40%;
18 }
19 }
20
21 /* Dieses CSS-Layout wird angewendet, wenn die Auflösung der Breite
      kleiner als 600 Pixel ist und das Gerät im Querformat genutzt wird*/
22 @media screen and (max-width: 600px) and (orientation: landscape){
23 body {
24 background-color: lightyellow;
25 }
26
27 img{
```

```
28  display: none;
29  }
30  }
```

Abbildung 4 zeigt die einzelnen Ansichten mit verschiedenen Auflösungen an.

(a) Mobile Ansicht mit einer Auflösung von 320 x 568 Pixel

(b) Mobile Ansicht im Querformat mit einer Auflösung von 568 x 320 Pixel



(c) Darstellung eines Desktop-Systems mit einer Auflösung von 800 x 600 Pixel

Abbildung 4: Darstellung der einzelnen Ansichten mit der *style.css*-Datei unter Verwendung von Media Queries

Hinweis: Dieses Beispiel soll nur anschaulich die Verwendung von Media Queries demonstrieren und hat keinen näheren Praxisbezug.

Aus Gründen der Vereinfachung, wurden in der oben dargestellten CSS-Datei nur *body*-Elemente sowie *img*-Elemente verändert. Diese Änderungen können darüber hinaus auch auf *class*- oder *id*-Objekte angewendet und alle in CSS üblichen Kombinationen benutzt sowie verschachtelt werden. Bei Webseiten mit sehr vielen Elementen und verschiedenen Anzeigebereichen kann die Gestaltung der CSS-Datei überaus komplex, sowie sehr lange und damit auch unübersichtlich werden. Es wird an dieser Stelle deswegen darauf hingewiesen, sich über das Design sowie über die Wahl der verschiedenen Auflösungen in den Media Queries im Vorhinein Gedanken zu machen, bevor weitere Details des CSS-Designs definiert werden. Abhilfe schaffen in diesem Fall „responsive Web-Frameworks“, wie zum Beispiel das weltweit am meisten genutzte *Bootstrap*-Framework, welches im Kapitel 3 im Detail vorgestellt wird.

3 Bootstrap - Ein responsive Web-Framework

In diesem Kapitel wird die Verwendung und der Einsatz des bekanntesten responsive Web-Frontend-Frameworks[5] namens „Bootstrap“ beschrieben. Bootstrap wurde ursprünglich von Twitter entwickelt und im Jahr 2011 als Open-Source-Produkt veröffentlicht und beinhaltet ein breites Spektrum an HTML-, CSS- sowie JavaScript-Templates unter Beachtung des responsive Web-Design Ansatzes. Im Sommer 2014 erfreute sich Bootstrap einer derart hohen Nachfrage, dass es auf GitHub zum beliebtesten Projekt wurde. URL zur Homepage: `www.getbootstrap.com`

3.1 Einführung in Bootstrap

Bei Bootstrap handelt es sich um ein umfangreiches HTML5, CSS3 und JavaScript Framework. Allerdings werden für die reine Erreichung eines ansprechenden Webauftritts („responsiveness“) nur die CSS-Dateien benötigt. Dabei empfiehlt Bootstrap jeweils die aktuellste Version der gängigsten Browser zu verwenden. Eine Auflistung der unterstützten Webbrowser kann auf der Webseite von Bootsrap unter `http://getbootstrap.com/getting-started/#support` eingesehen werden. In diesem Dokument wird die aktuellste, zum Zeitpunkt der Erstellung des Dokuments veröffentliche Bootstrap-Version 3.3.7 verwendet, und kann als Offline-Paket oder als online Ressource, als sogenannter CDN(Content-Delivery-Network)-Link, verwendet werden. Die Offline-Variante weißt folgende Ordnerstruktur auf:

```
├── css
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   ├── bootstrap-theme.min.css
│   ├── bootstrap-theme.min.css.map
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   └── bootstrap.min.css.map
├── fonts
│   ├── glyphicons-halflings-regular.eot
│   ├── glyphicons-halflings-regular.svg
│   ├── glyphicons-halflings-regular.ttf
│   ├── glyphicons-halflings-regular.woff
│   └── glyphicons-halflings-regular.woff2
└── js
    ├── bootstrap.js
    ├── bootstrap.min.js
    └── npm.js
```

Die benötigte CSS-Datei zur Verwendung von Bootstrap liegt in zwei Versionen im Ordner *css* vor. Die *bootstrap.css* entspricht dabei der für den Menschen leserlichen Form inkl. Zeileneinrückungen und „Code-Formatierungen“. Bei der *bootstrap.min.css* wurden alle unnötigen Leerzeichen sowie Leerzeilen entfernt um eine „schlankere“ Version der Datei zu erhalten

[5] [Boo17e]

und Ladezeiten zu optimieren. Bei allen Beispielen, welche mit Bootstrap gezeigt werden, wird die lokale (offline) *bootstrap.min.css* Datei mittels folgendem Eintrag im HTML-Header verwendet.

```
1   <link rel="stylesheet" href="css/bootstrap.min.css">
```

Alternativ kann auch ein online CDN von einem Drittanbieter, namens „MaxCDN“, verwendet werden.

```
1 <link rel="stylesheet"    href="https://maxcdn.bootstrapcdn.com/bootstrap
    /3.3.7/css/bootstrap.min.css">
```

Die Verwendung des CDN-Pfad bietet darüber hinaus den Vorteil, dass der Browser die Dateien aus dem Cache laden kann, sofern der Benutzer bereits eine andere Webseite besucht hat, welche ebenfalls den gleichen CDN-Pfad für die Nutzung von Bootstrap benutzt. Auf der anderen Seite ist man vom CDN-Anbieter sowie von einer funktionierenden Internet-Verbindung abhängig und mag bei unternehmenskritischen Anwendungen, welche in abgeschotteten Bereichen der Netzwerk-Infrastruktur eingesetzt werden, nicht praktikabel sein. Eine neue Version von Bootstrap 4 ist bereits in Aussicht. Allerdings befindet sich diese seit Januar 2017 noch in der Alpha-Phase und wird deswegen in diesem Dokument nicht weiter behandelt.

Hinweis: Um alle Funktionen von Bootstrap nutzen zu können muss darüber hinaus die dazugehörige JavaScript-Datei *bootstrap.min.js* und die dafür benötigte jQuery-Library (Download unter `www.jquery.com`) eingebunden werden. Diese können wiederum lokal oder über einen CDN-Pfad verlinkt werden. Anbei ein Beispiel über den CDN-Pfad:

```
1 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min
    .js"></script>
2 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.
    min.js"></script>
```

3.2 Die Container-Klasse

Um die CSS-Styles und Funktionen vom Bootstrap nutzen zu können, muss die Website in einer sogenannten „Container-Klasse“ entstehen. Dabei wird zwischen zwei Container-Klassen[6] unterschieden:

.container Stellt einen Container mit einer definierten Breite sowie einem kleinem Seitenabstand (links und rechts) vom Viewport dar. Die Breite skaliert dabei automatisch anhand der verwendeten Bildschirmauflösung.

[6] [Boo17a]

.container-fluid Stellt einen Container zur Verfügung, welcher über die gesamte Breite des *Viewports* geht. (kein Seitenabstand)

Hinweis: Container-Klassen dürfen nicht geschachtelt werden!

Folgender HTML-Code stellt ein Minimal-Beispiel unter Verwendung der *Container*-Klasse dar:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta name="viewport" content="width=device-width, initial-scale=1.0">
5  <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
6  <meta charset="UTF-8">
7  <title>Responsive Web-Design Beispiel</title>
8  </head>
9  <body>
10 <div class="container">
11 Hier entsteht der responsive Webauftritt!
12 </div>
13 </body>
14 </html>
```

3.3 Das Grid-System

Bootstrap definiert innerhalb der Container-Klasse ein Layout-Design bis zu 12 Spalten. Dabei können diese individuell zusammengefasst werden. Wichtig ist, dass in Summe einer Zeilendefinitionen eine Breite von 12 Spalten eingehalten wird. Abbildung 5 zeigt dabei den Aufbau des Grid-Systems anhand individueller Einteilungen. Analog dazu können beliebige Kombinationen der Layout-Darstellung durchgeführt werden, sofern die Gesamt-Zeilenbreite von genau 12 Spalten eingehalten wird[7]. Sobald die Summe mehr als 12 ergibt, werden alle Spaltenelemente, welche die Summe von 12 übersteigen, automatisch auf die nächste Zeile verschoben. Ändert man die Fensterbreite des Webbrowsers, so passen sich die Spalten ebenfalls automatisch an die Größe des Browser-Fensters an.

[7] [Boo17b]

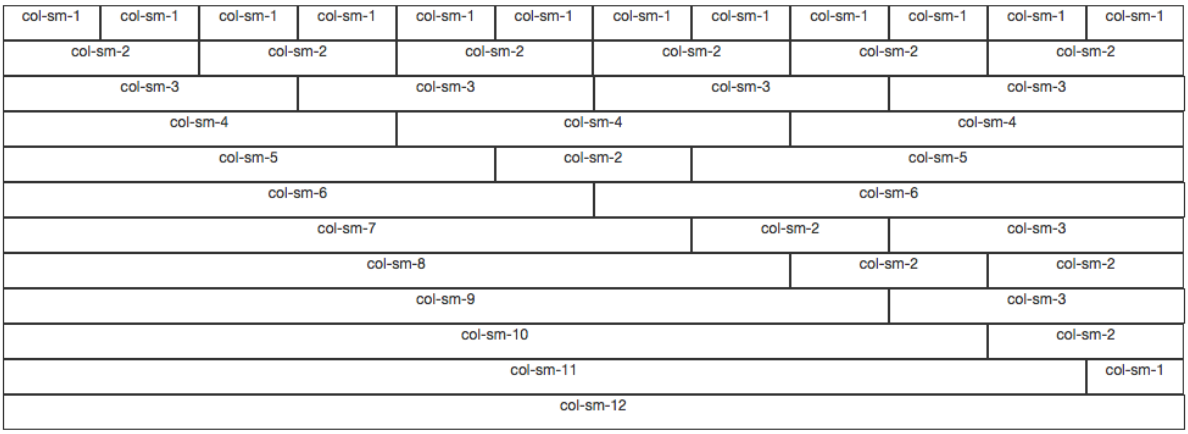


Abbildung 5: Übersicht des Bootstrap Grid-Systems

Darüber hinaus sind im Bootstrap-Framework vier verschiedene Grid-Klassen verfügbar, welche bestimmen, wie die Spalten bei gewissen Auflösungen anzuordnen sind und somit ein ansprechendes Webdesign einfach ermöglichen. Angenommen vier nebeneinander angereihte Spalten mögen bei der Verwendung eines Desktop-Systems kein Problem darstellen, so kann es bei einem Smartphone angenehmer sein, die Spalten nicht nebeneinander, sondern untereinander anzuordnen. Bootstrap stellt genau für diesen Fall vier verschiedene Grid-Klassen zur Verfügung, welche dem Entwickler der Homepage ermöglichen, schnell und einfach einen ansprechenden und automatisch skalierenden Webauftritt zu erstellen. Dabei werden die Grid-Klassenangaben *xs*, *md*, *sm* sowie *lg* innerhalb eines HTML-*div*-Containers eingesetzt. Die *xs*-Klasse entspricht dabei der Darstellung auf kleinen Bildschirmgeräten und ändert die Anordnung der Spalten niemals, unabhängig davon, wie klein die Auflösung beim Endgerät ist. Bei der *lg*-Klasse hingegen, werden alle Spalten in einer Zeile solange untereinander angeordnet, bis die Breite der Bildschirmauflösung 1200 Pixel übersteigt. Erst in diesem Fall werden die Spalten nebeneinander angeordnet. Für die *md*- und *lg*-Klassen gelten ebenfalls spezifische Schwellwerte. Folgende Tabelle zeigt dabei die Charakteristika der einzelnen Grid-Klassen in einer Gesamtübersicht[8]:

[8] [Boo17c]

	Sehr kleine Auflösung (Smartphones)	Kleine Auflösung (Tablets)	Mittlere Auflösung (Desktop-Systeme)	Sehr große Auflösung (Desktop-Systeme)
Klassenprefix	col-xs-*{Spaltenbreite}*	col-sm-*{Spaltenbreite}*	col-md-*{Spaltenbreite}*	col-lg-*{Spaltenbreite}*
Auflösungs-Schwellwerte (horizontal)	< 768 px	>= 768 px	>= 992 px	>= 1200 px
Spalten-Anordnung	Spalten werden immer nebeneinander und niemals untereinander angezeigt	Unterhalb der Auflösungs-Schwellwerts werden die Spalten untereinander angeordnet, sonst nebeneinander		

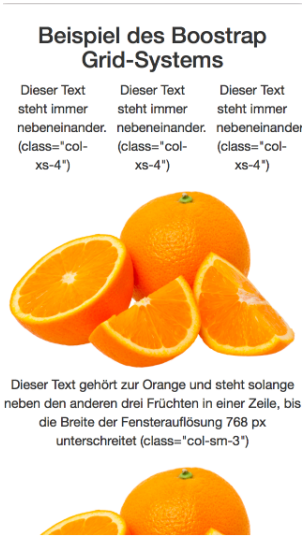
(*{Spaltenbreite}* steht für die Nummer der oben beschriebenen Spaltenwerte.)

Folgender HTML-Code verdeutlicht die Kombination der Grid-Spaltenbreite sowie die Verwendung der Grid-Klassen anhand eines anschaulichen Beispiels:

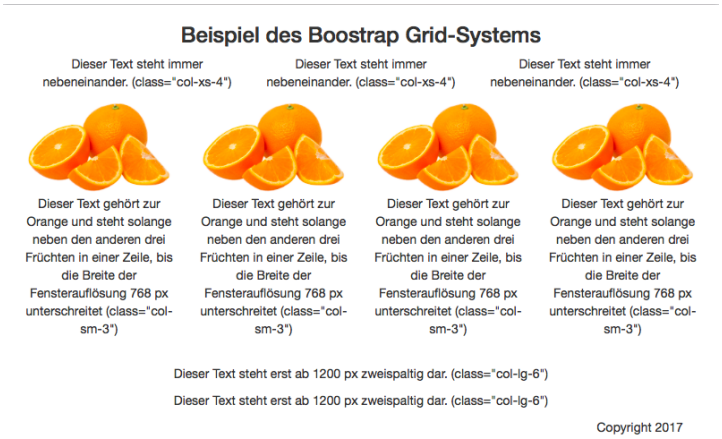
```
1  <!DOCTYPE html>
2  <html lang="de">
3  <head><title>Bootstrap - Grid-Beispiel</title><meta charset="utf-8">
4  <meta name="viewport" content="width=device-width, initial-scale=1">
5  <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css"></head>
6  <body>
7  <div class="container">
8  <h3 class="text-center">Beispiel des Boostrap Grid-Systems</h3>
9  <div class="row text-center">
10 <div class="col-xs-4"><p>Dieser Text steht immer nebeneinander. (class="
       col-xs-4")</p></div>
11 <div class="col-xs-4"><p>Dieser Text steht immer nebeneinander. (class="
       col-xs-4")</p></div>
12 <div class="col-xs-4"><p>Dieser Text steht immer nebeneinander. (class="
       col-xs-4")</p></div>
13 <br>
14
15 <div class="row">
16 <div class="col-sm-3"><img src="pics/orange.jpg" style="width:100%" alt="
       Image"><p>Dieser Text gehört zur Orange und steht solange neben den
       anderen drei Früchten in einer Zeile, bis die Breite der
       Fensterauflösung 768 px unterschreitet (class="col-sm-3")</p></div>
17 <div class="col-sm-3"><img src="pics/orange.jpg" style="width:100%" alt="
       Image"><p>Dieser Text gehört zur Orange und steht solange neben den
       anderen drei Früchten in einer Zeile, bis die Breite der
       Fensterauflösung 768 px unterschreitet (class="col-sm-3")</p></div>
18 <div class="col-sm-3"><img src="pics/orange.jpg" style="width:100%" alt="
```

```
    Image"><p>Dieser Text gehört zur Orange und steht solange neben den
    anderen drei Früchten in einer Zeile, bis die Breite der
    Fensterauflösung 768 px unterschreitet (class="col-sm-3")</p></div>
19  <div class="col-sm-3"><img src="pics/orange.jpg" style="width:100%" alt="
    Image"><p>Dieser Text gehört zur Orange und steht solange neben den
    anderen drei Früchten in einer Zeile, bis die Breite der
    Fensterauflösung 768 px unterschreitet (class="col-sm-3")</p></div></
    div><br>
20  <div class="col-lg-6"><p>Dieser Text steht erst ab 1200 px zweispaltig dar
    . (class="col-lg-6")</p></div>
21  <div class="col-lg-6"><p>Dieser Text steht erst ab 1200 px zweispaltig dar
    . (class="col-lg-6")</p></div>
22  </div>
23  </div>
24  </body>
25  </html>
```

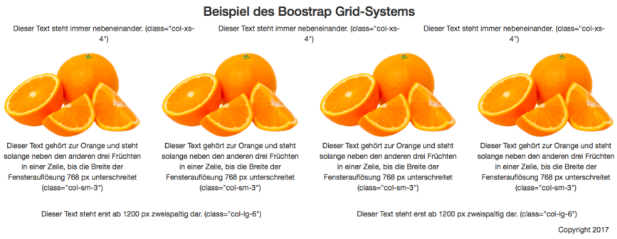
Es wird darauf hingewiesen, dass pro Zeile (engl. „Row“) die Summe aller Spaltenangaben zwölf ergeben muss! Abbildung 6 stellt die Anzeige der Webseite mit verschiedenen Auflösungen dar. Man beachte , dass die entsprechenden Zeilenabschnitte zu einem nebeneinander und bei Unterschreitung gewisser Auflösungsbreiten auch untereinander angeordnet werden.



(a) Mobile Ansicht mit einer Auflösung von 320 x 568 Pixel



(b) Ansicht mit einer Auflösung von 800 x 600 Pixel



(c) Desktop-Ansicht mit einer Auflösung von 1366 x 768

Abbildung 6: Demonstration der Bootstrap Grid-Klassen

Bisher wurde bei einer Spaltenangabe immer nur eine Grid-Klasse verwendet. Sobald man den Schwellwert überschritten hat bleiben die Spaltengrößen, wie sie definiert wurden und skalieren nicht mit der größer werdenden Auflösung mit. Mittels folgender Angabe kann man darüber hinaus für die jeweilige Grid-Klasse spezifische Angaben zur Spaltengrößen angeben[9]:

```
1 <div class="row">
2   <div class="col-xs-4 col-md-6 ">Text 1</div>
3   <div class="col-xs-8 col-md-6 ">Text 1</div>
```

[9] [?]

```
4 </div>
```

Anhand diesem Beispiel wird ersichtlich, dass für die Ansicht auf einem Smartphone die Spaltengrößen mit dem Seitenverhältnis „4 zu 8“ angezeigt wird. Hingegen bei einer Auflösung mit 992 px (oder größer) wird die Spaltenverteilung auf jeweils 50 Prozent der Bildschirmgröße angepasst. Dieses Beispiel kann auch um die weiteren (in diesem Beispiel nicht verwendeten) Grid-Klassen dementsprechend erweitert werden. Damit wurden die wichtigsten Grundlagen des Bootstrap Grid-Systems behandelt. Weitere Informationen zum Grid-System sind unter `http://getbootstrap.com/css/#grid` verfügbar.

3.4 Navigationsleisten ansprechend gestalten

Bis dato wurden bisher nur einseitige Webseiten behandelt. In der Praxis existieren meist mehrere Unterseiten, welche über eine Navigationsleiste übersichtlich dargestellt und aufgerufen werden können. Diese befindet sich meistens an der Seite oder im oberen Bereich des Webauftritts. Am Anfang des Dokuments wurde die Webseite der Hochschule München gezeigt. Betrachtet man den mobilen Webauftritt, so wird deutlich, dass die Navigationsleiste in die rechte obere Ecke „minimalisiert“ wurde. Um das Navigationsmenü zu erhalten, muss der Benutzer auf das Symbol mit den drei horizontalen Querbalken klicken, um dieses aufzuklappen und anschließend zur nächsten Seite navigieren zu können. Dies hat den praktischen Vorteil, dass die Anzeige des Navigationsmenüs nicht einen Großteil des Bildschirms einnimmt und dennoch die Bedienung des Navigationsmenüs benutzerfreundliche sowie übersichtlich gestaltet. In diesem Abschnitt wird abschließend die Erstellung eines Navigationsmenüs dargelegt.

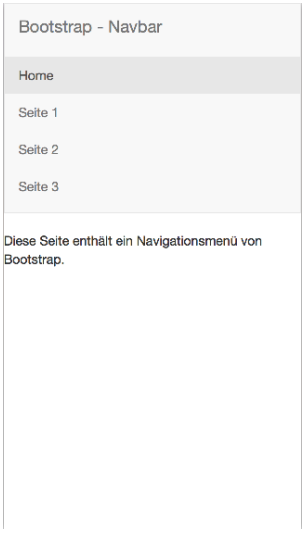
3.4.1 Navigationsleisten ohne Minimalisierung

In Bootstrap können Navigationsleisten komfortabel mittels der Klasse „navbar“ erstellt werden. Im ersten Schritt wird dabei eine Navigationsleiste demonstriert, welche sich entsprechend des Bildschirminhalts anpasst. Dabei wird die Navigationsleiste für mobile Geräte noch nicht minimalisiert dargestellt. Im Fachchargon nennt man dabei das aufklappen bzw. minimieren des Menüs das vom englischem abgeleiteten Wort „*toggeln*“. Dies bedeutet, dass beim ersten Aufruf ein Großteil des Bildschirminhalts vom Navigationsmenü eingenommen wird, sofern es sich um die mobile Version handelt. Eine Minimalisierung in die rechte obere Ecke wird im nächsten sowie auch letzten fachlichen Abschnitt dieser Seminararbeit behandelt. Folgender HTML-Code stellt ein Beispiel einer einfachen Navigationsleiste dar:

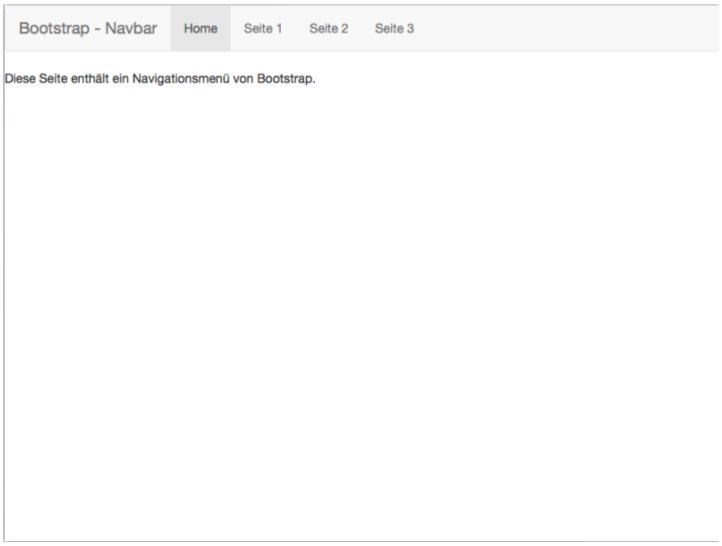
```
1 <!DOCTYPE html>
2 <html lang="de">
```

```
3  <head><title>Bootstrap - Navbar-Beispiel</title><meta charset="utf-8">
4  <meta name="viewport" content="width=device-width, initial-scale=1">
5  <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
6  </head>
7  <body>
8  <nav class="navbar navbar-default">
9  <div class="container-fluid">
10 <div class="navbar-header">
11 <a class="navbar-brand" href="index.html">Bootstrap - Navbar</a>
12 </div>
13 <ul class="nav navbar-nav">
14 <li class="active"><a href="#">Home</a></li>
15 <li class=><a href="page1.html">Seite 1</a></li>
16 <li><a href="page2.html">Seite 2</a></li>
17 <li><a href="page3.html">Seite 3</a></li>
18 </ul>
19 </div>
20 </nav>
21 <p>Diese Seite enthält ein Navigationsmenü von Bootstrap.</p>
22 <p>
23 </body>
24 </html>
```

Abbildung 7 zeigt die Darstellung der Navigationsleiste sowohl für die Desktop als auch für die mobile Darstellung. Bei genauerer Betrachtung des HTML-Codes fällt auf, dass keinerlei Gridklasse für die Anzeigegröße angegeben wurde. Standardmäßig wechselt Bootstrap bei der Verwendung der *nav*-Klasse zur mobilen Ansicht der Navigationsleiste unterhalb einer Darstellungsbreite von 768 px. Dieser Wert kann allerdings in der CSS-Datei angepasst, oder besser mit einer eigenen CSS-Datei überschrieben werden. Dabei ist wichtig, dass die eigene CSS-Datei im HTML-Header nach der Bootstrap-CSS-Datei eingebunden wird, da andernfalls keine Überschreibung stattfindet. Es wird dringend empfohlen, das Überschreiben von Werten in einer eigener CSS-Datei vorzunehmen, da bei einem Update von Boostrap die bestehenden Bootstrap-Dateien einfach ersetzt werden können, ohne die spezifischen Änderungen pflegen zu müssen.



(a) Mobile Ansicht mit Anzeige des untereinander angeordneten Navigationsleiste



(b) Ansicht mit einer Auflösung von 800 x 600 Pixel

Abbildung 7: Demonstration einer einfachen Navigationsleiste

3.4.2 Minimalisierte Navigationsleiste mit Toggle-Menü

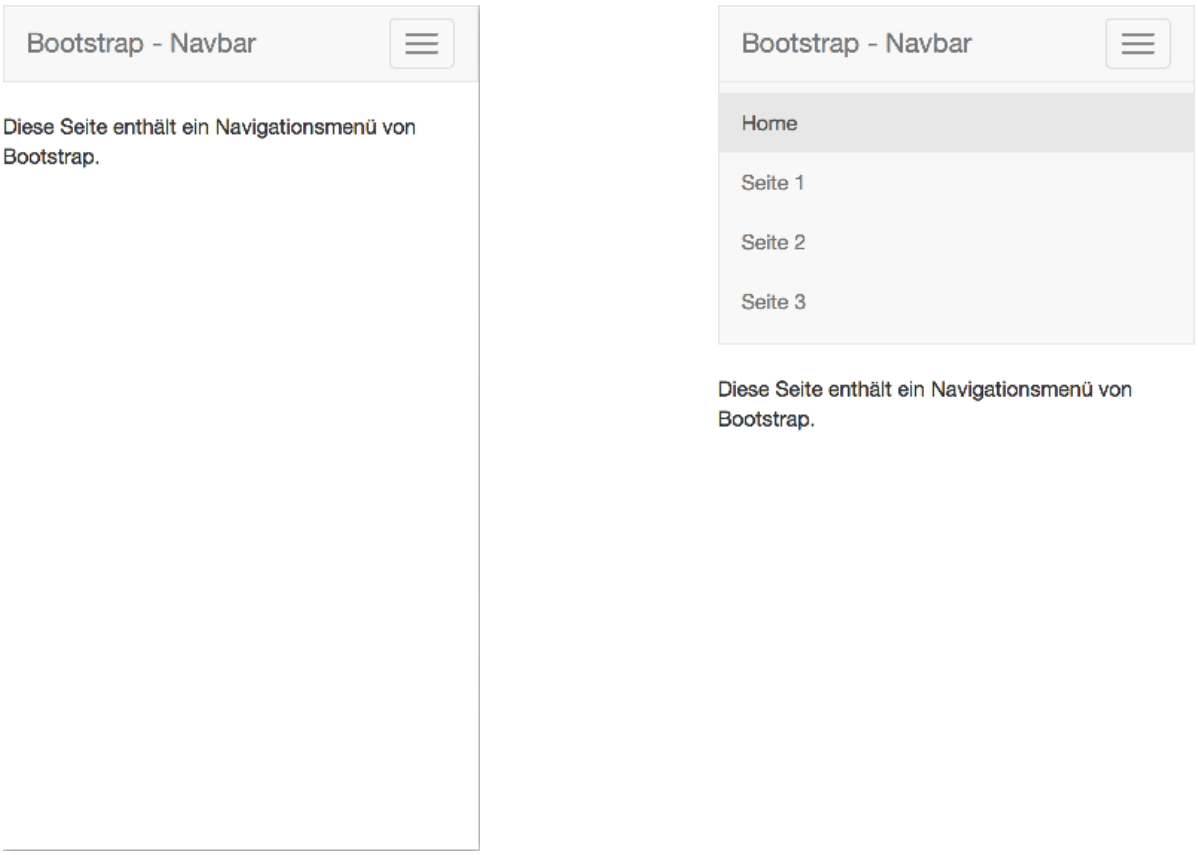
Als letzten (fachlichen) Abschnitt der Ausarbeitung wird auf Basis der vorhergehenden Navigationsleiste die Menüführung dahingehend abgeändert, dass das Navigationsmenü bei der Anzeige für mobile Geräte in die obere rechte Ecke minimiert wird und mittels dem „Toggle“-Mechanismus auf- bzw. eingeklappt werden kann. Bootstrap stellt genau für diesen Zweck die Klasse *navbar-collapse* zur Verfügung. Diese Klasse verwendet das von Bootstrap mitgelieferte JavaScript-Framework[10]. Die Einbindung und Verwendung der JavaScript-Dateien wurde Anfangs bei der Einführung zu Bootstrap erklärt. Im Folgendem wird der dazugehörige HTML-Code dargestellt und dient als einfache sowie minimale Vorlage für die Erstellung eines Webauftritts mittels Bootstrap:

```
1 <!DOCTYPE html>
2 <html lang="de">
3 <head><title>Bootstrap - Navbar-Beispiel</title><meta charset="utf-8">
4 <meta name="viewport" content="width=device-width, initial-scale=1">
5 <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
6 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min
    .js"></script>
```

[10][Boo17d]

```
7  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.
       min.js"></script>
8  </head>
9  <body>
10 <nav class="navbar navbar-default">
11 <div class="container-fluid">
12 <div class="navbar-header">
13 <button type="button" class="navbar-toggle" data-toggle="collapse" data-
       target="#Navigationsleiste">
14 <span class="icon-bar"></span>
15 <span class="icon-bar"></span>
16 <span class="icon-bar"></span>
17 </button>
18 <a class="navbar-brand" href="index.html">Bootstrap - Navbar</a>
19 </div>
20 <div class="collapse navbar-collapse" id="Navigationsleiste">
21 <ul class="nav navbar-nav">
22 <li class="active"><a href="#">Home</a></li>
23 <li><a href="seite1.html">Seite 1</a></li>
24 <li><a href="seite2.html">Seite 2</a></li>
25 <li><a href="seite3.html">Seite 3</a></li>
26 </ul>
27 </div>
28 </div>
29 </nav>
30 <p>Diese Seite enthält ein Navigationsmenü von Bootstrap.</p>
31 <p>
32 </body>
33 </html>
```

Abbildung 8 zeigt die mobile Version des Webauftritts sowie die Ansichten beim ersten Aufruf der Webseite sowie das aufgeklappte Navigationsmenü, sofern dieses „getoggelt“ wurde. Weiterführende Informationen können unter `http://getbootstrap.com/components/#navbar` eingesehen werden.



(a) Standard-Ansicht beim ersten Aufruf der Webseite

(b) Ansicht mit aufgeklapptem Navigationsmenü

Abbildung 8: Anzeige eines in der Praxis gebräuchlichen Navigationsleiste

4 Fazit

Mit der Einführung von HTML5 und CSS3 wurden weitgehende Neuerungen eingeführt, um ansprechende Webseiten sowohl für Desktop als auch für mobile Systeme erstellen zu können. Die zentrale Komponente stellt dabei die Verwendung von *Media Queries* dar, welche ein mächtiges sowie vielseitiges Werkzeug zur Darstellung von Webseiten unter verschiedenen Auflösungen anbietet. Allerdings ist es aus Sicht des Autors wichtig, vor der Gestaltung der CSS-Dateien eine Feinplanung auszuarbeiten, da die Komplexität der CSS-Dateien, gerade bei großen Homepageprojekten, nicht unterschätzt werden darf. Die Verwendung von responsive Web-Design-Frameworks, wie z.B. Bootstrap, bietet dabei einen sehr eleganten und vor allem einfachen Weg, um ansprechende Webseiten zu entwickeln, da sich der Entwickler der Homepage weniger auf die technische Umsetzung der Homepage, sondern mehr auf die visuelle Gestaltung konzentrieren kann, ohne eigene Media-Queries definieren zu müssen. Besonders erwähnenswert ist dabei die einfach gehaltene Funktionsweise des Frameworks. Darüber hinaus bietet Bootstrap eine weitreichende Bibliothek über standardisierte Buttons, Tabellen, Icons, Drop-Down-Menüs, Formularfelder sowie viele weitere Komponenten an, welche auf den Ansatz von responsive Webseiten abgestimmt sind. Da diese Komponenten nur bedingt zum Grundverständnis des responsive Web-Designs weiterhelfen, wurden diese nicht weiter in diesem Dokument behandelt. Eine gut strukturierte und umfangreiche Dokumentation ermöglicht dabei eine schnelle, sowie praxisnahe Einarbeitung und runden das Paket in einer angenehmen Weise ab. Neben der Beliebtheit von Bootstrap lassen sich viele Lösungen zu speziellen Problemstellungen aufgrund einer starken Community in vielerlei Foren wiederfinden. Es wird daher empfohlen, responsive Web-Frameworks (gegenüber der manuellen Erstellung der Cascading-Style-Sheets) den Vorzug zu gewähren, da die Entwicklung der eigenen Webseite, schneller, einfacher und mit einem etablierten Standard gewährleistet wird.

Literatur

[Boo17a] BOOTSTRAP: *Bootstrap - Container-Overview*. Website, 2017. – Online erhältlich unter `http://getbootstrap.com/css/#overview-container`; abgerufen am 7. Mai 2017

[Boo17b] BOOTSTRAP: *Bootstrap - Grid*. Website, 2017. – Online erhältlich unter `http://getbootstrap.com/css/#grid`; abgerufen am 15. Mai 2017

[Boo17c] BOOTSTRAP: *Bootstrap - Grid Media Queries*. Website, 2017. – Online erhältlich unter `http://getbootstrap.com/css/#grid-media-queries`; abgerufen am 6. Mai 2017

[Boo17d] BOOTSTRAP: *Bootstrap - Navbar*. Website, 2017. – Online erhältlich unter `http://getbootstrap.com/components/#navbar`; abgerufen am 7. Mai 2017

[Boo17e] BOOTSTRAP: *Bootstrap Main Page*. Website, 2017. – Online erhältlich unter `http://www.heise.de/tp/deutsch/inhalt/te/2860/1.html`; abgerufen am 28. April 2017

[W3S17a] W3SCHOOLS: *Responsive Web Design - Introduction*. Website, 2017. – Online erhältlich unter `https://www.w3schools.com/css/css_rwd_intro.asp`; abgerufen am 25. April 2017.

[W3S17b] W3SCHOOLS: *Responsive Web Design - Media Queries*. Website, 2017. – Online erhältlich unter `https://www.w3schools.com/css/css_rwd_mediaqueries.asp`; abgerufen am 01. Mai 2017.

[W3S17c] W3SCHOOLS: *Responsive Web Design - The Viewport*. Website, 2017. – Online erhältlich unter `https://www.w3schools.com/css/css_rwd_viewport.asp`; abgerufen am 16. Mai 2017.

[WDW12] WEB DESIGNER WALL, Nick L.: *5 useful CSS tricks for responsive design*. Website, 2012. – Online erhältlich unter `http://getbootstrap.com/css/#grid-example-mixed`; abgerufen am 27. April 2017

Abbildungsverzeichnis