Praxis-Guide: Webdesign-Grundlagen für mobile Websites

Autor: Jonas Hellwig 02.11.2012, 16.04 Uhr

Wenn heutzutage vom „mobilen Web“ gesprochen wird, ist damit meistens das so genannte „Responsive Webdesign“ (kurz: RWD) gemeint. Responsive Webdesign heißt soviel wie ansprechbares oder reaktionsfähiges Webdesign und bedeutet, dass eine Website so umgesetzt wird, dass sich das Design sowie alle Inhalte automatisch dem zur Verfügung stehenden Platz im Browser anpassen. Die Darstellung wird dabei mit der CSS3-Technologie „Media Queries“ entsprechend der verschiedenen Display-Größen angepasst.

Der Besucher kann bei einer RWD-Site also alle Inhalte konsumieren, unabhängig davon, ob die Website auf einem 27-Zoll-Display, auf einem Tablet oder mittels Smartphone dargestellt wird. Er sieht immer dieselben Inhalte, lediglich visuell für seine momentanen Bedürfnisse optimiert.

Das Responsive Design der Website orestis.nl mit drei Breakpoints: Smartphone-Version, Tablet-Variante und Desktop-Ansicht.

Doch RWD ist nicht immer das Maß der Dinge. Bereits vor der Umsetzung eines Web-Projekts sollte die Frage stehen, ob die angebotenen Informationen überhaupt dafür geeignet sind. So muss geklärt werden, ob die Benutzer identische oder abweichende Informationen erwarten, wenn sie die Website von unterwegs aufrufen. Besucht ein User beispielsweise ein News-Portal, möchte er Nachrichten lesen. Dabei ist es irrelevant, von welchem Gerät aus er die Website besucht. Doch manchmal haben Nutzer spezielle Erwartungen.

Mobile-only

Bei dieser Art mobiler Site handelt es sich um eine unabhängig konzipierte und programmierte Website. Diese soll nicht auf jedem Gerät eine gute Figur machen – sie wurde ausschließlich für die mobile Nutzung entwickelt und muss entsprechend auch separat gepflegt werden.

Ein solches Konzept ist vor allem dann sinnvoll, wenn der Besucher unterwegs andere Informationen als von einem stationären Computer aus sucht. Ein Beispiel ist die Website der Deutschen Bahn: Werden am PC in erster Linie Tickets gekauft und allgemeine Informationen gelesen, so stehen in der mobilen Version die Fahrplanauskunft und Hinweise zu Verspätungen im Fokus. Ein RWD würde dem mobilen Nutzer in diesem Fall viele überflüssige Informationen anbieten.

Apps

Der Vollständigkeit halber seien auch die Apps kurz erwähnt. Hierbei steht meist eine spezielle Funktion im Vordergrund. Der Benutzer muss sich zudem bewusst für die Nutzung der App entscheiden.

Die unabhängige mobile Webseite der Deutschen Bahn bietet speziell aufbereitete Inhalte; die Umleitung vom Handy geschieht automatisch.

Bei Apps wird zwischen nativen Apps und Web-Apps unterschieden. Native Apps werden speziell für einen Gerätetyp programmiert und müssen heruntergeladen und installiert werden. Web-Apps hingegen basieren auf Webstandards wie HTML5 und CSS3, sind gegenüber nativen Apps im Funktionsumfang eingeschränkt und können ohne Installation verwendet werden. Web-Apps sind auch nicht auf ein spezielles Gerät beschränkt; stattdessen werden sie über den Browser genutzt. Entsprechend groß sind die konzeptionellen Überschneidungen mit „echten“ mobilen Websites.

Abspecken vs. Aufblasen

Bei der Planung einer RWD-Site existieren im Wesentlichen zwei Prinzipien: „Graceful Degradation“ und „Progressive Enhancement“ (auch „Mobile First“). Keine der beiden Vorgehensweisen ist immer die richtige; die Frage ist, ob eine Website vorrangig für große Monitore oder eher für die mobile Nutzung konzipiert wird.

Wer in erster Linie große Monitore bedienen möchte, arbeitet nach dem Prinzip „Graceful Degradation“. Dabei wird die Site zunächst für die große Ansicht konzipiert, gestaltet und programmiert sowie mit allen visuellen Feinheiten, technischen Spielereien und Skripten ausgestattet. Anschließend optimiert man diese Site für die mobile Ansicht. Die Site wird, so gut es geht, „abgespeckt“.

Steht die mobile Nutzung im Vordergrund, ist der Weg genau umgekehrt. Zunächst wird die Website für mobile Endgeräte erstellt. Dabei stehen geringe Dateigrößen, vereinfachte Navigationskonzepte und reduzierte Inhalte im Vordergrund. Anschließend wird die Site für die Desktop-Ansicht optimiert, mithin also „aufgeblasen“.

Design für mehrere Displays

Bei einer „mobile-only“-Website oder einer Web-App lässt sich das Design gut auf die Zielgruppe und das jeweilige Ausgabegerät zuschneiden. Bei der Gestaltung des Layouts wird meist mit festen Abmessungen gearbeitet, da das Design nur geringe Unterschiede zwischen den verschiedenen Display-Größen ausgleichen muss. Der Workflow ähnelt daher stark der Umsetzung einer klassischen Website: Zunächst wird das Layout erstellt, anschließend das Design technisch mit HTML, CSS & Co umgesetzt.
Beim Responsive Webdesign sieht es etwas anders aus, denn das Design muss enorme Größenunterschiede ausgleichen können. Ein Bestandteil des RWD sind deshalb sogenannte Breakpoints. Ein Breakpoint wird im CSS-Code durch ein Media Query definiert, das es erlaubt, ausgewählten HTML-Elementen neue CSS-Eigenschaften zuzuweisen, sobald bestimmte Bedingungen erfüllt sind. Beispielsweise rutscht das Logo über die Navigation, wenn das Display eine bestimmte Breite in Pixeln unterschreitet, oder die Schriftgröße verkleinert sich, wenn das Gerät im Querformat gehalten wird. Media Queries bieten vielfältige Möglichkeiten [1].

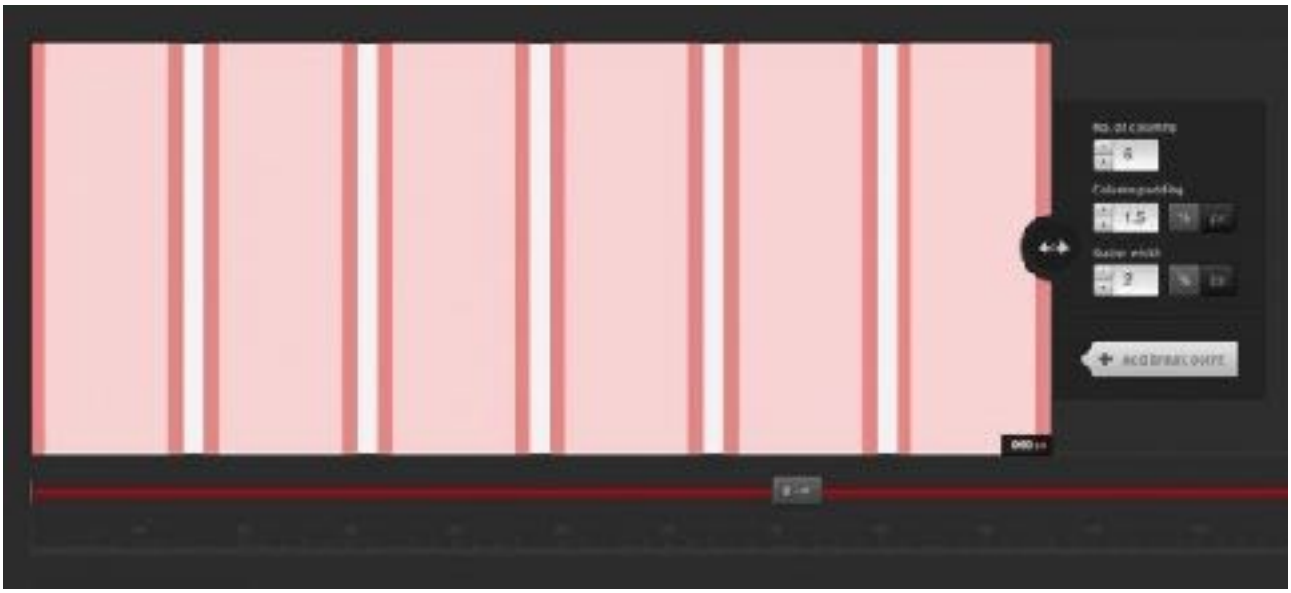
Woran sich Breakpoints orientieren, ist grundsätzlich egal. Häufig orientieren sich Media Queries jedoch an gängigen Display-Größen wie denen von iPad und iPhone. Im Listing wird der standardmäßig rote Hintergrund in der iPad-Ansicht gelb, am iPhone hingegen grün.

Breakpoints

```
body { background: red; }
@media only screen and (max-width: 1024px) {
        body { background: yellow; }
}
@media only screen and (max-width: 480px) {
        body { background: green; }
}
```

Die Bereiche zwischen den Breakpoints werden flexibel gefüllt. Viele Webdesigner arbeiten daher mit einem flexiblen Gestaltungsraster [2], das auf Prozentwerten anstelle von Pixeln basiert. Bereits bei der Gestaltung des Layouts ergeben sich so allerdings

Schwierigkeiten, denn flexible Elemente des Designs lassen sich in Photoshop & Co nicht darstellen.



Mit Gridpak lassen sich Raster auf Basis von Prozentwerten erstellen. Auch Breakpoints können im Raster definiert werden [2].

Da sich das Design an einem Breakpoint deutlich verändert, lohnt es sich, für die verschiedenen Breakpoint-Ansichten separate Layout-Dateien zu verwenden. Responsive Webdesign komplett in Photoshop umzusetzen, ist entsprechend zeitaufwändig und komplex. Viele Webdesigner erstellen daher nur eine Layoutdatei und gestalten anschließend die optimierten Versionen per Code im Browser.

Design ist nicht alles

Natürlich soll mobiles Design gut aussehen, doch dabei wird gerne die Usability und die oft sehr geringe Bandbreite des Ausgabegeräts vergessen. Da mobile Geräte heute meist über Touchscreens bedient werden, sollte mobiles Design so gestaltet sein, dass es mit Daumen und Zeigefinger selbst auf einem kleinen Display gut bedienbar ist. Ausreichend große Interaktionselemente sowie entsprechender Freiraum zwischen den Elementen verhindern, dass Benutzer Probleme mit der Bedienung haben oder versehentlich falsche Funktionen auslösen.

Auch die Schrift sollte größer dargestellt werden, um eine gute Lesbarkeit zu gewährleisten. Smartphones rechnen die Schriften in der Regel automatisch um und stellen sie etwas größer dar. Damit die verschiedenen Schriftgrößen das festgelegte Verhältnis zueinander beibehalten, bietet es sich an, im CSS-Code als Einheit „em“ anstelle von „px“ zu verwenden. Darüber hinaus sollte der Text nie bis ganz zum Rand des Displays laufen, sondern einen ausreichend großen Innenabstand („padding“) einhalten.

Mouse-Over- bzw. Hover-Effekte gibt es bei einem Touchscreen nicht. Entsprechend sollten keine Navigationselemente oder Informationen von einem Mouse-Over-Effekt abhängig sein, ohne eine Alternative für Touchscreens zu bieten. Damit der Benutzer eine Rückmeldung erhält, wenn er ein Interaktionselement berührt, sollte der „:active“-Zustand visuell unterstützt werden. Eine Schaltfläche kann beispielsweise wie eingedrückt dargestellt werden, sobald sie berührt wird.

Auch die Größe der Navigation ist entscheidend. In einer mobilen Version werden die Buttons häufig untereinander dargestellt, da in der Breite nicht genügend Platz ist. Beim Aufruf der Site kann es daher vorkommen, dass das gesamte Display mit der Navigation ausgefüllt ist. Um dies zu vermeiden, wird die Navigation häufig in den Footer gesetzt oder ganz ausgeblendet. Über einen Button im Header kann die Navigation dann ggf. wieder eingeblendet werden. Alternativ bietet es sich an, die Navigation per Skript in eine Select-Liste umzuwandeln [3].

Design für geringe Bandbreiten

Die häufig langsame Internetverbindung mobiler Endgeräte spielt bei der Gestaltung ebenfalls eine wichtige Rolle. Damit eine Website unterwegs zügig lädt, sollte die Dateigröße von Grafiken möglichst gering gehalten und – wenn möglich – Gestaltungselemente über CSS3 realisiert werden. So lässt sich eine Website später unkompliziert für die mobile Nutzung optimierten. Besonders Navigationselemente, Buttons und Schrifteffekte bieten sich für eine Umsetzung mit CSS3 gut an.

Und der Haken?

So schön es auch ist, dass Responsive Webdesign alle Inhalte flexibel auf jeden Monitor transportiert, so ärgerlich sind die damit einhergehenden Nachteile. Die Bilder im Inhaltsbereich einer Website lassen sich zwar mit CSS auf die Breite eines Smartphones herunterskalieren (Listing 2), die Dateigröße bleibt jedoch gleich. Auf dem Smartphone werden also mehr Daten als notwendig geladen, und das ausgerechnet dort, wo nur wenig Bandbreite zur Verfügung steht.

Bilder proportional skalieren

img { width:100%; height:auto; }
Für dieses Problem gibt es leider keine komfortable Lösung. Zwar existieren Workarounds, die es ermöglichen, unterschiedliche Bildgrößen für verschiedene Media Queries anzubieten [4], aber von einem Standard kann keine Rede sein.

Denkbar wäre etwa ein neues HTML-Element („<picture>“). Mit einem solchen Element könnten dann, ähnlich den Elementen „<video>“ und „<audio>“, verschiedene Medienquellen angegeben werden. Auch gänzlich neue Bildformate werden bereits lebhaft diskutiert [5].

Um Videos oder iFrames anzupassen, ist ebenfalls ein wenig Bastelei erforderlich. Nachfolgender CSS-Code skaliert jedoch auch diese Elemente korrekt, indem die unflexiblen Elemente „<iframe>“, „<object>“ und „<embed>“ an die Größe des Elternelements angepasst werden. Nötig dafür ist lediglich ein DIV-Container mit der Klasse „responsiveContainer“. Website-Betreiber, die den Code nicht bei jedem Video manuell einpflegen wollen, können das ganze auch per Skript automatisieren [6].

Dynamische Anpassung für Video und iFrame

```
.responsiveContainer {
        position: relative;
        padding-bottom: 56%;
        padding-top: 30px;
        height: 0;
        overflow: hidden;
}
.responsiveContainer iframe,
.responsiveContainer object,
.responsiveContainer embed {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
}
```

Fazit

Mobile Websites lassen sich mit vertretbarem Aufwand unkompliziert und optisch ansprechend realisieren. Wenn man die üblichen Probleme kennt, kann man bereits im Vorfeld nach Lösungen suchen oder Alternativen ausarbeiten.

Die meisten Fehler entstehen bei der schlechten Planung einer mobilen Version. Zu Beginn des Projekts sollte daher die Frage stehen, ob es ein RWD, eine unabhängige mobile Version oder gar eine (Web-)App sein soll. Entscheidet man sich für RWD, lautet die nächste Frage, ob überwiegend mobile oder eher stationäre Nutzer angesprochen werden sollen. Erst wenn diese Basics geklärt sind, sollte man mit dem Design der Site beginnen.

Da CSS3 in vielen Bereichen Pixelgrafiken und die Arbeit mit Photoshop ablöst, ist es für klassische Screendesigner höchste Zeit, über den Tellerrand zu schauen.

Jonas Hellwig
hat eine Vorliebe für illustrative und technisch fortschrittliche Websites und arbeitet als Designer, Frontend-Entwickler und Trainer in Berlin. Neben Photoshop hat er sich auf WordPress und Social-Media-Integration spezialisiert und ist Gründer der Agentur kulturbanause (kulturbanause.de). Auf blog.kulturbanause.de