Frank L. Schad

Responsive Webdesign mit CSS 3

Ein Webmasters Press Lernbuch

Version 2.0.0 vom 15.11.2016

Autorisiertes Curriculum für das Webmasters Europe Ausbildungs- und Zertifizierungsprogramm.

www.webmasters-europe.org

Inhaltsverzeichnis

5 Grundlagen des Responsive Designs

In dieser Lektion lernen Sie

- die Unterschiede zwischen Desktop- und mobilen Browsern kennen.
- das Verhalten des Viewports anzupassen.
- was man unter *Liquid Layouts* und *Flexible Images* versteht.
- mobile Websites zu testen.

Eine essenzielle Herausforderung im modernen Webdesign ist die Vielfalt an Geräten, die heutzutage zum Surfen im Web benutzt werden. Eine Website, die auf einem großen Desktop-Monitor gut aussieht, ist auf einem kleinen Smartphone-Display mit Touchscreen u. U. kaum benutzbar. Dies kann unterschiedliche Gründe haben:

- Klassische Websites hatten häufig eine **feste Breite** von z. B. 980px. Dies ist für kleine Smartphone-Bildschirme natürlich viel zu groß. Die meisten Smartphones verkleinern solche Websites, sodass sie möglichst vollständig ins Browserfenster passen. Das Resultat: Der Text ist kaum mehr lesbar und die Interface-Elemente sind so winzig, dass sie nicht mehr bedienbar sind. Weblayouts für Mobilgeräte dürfen keine feste Breite haben, sondern müssen sich der Breite des jeweiligen Displays anpassen.
- Konventionelle Websites sind auf das **Querformat** von Desktop-Monitoren ausgelegt: Sie haben meist horizontale Menüleisten und ordnen die Inhalte in mehreren Spalten nebeneinander an. Smartphone-Displays hingegen nutzen primär das Hochformat, Menüs und Inhalte sollten hier also eher untereinander angeordnet werden.
- Die Bedienung von Touchscreens mit den Fingern unterscheidet sich enorm von der klassischen **Mausbedienung**. Viele bewährte Interface-Elemente sind für Touchscreens zu klein oder schlichtweg ungeeignet — man trifft sie nur schlecht mit den Fingern, oder sie reagieren nicht wie gewünscht.

  Einer der Hauptgründe dafür ist, dass es auf Touchscreens naturgemäß keinen `:hover` gibt. Dieser wird zwar von den meisten mobilen Browsern emuliert, verhält sich aber dennoch anders als auf Desktop-Geräten. So lässt sich z. B. ein Dropdown-Menü, das sich per `:hover` öffnet, auf Touchscreens nicht mehr schließen, weil das *Mouseout*-Ereignis fehlt.

  Die Bedienung mobiler Interfaces kann oftmals eher durch **Gesten** (*swipe, pinch and zoom*) als durch Klicken/Antippen erfolgen. So können z. B. die bewährten *Weiter*- und *Zurück*-Buttons bei Bildergalerien, Slideshows o. ä. entfallen, wenn der Benutzer die Bilder stattdessen durch Wischen weiterblättern kann.

Moderne Weblayouts müssen also *flexibel* sein, sie müssen sich an die verschiedenen Bildschirmgrößen *anpassen*. Man spricht deshalb von *anpassungsfähigen* oder *reaktionsfähigen* Designs, auf englisch **Responsive Designs**.

Das Prinzip dabei ist, dass jedes Gerät — ob Desktop-PC oder Smartphone — zwar grundsätzlich die identischen *Inhalte*, also dieselben HTML-Dokumente, angezeigt bekommt. Lediglich die *Darstellung* der Inhalte passt sich dem jeweiligen Gerät an. Das bedeutet, dass Sie hier mehr denn je auf eine strikte Trennung zwischen Inhalt (HTML) und Design (CSS) achten müssen!

Die Reaktionsfähigkeit selbst wird mit CSS realisiert. Dies geschieht mit Hilfe sogenannter **Media Queries** (»Medienabfragen«), mit deren Hilfe Sie **Layout-Alternativen** für verschiedene Bildschirmgrößen definieren können.

Neben dem Prinzip *Responsive Design* gib es natürlich noch andere Möglichkeiten, Webinhalte für mobile Geräte bereitzustellen: Zum einen die Gestaltung einer völlig unabhängigen Website, die auch unter einer anderen URL erreichbar ist (oftmals beginnend mit *mobile* oder einfach *m* anstelle von *www*), oder aber die Entwicklung nativer Apps für bestimmte Geräte. *Responsive Design* gehört hier jedoch zu den effizientesten (und kostengünstigsten) Varianten.

Zuvor gilt es jedoch, ein Problem zu lösen, das die Grundvoraussetzung für Responsive Design ist: Das unterschiedliche Verhalten der **Viewports** von Desktop- und mobilen Browsern:

5.1 Anpassung des Viewports

Der **Viewport** (auf deutsch etwa *Ansichtsbereich*, *Sichtfenster*) ist der Bereich innerhalb eines Browserfensters, der die eigentliche Website enthält (ohne Titelleiste, Adresszeile, Tableiste usw.). Dieser Viewport verhält sich auf Desktop- und mobilen Browsern unterschiedlich, wenn eine Website zu groß ist, um vollständig hineinzupassen: Auf Desktop-PCs werden einfach Scrollbalken erzeugt, in mobilen Browsern auf Touchscreens wird die Website stattdessen **verkleinert**, sodass möglichst viel davon zu sehen ist. Dies führt natürlich im Normalfall dazu, dass die Texte kaum mehr lesbar sind: Der Benutzer muss Teile der Website mit den Fingern wieder heranzoomen.

Auf iOS und vielen anderen Mobilgeräten wird die Breite des Viewports standardmäßig auf exakt **980px** skaliert, d. h. Websites werden so verkleinert, dass 980px davon in den Viewport passen (dieser Wert galt lange Zeit als durchschnittliche Standardbreite für Websites).

Dieses Verhalten müssen wir deaktivieren. Die Grundvoraussetzung für Responsive Design ist, dass der Viewport immer die **Originalgröße** des jeweiligen Geräts hat, d. h. dass ein Pixel der Website einem Gerätepixel entspricht.

Dies geschieht interessanterweise (noch) nicht mit CSS, sondern in HTML mit Hilfe eines **Meta-Tags**:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
shrink-to-fit=no" />
```

Das Schlüsselwort *device-width* sorgt dafür, dass der Viewport 1:1 an die jeweilige Breite des Geräts angepasst wird. Für die *width* könnten Sie theoretisch auch feste Pixelwerte angeben, was jedoch in der Praxis wenig Sinn ergibt.

initial-scale=1.0 und *shrink-to-fit=no* verhindern, dass die Website vom Browser automatisch skaliert wird.

Als weiterer Wert neben *width=device-width*, *initial-scale=1.0* und *shrink-to-fit=no* wäre theoretisch auch noch *user-scalable=no* möglich. Damit könnten Sie das manuelle Zoomen mit zwei Fingern komplett deaktivieren. Dies ist aber in der Praxis selten sinnvoll, denn warum sollte man den Benutzern dieses Feature nehmen?[9]

Dieser Meta-Tag wirkt sich nur auf Mobilgeräte (Smartphones und Tablets) aus. Auf Desktop-Browser hat er keinerlei Einfluss!

In absehbarer Zukunft soll der Meta-Tag durch eine entsprechende CSS-@-Regel ersetzt werden. Diese wird jedoch zur Zeit (Juli 2016) ausschließlich vom Internet Explorer ≥ 10 mit *-ms*-Präfix unterstützt:

```
@-ms-viewport {
    zoom: 1.0;
    width: device-width;
}

@viewport {
    zoom: 1.0;
    width: device-width;
}
```

Beispiel:

Im Ordner *viewport* im Begleitmaterial zu diesem Kurs finden Sie eine gleichnamige HTML-Datei, die nur ein Bild enthält. In den CSS-Vorgaben im *head* wurde lediglich sichergestellt, dass der *body* kein *margin* und kein *padding* hat, was die Darstellung beeinflussen könnte.

Das Bild hat eine Originalbreite von **320px**. Dies entspricht exakt der Breite eines klassischen iPhone-Displays im Hochformat.[10] Das Bild müsste also eigentlich den gesamten Bildschirm eines iPhones ausfüllen. Wie Sie jedoch in Abb. 5.1 links sehen, erscheint das Bild dennoch viel kleiner: Der Browser skaliert die HTML-Datei wieder auf 980px. Mit dem Meta-Tag jedoch bekommen wir das gewünschte Ergebnis: Die Breite der Webseite entspricht der Breite des Displays (Abbildung rechts).

Diesen Meta-Tag müssen Sie von nun an im *head* jeder Ihrer HTML-Dateien notieren! Ohne ihn ist kein Responsive Design möglich!

5.2 Mobile Webseiten testen

Das oben gezeigte Beispiel können Sie auch testen. Dafür gibt es zwei Möglichkeiten: Einen Live-Test und eine Simulation im Browser.

9. Diese Angabe ist eher für Apps gedacht, die die nativen Interface-Elemente des Geräts verwenden. Solche Oberflächen müssen in der Regel nicht gezoomt werden.
10. Dies gilt auch für iPhones mit Retina-Display. Diese Geräte arbeiten zwar intern mit einer wesentlich höheren Auflösung, jedoch werden Bilder auf die Standardauflösung interpoliert.

Abb. 5.1 *Links die Webseite ohne, rechts mit Viewport-Meta-Tag.*

5.2.1 Live Test

Für einen Live-Test benötigen Sie zwei Dinge:

- Eine Webserver-Software wie z. B. den in Mac-Systemen enthaltenen Apache-Webserver oder alternativ XAMPP[11] (Mac, Windows, Linux) bzw. MAMP[12] (Mac).
- Ein Smartphone/Tablet oder den iOS-Simulator für Mac (Bestandteil von Xcode, kostenlos erhältlich im Mac App Store).

Auf neueren Mac-Systemen hat Apple die Systemeinstellung zum Starten des Apache-Webservers entfernt. Diese kann jedoch nachinstalliert werden unter clickontyler.com/web-sharing/.

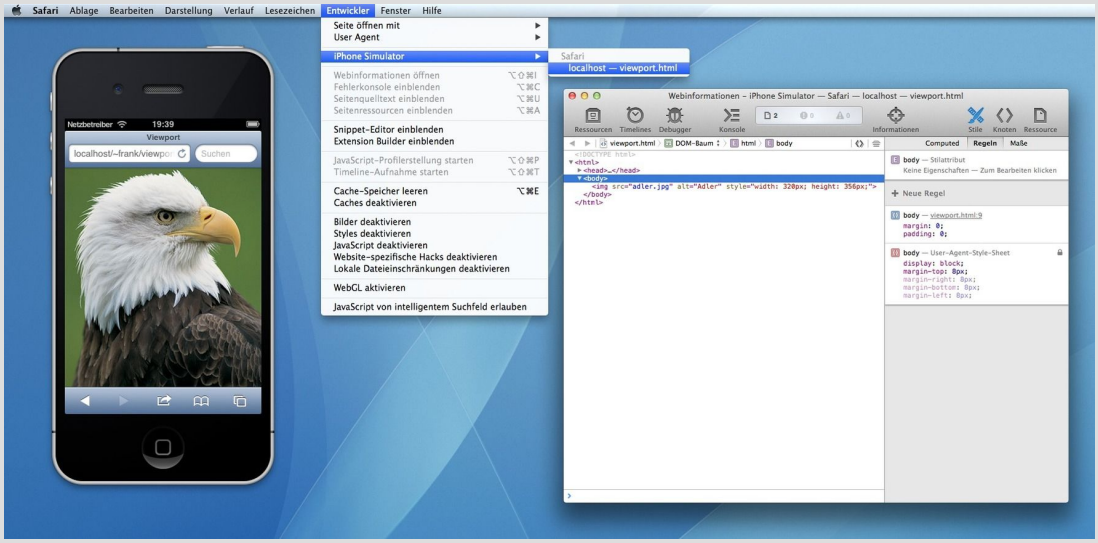
Kopieren Sie dann den Ordner *viewport* ins Stammverzeichnis des Webservers auf Ihrem Computer und starten Sie den Server. Rufen Sie nun die lokale Adresse im Browser Ihres Smartphones/Tablets (das Gerät muss sich im selben lokalen WLAN-Netzwerk wie der Computer befinden) oder des iOS-Simulators auf, z. B.

Smartphone/Tablet:
`http://192.168.178.21/viewport/viewport.html` oder ggf.
`http://192.168.178.21/~benutzername/viewport/viewport.html`
(ersetzten Sie die IP-Adresse durch die Ihres Computers)

11. http://www.apachefriends.org
12. http://www.mamp.info

iOS-Simulator:
`http://localhost/viewport/viewport.html` oder ggf.
`http://localhost/~benutzername/viewport/viewport.html`

Wenn Sie den iOS-Simulator verwenden, taucht die dort gerade angezeigte Webseite auch im *Entwickler*-Menü der Desktop-Version von Safari auf, sodass Sie sie mit dem *Web-Inspector* untersuchen können.



Für die weiteren Übungen in diesem Kurs ist dieses Setup jedoch **nicht erforderlich**. Sie können die Übungen direkt im Desktop-Browser testen. In der Praxis ist jedoch früher oder später ein Test auf möglichst vielen verschiedenen Geräten ratsam.

5.2.2 Testen im Browser

Alle modernen Browser bieten mittlerweile eine Menge nützlicher Hilfsmittel für Webentwickler an. Neben dem unverzichtbaren *Web-Inspector* gehört dazu auch das Testen responsiver Layouts in verschiedenen Viewport-Größen.

- **Firefox** bietet im Menü *Extras → Web-Entwickler* die Option *Bildschirmgrößen testen*. Dort können Sie komfortabel verschiedene Viewport-Größen ausprobieren und ablesen. Allerdings verhält sich dieses Tool ansonsten wie ein Desktop-Browser, d. h. es skaliert die Webseite nicht wie Mobilgeräte, wenn der entsprechende Meta-Tag fehlt (Stand Juli 2016). Zum Testen responsiver Layouts ist es jedoch völlig ausreichend.

- Auch **Chrome** verfügt über eine solche Funktion, die bei geöffneten Entwicklertools (Menü *Anzeigen → Entwickler → Entwicklertools*) verfügbar ist. Klicken Sie dafür in den Entwicklertools im oberen linken Bereich auf das Icon *Toggle device toolbar*. Im Gegensatz zu Firefox emuliert Chrome hier tatsächlich auch das Verhalten mobiler Browser: Fehlt der Viewport-Meta-Tag, wird die Webseite verkleinert.

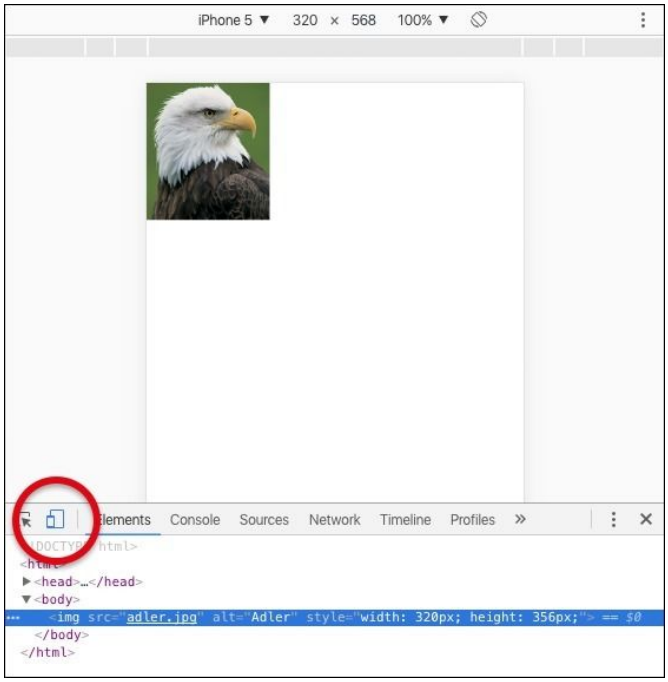


Abb. 5.2 *Mobilgeräte testen in Chrome: Hier fehlt der* `<meta name="viewport"/>`*. Links eingekreist der Button zum Aktivieren der Funktion.*

- In **Safari** finden Sie die Option im Menü *Entwickler → In Modus ›Responsive Design‹ wechseln* (⌥ ⌘ **R**). Das Menü *Entwickler* müssen Sie u. U. zuvor in den Voreinstellungen von Safari unter *Erweitert* aktivieren. Auch hier können Sie zwischen verschiedenen voreingestellten Bildschirmgrößen und Gerätetypen oder eigenen Größen wählen. Der Viewport-Meta-Tag wird berücksichtigt.

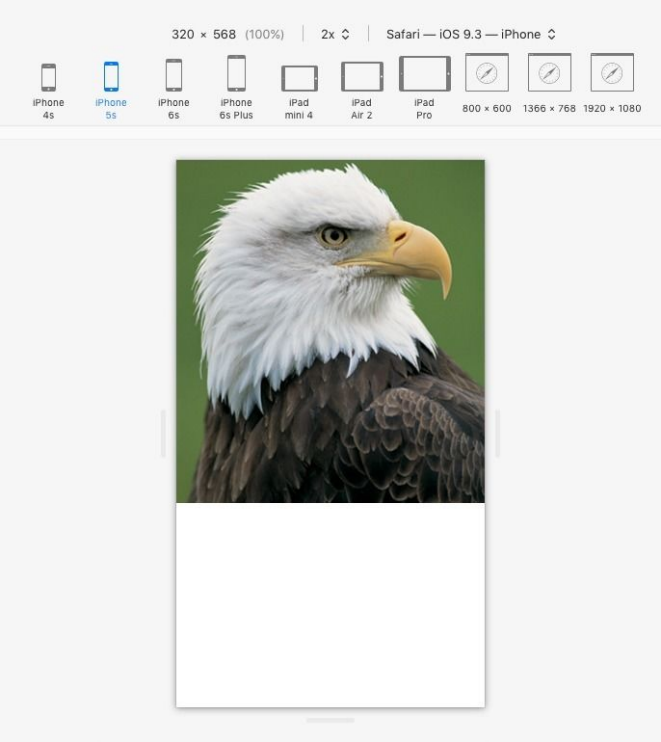


Abb. 5.3 *Modus ›Responsive Design‹ in Safari*

5.3 Fluid Content, Liquid Layouts und Flexible Images

Ich hatte eingangs bereits erwähnt, dass moderne Websites *flexibel* sein müssen. Das Besondere daran ist, dass HTML-Seiten *ohne CSS-Vorgaben* dies bereits von Haus aus sind! Bei einer Veränderung der Fenstergröße umbricht sämtlicher Text einer HTML-Seite neu, sodass er auf den meisten Bildschirmen immer gut lesbar bleibt. Lediglich die Bilder behalten ihre Originalgröße.

Erst mit CSS wurden Weblayouts in der Vergangenheit statisch. Designer haben angefangen, mit festen Breiten in Pixeln zu arbeiten. Dies hat zwar die Gestaltung und exakte Platzierung der Layoutelemente enorm erleichtert, ist aber dem Umstand geschuldet, dass die meisten Designer ihre Wurzeln im Print-Design haben und — bewusst oder unbewusst — dieses Prinzip auch auf Websites übertragen haben: Alles brauchte seinen festen Platz!

Erst in jüngerer Zeit, mit der Verbreitung winziger Smartphone-Displays, hat man eingesehen, dass dies der falsche Weg war. Websites sind keine Print-Dokumente und folgen ganz anderen Regeln.

Die Aufgabe ist deshalb, bei der Gestaltung von Websites mit CSS die Flexibilität von HTML-Seiten **beizubehalten** und sogar noch zu **optimieren**. Die Schlagworte sind hier **Liquid Layouts** und **Flexible Images** (»flüssige Layouts, flexible Bilder«). Und das Beste daran: Das haben wir bereits die ganze Zeit getan!

Wir haben in den vorangehenden Übungen für die meisten Breitenangaben sowohl für Layout-Elemente als auch für Bilder **Prozentwerte** benutzt. Dadurch skalieren diese Elemente bei einer Änderung der Fenstergröße mit und passen sich auch an kleinere Bildschirme an. Mit zusätzlichen Angaben von *min-width* und *max-width* können wir die Breite auf einen bestimmten Toleranzbereich einschränken.

Für Schriftgrößen und Höhenangaben haben wir die Maßeinheiten **em** oder **rem** verwendet. Dadurch passen sich diese Größen an die Standardschriftgröße des Browsers an und skalieren bei einer Änderung der Schriftgröße ebenfalls mit.

Diese Vorarbeiten sind für ein *Responsive Design* bereits die halbe Miete! Nur dort, wo Sie damit nicht weiterkommen, benötigen Sie tiefergreifende Layoutänderungen mit *Media Queries*.

5.4 Fragen zur Selbstkontrolle

1. Wie unterscheiden sich die Viewports von Desktop- und mobilen Browsern?

2. Wozu dient folgender Meta-Tag?

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
shrink-to-fit=no" />
```