Anforderungen an mobilkonforme Webseiten

Handreichung für die Umsetzung und Prüfung der mobilen Nutzbarkeit von Webseiten

Stand 20.06.2017

Arbeitsgruppe Mobile Government des Arbeitskreises Informationstechnik

Baden-Württemberg

MINISTERIUM FÜR INNERES, DIGITALISIERUNG UND MIGRATION

Status	Datum	Bearbeiter/in
Erster Grobentwurf	19.07.2016	Thorsten Schlachter
Vervollständigung	9.10.2016	Dr. Clemens Düpmeier
Kommentierung im Rahmen der AG Mobile Government	21.10.2016	Johannes Föll
Änderungen im Sinne einer Handreichung für Auftragsvergaben	10.03.2017	Renate Ebel
Anmerkungen BITBW	30.05.2017	Ingrid Ehrmann
KIT Checkliste ergänzt	14.06.2017	Dr. Clemens Düpmeier
Schlussredaktion	19./20.06.2017	Renate Ebel, Ruth Wahl

Inhalt

1. Einleitung

Die AG Mobile Government hat im Auftrag des Arbeitskreises Informationstechnik der Landesverwaltung Empfehlungen entwickelt, wie zukünftige Webangebote mobilkonform angeboten werden können. Hierzu reicht es in der Regel nicht, das Rahmendesign des Webangebotes bei Bedarf auf mobile Geräte anzupassen (Nutzung eines „responsiven Designs“). Vielmehr müssen u. a. auch die Darstellung der Inhalte sowie das Kommunikationsverhalten der Webanwendung an das jeweilige Gerät (bzw. dessen Größe) angepasst werden.

Dieses Dokument fasst die wichtigsten Anforderungen an ein mobilkonformes Webangebot zusammen und klassifiziert diese gemäß den Anforderungen an die technische Plattform, den Anforderungen an das Rahmendesign bzw. dasTemplate für die technische Plattform bzw. den Anforderungen an die Autoren. Darüber hinaus werden Sonderthemen, wie z.B. eine Optimierung der Webseite für Suchmaschinen, betrachtet.

Das Dokument kann als Teil der Leistungsbeschreibung für Auftragsvergaben genutzt werden. Es ist geplant, die Empfehlungen als verbindliche Vorgaben in die künftigen IT-Standards der Landesverwaltung aufzunehmen.

2. Varianten mobiler Websites

Es gibt grundsätzlich drei verschiedene Konfigurationsvarianten[1] für mobile Versionen einer Website.

1. Die von Google empfohlene Variante (dies schließt die Sicht einer Suchmaschine auf die Website ein) ist **responsives Webdesign**[2], da sich hier weder URL noch HTML-Code ändern. Die Inhalte der Seiten werden lediglich an die Größe des zugreifenden Mobilgerätes angepasst (skaliert). Der große Vorteil dieser Variante ist, dass das Design immer gleich bleibt und dadurch eine gewisse Konsistenz gewährleistet wird.

2. Die zweite Variante ist die **dynamische Geräteerkennung**, bei welcher zwar die URL gleich bleibt, jedoch auf Basis von Informationen aus dem HTTP-Request (insbes. des *User-Agent*) der HTML-Code der Antwort verändert wird. Diese Art der Konfiguration ist fehleranfällig, da beispielsweise die Liste der User-Agents (aller Browser und ihrer Versionen) ständig aktualisiert werden muss oder die User-Agents nicht immer richtig erkannt werden (z.B. werden einem Desktop-User mobile Inhalte angezeigt oder umgekehrt). Daher sollte

[1] https://developers.google.com/webmasters/mobile-sites/mobile-seo/

[2] Responsives Webdesign ist ein gestalterisches und technisches Paradigma zur Erstellung von Websites, welches die Möglichkeit bietet, auf die jeweiligen Eigenschaften des benutzten Endgerätes zu reagieren.

diese Methode nur bei sehr sorgfältiger Pflege verwendet werden. Außerdem sollte bei dieser Methode der **Vary** HTTP-Header gesetzt werden, damit Suchmaschinen wissen, dass es von dieser Webseite mehrere Varianten für verschiedene Gerätetypen gibt.

3. Die dritte Variante ist die Erstellung einer komplett getrennten, **eigenen mobilen Site**, bei der dem User Inhalte statt über www.baden-wuerttemberg.de z.B. über m.baden-wuerttemberg.de angeboten werden. Die Desktop-Seiten bleiben unbeeinflusst. Ruft ein User über ein Smartphone die Desktop-Seite auf, wird er auf die entsprechende mobile Seite umgeleitet, sofern auf der Desktop-Seite ein rel="alternate" auf die mobile Seite gesetzt wurde.

Es können mehrere dieser Varianten auf derselben Domain verwendet werden, allerdings empfiehlt sich die Beschränkung auf eine davon.

Empfohlen wird das responsive Webdesign für alle Webangebote der Landesverwaltung im Internet und Intranet.

3. Empfehlungen für die Konzeptionsphase eines Webangebotes

Es empfiehlt sich von klein nach groß zu denken: Je kleiner das Display, desto weniger Informationen können (und sollen) angezeigt werden. Es lohnt sich daher, zunächst mit der Konzeption der Seite für Smartphones zu beginnen und zu überlegen, welche Informationen *mindestens* dargestellt werden sollen.

Es sollten mehrere relevante Use-Cases aufgestellt werden, z.B. ist die Situation eines Smartphone-Nutzers, der gerade in der Stadt unterwegs ist, eine andere als die des Besuchers, der mit seinem Tablet zu Hause auf der Couch sitzt, oder eines Mitarbeiters, der an seinem Arbeitsplatz mit einem Desktop-Computer arbeitet. In allen Situationen können dieselben Informationen unterschiedliche Relevanz haben, oder es können sogar unterschiedliche Informationen notwendig sein.

4. Generelle Anforderungen an die technische Umsetzung

Bei einem Webangebot, das vor allem auch mobilen Geräten gerecht werden soll, ist eine der wichtigsten Anforderungen die Verwendung aktueller Webstandards. Es dürfen nur mobilfreundliche Technologien genutzt werden, deren technisch einwandfreie Funktion auf diversen verschiedenen Plattformen (iOS, Android, Windows Mobile etc.) gewährleistet ist und getestet wurde.

Historische Versionen des HTML-Standards aus einer Zeit vor der Entwicklung aktueller Mobiltechnologien bieten naturgemäß keinerlei Unterstützung für aktuelle Mobilplattformen. Des Weiteren orientieren sich die Implementierungen gängiger Webbrowser für mobile Geräte ebenfalls nur an den aktuellen Web-Standards und implementieren keine vollständige Rückwärtskompatibilität zu älteren Standardversionen oder älteren Webtechnologien. Beim HTML-Standard beziehen sich alle aktuellen Entwicklungen in der einen oder anderen Form auf den HTML5 Standard. Der Begriff HTML5 Standard umfasst dabei nicht nur den eigentlichen HMTL Standard in der Version 5, sondern auch alle weiteren Standards, die für moderne Webseiten oder Anwendungen gebraucht werden (z.B. aktuelle Standards zu JavaScript/ECMAScript und dessen Schnittstelle zum HTML-Objektbaum, dem DOM, aktuelle Versionen des CSS-Standards für Stylesheets, sowie standardisierte Schnittstellen der Webbrowser zum Zugriff auf Gerätefunktionen wie GPS oder persistenten Speicherplatz). Des Weiteren inkludiert er auch Zusatzstandards für z. B. Barrierefreiheit oder eben gerade auch für die Unterstützung spezifischer Funktionalitäten von Mobilgeräten (Touchscreen, Bedienung oder Sensorik).

HTML5 ist jedoch nicht für alle technischen Plattformen übergreifend definiert und in jedem Browser implementiert. Browser- und Mobilgerätehersteller sprechen an Stelle des HMTL5 Standards gerne von dem „HTML5 Living“-Standard. Dieser beschreibt an Stelle des offiziellen WWW-Standards eher die Teile des HTML5-Standards, inklusive der oben erwähnten weiteren Standards, die von Browser- und Mobilgeräteherstellern (zumindest von den Mitgliedern der WHATWG-Gruppe) auch tatsächlich implementiert werden bzw. auf der Agenda der Hersteller ist. Bei der Priorisierung der Implementierung des „HTML5 Living“-Standards schauen die Hersteller mittlerweile zunächst auf die Unterstützung der Mobilplattformen und dann erst auf die Unterstützung von Desktop-Browserlaufzeitumgebungen („Mobile first“).

Daraus ergibt sich als eine der wesentlichen generellen Anforderungen an die technische Umsetzung einer Website, dass diese auf einer möglichst aktuellen Version des HTML5-Standard bzw. auf den aktuellen „HTML5 Living“-Standard basiert und hierbei nach Möglichkeit nur Konstrukte dieses Standards verwendet, die auf allen zu unterstützenden Geräteklassen implementiert sind. Hierbei sollten allerdings ältere Legacy Browser (z.B. IE < Version 10) außen vor gelassen werden, da diese Browser nicht nur veraltet und von den Herstellern nicht mehr unterstützt werden, sondern auch ein Sicherheitsrisiko sowohl für Nutzer als auch für Websites selbst darstellen. Hierfür ist innerhalb der Landesverwaltung das Verständnis zu schaffen, dass entsprechende Browser (auch aus Sicherheitsgründen) aktuell gehalten werden müssen. Häufig werden veraltete Browser-Versionen noch eingesetzt, da die Fachverfahren langsameren Release-Zyklen unterliegen. Dies behindert jedoch die Umsetzung der Mobilstrategie der Landesverwaltung. Die IT-Standards des Landes werden noch entsprechend angepasst.

Eine etwas schwächere Version dieser Aussage gilt auch für das im Web verwendete Kommunikationsprotokoll HTTP (Hypertext Transfer Protokoll), über das Webserver und Webbrowser oder andere Clients miteinander kommunizieren. Zwar unterstützen

alle gängigen Browser und mobilen Clients noch die Nutzung älterer Versionen des HTTP-Protokolls, allerdings sind diese älteren Versionen nicht dafür konzipiert worden, die Kommunikation zwischen mobilen Clients und dem Server in Bezug auf die Menge der zu übertragenden Daten bzw. auf die Übertragungsperformanz, zu optimieren. Neuere Standards des HTTP-Protokolls, wie der aktuelle HTTP/2 Standard, erlauben hier z.B. die Komprimierung von Daten bzw. ein verbessertes Caching von Daten auf den mobilen Clients.

Bei größeren Webangeboten werden Webseiten typischerweise nicht mit der Hand erstellt, sondern technische Plattformen, wie Portalserver (z.B. Liferay) oder Content Management Systeme (z.B. Typo3, Wordpress oder Joomla) verwendet, die die Webautoren bei der Erstellung von Informationen, die auf einer Webseite dargestellt werden sollen, unterstützen. Diese Systeme generieren dann die eigentlichen Webseiten (den HTML-Code) automatisch, unter Einbeziehung eines Corporate Rahmendesign-Templates und des Inhaltes. Dabei sollte klar sein, dass der erzeugte HMTL-Code nur dann HTML5 konform ist, wenn sowohl das Rahmendesign sauber, auf Basis des HTML5-Standards, entwickelt wurde, als auch die durch das technische System generierten HTML-Artefakte sowie die von den Autoren erstellten Inhalte ebenfalls aus sauberen HTML5-Code bestehen. Hierzu müssen aber die technischen Komponenten des Portalservers oder CMS-Systems, z.B. der eingebettete HTML-Editor für die Autoren oder die Komponenten, die z.B. Inhaltslisten oder Navigationselemente automatisch erzeugen, bereits für die Erzeugung von HTML5 optimiert sein. Dies setzt aber voraus, dass das verwendete technische System zumindest in einer Version verwendet wird, die bereits die wesentlichen Elemente des HMTL5-Standards und zusätzlich auch bereits aktuelle Versionen des HTTP-Standards unterstützt.

Ein weiterer wichtiger Trend für Websites bei der Unterstützung mobiler Geräte bezieht sich auf die Sicherheit persönlicher Daten bzw. auf die Verhinderung von Hackerangriffen auf die mobilen Geräte der Website-Nutzer. Unverschlüsselte HTTP-Verbindungen zwischen einem nutzenden Client und einem Webserver ermöglichen es prinzipiell Dritten, Man-in-the-Middle Attacken sowohl auf den Webserver als auch auf den nutzenden Client, durchzuführen. Hierbei klinkt sich ein Angreifer in den Kommunikationsverkehr zwischen Server und Client ein, hört diesen ab und verändert die übertragenden Daten für seine Zwecke. Dies ist allerdings nur möglich, wenn der Angreifer die übertragenden Daten auch lesen kann, was bei verschlüsselten Daten zumindest nicht ohne einen sehr hohen Aufwand möglich ist. Selbst wenn der eigene Webserver auch ohne Verschlüsselung als sicher betrachtet wird, stellen unverschlüsselte Webserver mittlerweile ein hohes Risiko für Personen dar, die diese Angebote nutzen, da Man-in-the-Middle Hackerattacken für die Nutzer auf Grund ihrer Professionalität kaum noch auszumachen sind. Da Internetangebote von Nutzern mobiler Geräte aber mittlerweile an vielen potentiell ungeschützten Stellen genutzt werden, z.B. in öffentlichen Gaststätten, Bahnhöfen, Flughäfen oder Hotels, deren Internetzugang gemeinsam von vielen Nutzern verwendet werden, steigt die Gefahr des heimlichen Mitlesens von Informationen und Man-in-the-Middle

Attacken stark an. Im Fall eines solchen Angriffs kann der Imageschaden für die Website, die der Client als vermeintlichen Verursacher ausmacht, fatal sein. Daher empfehlen mittlerweile Sicherheitsexperten, aber auch große Internetfirmen wie Google, dass Internetverkehr zwischen einem Client und einem Server grundsätzlich verschlüsselt stattfinden sollte, und mobile Plattformen wie Android, aber auch die Suchmaschine Google, bewerten nicht-verschlüsselte Sites als kritisch für die Mobilnutzung.

Dies führt zu den folgenden allgemeinen Empfehlungen:

- Verwendung des HTML5 bzw. „HTML5 Living“-Standards als Grundlage für die Erstellung von Webseiten unter Einschränkung auf Elemente des Standards, die auf allen zu unterstützenden technischen (Mobil)Plattformen unterstützt werden
- Neue Webangebote sollten nur verschlüsselten Verkehr zwischen Client und Server verwenden
- Bei Verwendung eines technischen Systems zur Generierung der Webseiten (Portal oder CMS)
    - Verwendung einer aktuellen Version des Systems, die HTML5-konform ist und aktuelle Versionen des HTTP-Standards unterstützt (andernfalls Überlegungen, ob mittelfristig die technische Plattform entsprechend aktualisiert werden kann)
    - Verwendung eines HTML5-konformen Corporate Design Templates für das technische System oder mittelfristig die Ablösung des Designs durch ein HTML5-konformes Design Template (am besten gleich unter Berücksichtigung eines responsiven Webdesigns, siehe nächstes Kapitel)
    - Konfiguration des technischen Systems zur Nutzung moderner HTML5-Konstrukte (oftmals erlauben die Systeme auch noch die Generierung älterer HTML-Konstrukte), z.B. Konfiguration des eingebauten HTML-Editors zur Erzeugung von HTML5 bei Eingabe von Inhalten durch den Nutzer
    - Das technische System (der Webserver) sollte so konfiguriert sein, dass es mit Clients nur über das sichere HTTP Secure (HTTPS)-Protokoll kommuniziert. Als Grundlage hierfür sollte TLS ab Version 1.0 in Verbindung mit einer sicheren Cipher Suite und Perfect Forward Secrecy zum Einsatz kommen.

Es wurde bereits auf die Bedeutung des Corporate Rahmendesigns bzw. dessen Implementierung hingewiesen, wenn ein technisches System zur Erstellung und Auslieferung der Webseiten verwendet wird. Die Anforderungen an dieses Rahmendesign bzw. das Design-Template oder „Theme“ für das technische System werden im folgenden Kapitel genauer beschrieben.

5. Anforderungen an das Rahmendesign

Smartphone-Nutzer haben sich längst daran gewöhnt, dass jeglicher Inhalt in vertikaler Ausrichtung präsentiert wird und vorzugsweise in diese Richtung navigierbar ist. Das bedeutet, dass bei einer festen Skalierung kein horizontales Scrolling für den Nutzer notwendig ist. Webdesigner und -entwickler müssen also vorzugsweise darauf achten, dass sich sämtliche Inhalte bei der Geräteausrichtung korrekt an die Displaybreite anpassen und nichts verloren geht, überlagert oder gestreckt wird.

Es empfiehlt sich die Verwendung eines **responsiven Webdesigns** (Variante 1), d. h. der Inhalt passt sich automatisch der Bildschirmgröße an. Die Vielzahl verschiedener Smartphones und Tablets bringt eine hohe Anzahl verschiedener Auflösungen mit sich. Für all diese möglichen Auflösungen muss die Website lesbar sein und entsprechend skalieren. Außerdem sollte darauf geachtet werden, dass die Nutzer die Seite nicht nur im Hochformat besuchen werden, sondern auch im Querformat (Unterstützung von Landscape- und Portrait-Modus). Eine gängige Lösung zur Implementierung eines responsiven Designs ist die Nutzung von CSS-Frameworks (z.B. Bootstrap), die eine Webseite in ein Raster kleinerer Kacheln zerlegen. Ein Inhaltsblock (z.B. ein Bild) kann in diesem Raster mehr als eine Kachel belegen, sich bei Verkleinerung der Fläche, aber auch mit weniger Kacheln, begnügen, d. h. der Inhaltsbereich reduziert damit seine Größe. Ein Designer kann dabei festlegen, welche minimale bzw. maximale Anzahl von Kacheln belegt werden sollen. Reicht der horizontale Platz nicht mehr aus, um die gewünschten Inhalte bei Verkleinerung auf die minimale Größe nebeneinander darzustellen, werden Inhaltselemente, die bei breiteren Flächen nebeneinander angeordnet werden, nun untereinander angeordnet (Fluid-Design). Der Designer kann dabei häufig bestimmen, in welcher Reihenfolge Elemente untereinander angeordnet werden sollen. Diese Strategie führt dann zwangsläufig zu Designs, die man daran erkennt, dass der Inhalt in Blöcken unterteilt ist, die sich dynamisch vergrößern bzw. verkleinern und an bestimmten Stellen (sogenannte Breakpoints) umbrechen. Diese Blöcke sind dabei in Zeilen einer unterschiedlichen Spaltenanzahl organisiert. Typischerweise wird dabei visuell auch noch etwas Platz zwischen den Inhalten der einzelnen Blöcke gelassen und das Design wirkt „luftig“. Beim responsiven Design sollte das Corporate Design ganz klar in den Hintergrund treten zugunsten von Seiten, die funktional und buchstäblich in Sekundenschnelle erfassbar sind. Ein pixelgenaues Design, wie es von Printmedien bekannt ist, ist dabei nicht mehr sinnvoll.

Eine wesentliche Voraussetzung zur Umsetzung solcher responsiven Design-Strategien ist die Unterstützung von sogenannten CSS3 „Media Queries“, einem Kernelement des zu HTML5 gehörigen CSS3-Standards. Media Queries erlauben es gerade, die Größe von Inhalten abhängig von der Browserfenstergröße (bzw. abhängig vom sogenannten Viewport) zu definieren. Browser wie der Internet Explorer in einer Version < 9 unterstützen solche Media Queries nicht, und der IE9

auch nicht vollständig. Daher sind neuere Browser mit Unterstützung dieses Features erforderlich.

Die Inhalte sollen gut lesbar zur Verfügung gestellt werden. Somit sollten Texte und andere Inhalte nicht zu klein sein, so dass der Benutzer den Inhalt für eine bessere Lesbarkeit nicht zoomen bzw. vergrößern muss. Aus Gründen der Barrierefreiheit sollte die Größe der Inhalte auch stets relativ (und nicht absolut) zu vorgegebenen Standardgrößen definiert sein, d.h. sie sollten mit Änderung der Standardgrößen skalieren. Ggf. müssen Inhalte auch entsprechend gekürzt geliefert und/oder dargestellt werden, d.h. auch der Inhalt passt sich ggf. der Bildschirmgröße an (siehe hierzu auch die Empfehlungen für Autoren).

5.1 Viewport und Wahl der Schriftgröße

Um die Anforderung der guten Lesbarkeit zu bedienen, muss beim responsiven Design unbedingt der Darstellungsbereich („Viewport“) der HTML-Seite unter Verwendung des Metadatums „viewport“ konfiguriert werden. Wird diese Information nicht gesetzt, gehen Browser standardmäßig von einer Viewport-Größe eines Desktop-Browsers von 980 Pixeln aus. Auf Mobilgeräten erscheinen solche Seiten als sehr klein, da sie mit Desktop-Browser-Breitenanforderungen dargestellt werden. Der Nutzer muss die Seiten zunächst Zoomen, um überhaupt etwas lesen zu können. Bei Verwendung des „viewport“-Tags

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

wird durch die Angabe width=device-width die Breite (width) des „Viewports“ der jeweiligen Browserseite jeweils an die Gerätebreite (z.B. 480 Pixel) angepasst. Die Angabe initial-scale=1 sorgt dafür, dass ein CSS-Pixel jeweils einer geräte-unabhängigen Pixelgröße entspricht und hiermit beim Drehen des Bildschirms eines Mobilgerätes ebenfalls die volle Breite genutzt wird[3]. Diese Angaben führen dann dazu, dass ein mobiler Browser die Webseite so darstellt, dass ihre Breite der Breite des Bildschirms des Mobilgerätes entspricht.

Nachdem ein Darstellungsbereich festgelegt wurde, sollte die Schriftgröße entsprechend konfiguriert werden, da Schriften ansonsten zu klein dargestellt werden könnten. Google empfiehlt eine Basisschriftgröße (in CSS definiert) von 16 CSS-Pixeln[4] zu verwenden, z. B.

body { font-size: 16px; line-height: 1.2em}

[3] Ohne diese Angabe wird in einigen mobilen Browsern die Seitenbreite konstant gehalten und hineingezoomt anstatt die gesamte Bildschirmbreite zu nutzen.
[4] Bei CSS-Pixeln handelt es sich im Gegensatz zu Bild-Pixeln und Geräte-Pixeln um eine (fast) absolute Größe.

Neben der Schriftgröße spielt auch der Zeilenabstand eine entscheidende Rolle. Wie oben im body-Tag bereits angedeutet, empfiehlt Google hier z. B. die Angabe 1.2em, welche die Standard-Zeilenhöhe von Browsern ist.

Die Schriftgröße anderer Paragraphtypen usw. sollte dann relativ (also in %, em oder rem) zu dieser Basisgröße gesetzt werden z.B.

```
p { font-size: 120%;}
h1 { font-size: 250%;}
```

Spezielle Tags zur Vergrößerung oder Verkleinerung des Fonts verwenden standardmäßig ebenfalls prozentuelle Angaben. Zum Finetuning der Schriftgrößen auf verschiedene Geräte können die bereits genannten Media Query CCS-Eigenschaften weiterverwendet werden. Damit wird die Fontgröße für verschiedene Gerätegrößen unterschiedlich eingestellt, z.B.

```
html{font-size:110%;}
@media(min-width:60em){html{font-size: 90%}}
```

Mit diesen Einstellungen wird definiert, dass bei kleinerer Gerätebreite die Schrift 110% der Defaultgröße (bei uns 16px) sein soll, während auf Geräten mit größerer Bildschirmbreite nur 90% der Default-Schriftgröße verwendet werden soll. Die oben vorgestellten Strategien funktionieren bereits für alle Gerätegrößen sehr gut und ermöglichen auch eine barrierefreie Skalierung aller Schriftgrößen.

Damit mobile Seiten schnell geladen werden können, sollte sich eine Seite möglichst nur auf eine oder wenige Schriftgrößen beschränken.

5.2 Regeln für Inhalte und Inhaltsbereiche

Beim responsiven Design werden Inhalte typischerweise in Blockelemente (z.B. ein -Tag) eingepasst. Die Größe der Blockelemente, z.B. die Breite, wird dabei prozentual in Bezug auf die Gesamtgröße angegeben (z.B. width=33%). Dies führt dazu, dass beim Verkleinern der Browserbreite sich die Größe der Box entsprechend anpasst bzw. verkleinert. Inhalte, die jetzt in solch einer Box untergebracht sind (z.B. Text oder ein Bild), sollten sich dann dementsprechend der Boxgröße anpassen bzw. skalieren (d.h. ihre Breite dementsprechend ändern), damit sie nicht die Skalierung der Box behindern oder aus dieser herauslaufen. Damit die Boxen und die Inhalte nicht zu klein werden, kann man Mindestbreiten angeben, ab denen sich die Box nicht mehr verkleinert. Eine fluide Platzierung der Boxen sorgt dann dafür, dass sich diese automatisch nicht mehr nebeneinander sondern untereinander anordnen, wenn seitlich kein Platz mehr ist (dies nennt sich Fluid Layout, was ein Teilaspekt des responsiven Designs ist).

Hieraus lassen sich die folgenden generellen Anforderungen an Inhalte und Inhaltsbereiche definieren:

- Container für Inhalte sollten ihre Breite relativ zur Viewportbreite anpassen und über minimale Größenzuordnungen festlegen, wann die schmalste Breite erreicht ist, bei der sich ihr Inhalt noch sinnvoll darstellen lässt.
- Inhaltsbereiche, die nebeneinander liegen, sollten automatisch untereinander angeordnet werden, wenn ihre minimale Breite erreicht ist und nicht mehr alle in einer Zeile nebeneinander passen.
- Es sollten keine großen Inhaltselemente mit fester Breite über verschiedene Geräte hinweg verwendet werden. Stattdessen sollten auch die Inhalte mit ihren Containern skalieren (Breitenangabe relativ, z.B. in %)
- Inhalte sollten für eine gute Darstellung nicht auf eine bestimmte Breite des Darstellungsbereichs ausgerichtet werden.
- Die Verwendung von Media Queries ermöglicht es, Eigenschaften wie Minimalgröße etc. für unterschiedliche Geräte angepasst festzulegen und damit indirekt auch das Umbruchverhalten von Inhaltsblöcken individuell für verschiedene Gerätegrößen einzustellen.
- Es sollte grundsätzlich eher min-width statt min-device-width verwendet werden, um möglichst von Gerätedetails unabhängig zu bleiben
- Übergabepunkte (Breakpoints), ab denen eine Box nicht mehr kleiner wird, sondern im Fluiddesign umbricht, sollten nicht abhängig von spezifischen Geräten (z.B. iPhone versus iPad) sondern sinnvollerweise abhängig vom Inhalt definiert werden.
- Bei der Definition solcher Breakpoints wird dabei zunächst von der kleinsten Darstellung ausgegangen. Dann wird größenmäßig auf die größeren Geräte vorgearbeitet („mobile first“).
- Die minimale Breite von Containern, in denen Text enthalten ist, sollte mindestens so groß sein, dass sie eine Textzeile mit 70 bis 80 Zeichen enthalten können
- Bei sehr großen Geräten führen auch zu breite Textblöcke dazu, dass Inhalte schlecht lesbar sind. Somit sollte eine maximale Größe festgelegt werden.
- Häufig ergibt es Sinn, neben den Haupt-Übergabepunkten (Breakpoints), sekundäre Übergabepunkte zu definieren, bei denen nicht „umgebrochen“ wird, sondern bestimmte Details des Layouts einer Inhaltsbox optimiert werden (z.B. die Fontgröße, den Zeilenabstand oder die Ränder so einzustellen, dass die Darstellung besser für diesen Größenbereich passt).

Für die Einbindung von Bildern in Layouts lassen sich bei mobilgerechten Webseiten weitere Regeln definieren. Gegenüber dem klassischen <img>-Tag des HTML4 Standards ist hier das <picture>-Tag vorzuziehen, wenn es darum geht, Bilder abhängig von der Bildschirmauflösung und dem Bildschirmplatz adaptiv in das Design einzubeziehen. Einige grundsätzliche Regeln für Bilder sind:

- Nutzen Sie wie bei anderen Inhalten relative Größen, die mit dem Container für das Bild skalieren
- Verwenden Sie das <picture>-Element, wenn verschiedene Bilder auf Grund der Gerätecharakteristik eingebunden werden sollen
- Verwenden Sie bei <img>-Tags die Attribute srcset und x-Deskriptor, um dem Browser mitzuteilen, welches Bild das am besten geeignete für eine gewisse Pixeldichte auf dem Gerät ist
- Für Barrierefreiheit sollten natürlich für Bilder und Medienelemente textuelle Beschreibungen des Inhaltes im alt-Attribut platziert werden
- Häufig ist es sinnvoll, für eine bessere mobile Nutzung an bestimmten Stellen in einem Design auf Bilder ganz zu verzichten (so werden Bilder häufig für Hintergrundeffekte verwendet, die heutzutage auch mit dem modernen CSS3 gestaltet werden können). Dies reduziert generell die Anzahl der zu ladenden Bilder.
- Immer wieder genutzte Bilder in einem Design können zu CSS-Sprites (eine Art großes Bild, in dem sich an bestimmten Positionen viele kleinere Bilder befinden) zusammengesetzt und die Sprite-Bilddatei dann im Browser gecached werden. Dies reduziert die Ladezeit von Webseiten mit einem Design mit einer Menge von Bildern.
- Richtlinien für den Einsatz unterschiedlicher Bildformate
    - Verwenden Sie JPG für Fotos
    - Verwenden Sie SVG für Vektorgrafiken und Grafiken mit Volltonflächen (z. B. Logos)
    - Verwenden Sie WebPG oder PNG, falls die SVG-Varianten nicht vorhanden sind
    - Verwenden Sie PNG statt GIF, da das Format mehr Farben und eine bessere Kompression unterstützt
    - Verwenden Sie für längere Animationen nicht animierte GIF-Dateien, sondern das <video>-Element

Der folgende CSS-Code zeigt z.B. die Nutzung des <picture>-Elementes, um abhängig von der Größe unterschiedliche Bilder auf das Gerät zu laden.

```
<picture>
    <source media="(min-width: 800px)" srcset="head.jpg, head-2x.jpg 2x">
    <source media="(min-width: 450px)" srcset="head-small.jpg, head-small-2x.jpg 2x">
    <img src="head-fb.jpg" srcset="head-fb-2x.jpg 2x" alt="a head carved out of wood">
</picture>
```

Für weitere Ausführungen zu dem Thema der Bildnutzung wird auf die Einführung in moderne mobilkonforme Webtechnologie unter https://developers.google.com/web/fundamentals/ verwiesen.

Häufig verwenden Designs von Webseiten auch Piktogramme (Icons). Für ein responsives und mobilkonformes Design ist hierbei von besonderer Bedeutung, dass diese Piktogramme bei allen Größenvarianten lesbar sind und sich an die Gerätegröße anpassen. Dies lässt sich mit Piktogrammen, auf Basis von Rastergrafik (Bildern), nur ungenügend erreichen. Als Lösung hierfür können entweder skalierbare Fonts (z.B. Font Awesome) oder aber Vektorgrafikbibliotheken auf Basis des SVG (Scalable Vector Graphic) Standards verwendet werden. Die Fontvariante eignet sich dabei vor allem dazu, Piktogramme in laufender Schrift zu positionieren, da diese als Schrift Layoutbedingungen, wie einer Zeilenhöhe, unterworfen sind. Für die freie Platzierung komplexer Piktogramme sind dagegen SVG-Grafiken besser geeignet.

Für die Einbindung von (statischen) Videos sollten generell keine proprietären Technologien, wie Flash, Java Applets oder Realmedia Browser-Plugins etc. sondern das Standard-HTML5-Element <video> verwendet werden. Dabei kann für unterschiedliche technische Plattformen unterschiedliche Videoformate des gleichen Videos angeboten werden, wie dies im folgenden Beispiel gezeigt wird:

```
<video controls>
  <source src="chrome.webm" type="video/webm">
  <source src="chrome.mp4" type="video/mp4">

 <p>This browser does not support the video element.</p>

</video>
```

Hierbei sucht sich ein Browser das erste Format aus der angegebenen Liste aus, das er unterstützt. Weitere Einzelheiten zum <video>-Element lassen sich wieder unter https://developers.google.com/web/fundamentals/ nachlesen.

5.3 Barrierefreiheit des Rahmendesigns

Es gelten durch das L-BGG die Anforderungen der *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz* (http://www.gesetze-im-internet.de/bitv_2_0). Viele Maßnahmen daraus tragen ebenfalls zur Mobiloptimierung einer Website bei. Generell sollten Webseiten so weit wie möglich barrierefrei gestaltet werden. Bei technischen Systemen setzt dies in der Regel voraus, dass ein Rahmendesign entsprechend barrierefrei konzipiert ist.

5.4 Menüs und Navigationsstrukturen

Für die Navigationselemente eines Designs (z. B. Hauptmenü) gelten gegenüber der Desktop-Version veränderte Anforderungen. Eine große Menge an horizontal

angeordneten Navigationspunkten kann zwar sehr gut in Darstellungen mit einer sehr großen Breite angezeigt werden, führt aber bei Geräten mit geringerer Breite zwangsläufig zu Zeilenumbrüchen, die den Inhalt (Content) stark nach unten verschieben und somit wertvollen Displayplatz verschwenden. Bei einem responsiven Design werden daher horizontale Navigationsmenüs, typischerweise ab einem bestimmten Breakpoint, zusammengeklappt und durch einen Navigationsbutton mit einem „Drop Down"-Navigationsmenü ersetzt. Der Content kann so einen Großteil des sichtbaren Bildschirmbereiches ausnutzen und wird nur bei Bedarf von der Navigation überlagert. Die Anzahl der Menüelemente in einem horizontalen Navigationsmenü sollte auch nicht zu groß sein und gegebenenfalls auf zwei bis höchstens drei Ebenen verteilt werden. Bei Aktivierung des Navigationsbuttons werden dabei häufig alle Navigationselemente gleichzeitig in einem „Mega-Menü" als „Drop Down" angezeigt, um den Benutzer davor zu bewahren, dass er sich durch die Hierarchieebenen manuell durchklicken muss.

Eine weitere Möglichkeit sind senkrecht angeordnete Navigationsmenüs, die sich aus dem linken oder rechten Rand einer Webseite herausfahren lassen (Slide Out Navigation). Bei Mobilgeräten kann diese Form von Menü über eine Wischbewegung vom Rand nach innen ausgefahren werden. Bei Desktopbrowsern erfolgt dies dagegen eher über die Aktivierung durch ein Buttonsymbol.

Die Navigationsmenüs sollten nicht zu tief verschachtelt sein und nur die wichtigsten Menüpunkte auflisten. Das Ziel sollten schlanke und klar strukturierte Menüs mit wenig Text sein. Zentrale Call-to-Action Buttons[5] sollten in direkt sichtbaren Bereichen platziert werden.

Es muss ausreichender Abstand zwischen allen Buttons und weiteren Navigationselementen vorhanden sein (fingerfreundliche Buttons in entsprechender Mindestgröße). Es empfiehlt sich die Verwendung von absoluten Maßen für die Abstände bei bestimmten Breiten unter Nutzung von Media Queries![6]

Die Rückwärtsnavigation durch den Back-Button-Mechanismus des Geräts muss von jeder Seite aus möglich und ein direkter Sprung zurück zur Startseite sollte ebenfalls instrumentiert sein.

Gängige Gesten sollten nach Möglichkeit unterstützt werden (siehe hierzu später die Anforderungen an Interaktionselemente).

Der Zugang zur Desktop-Variante der Website sollte immer zur Verfügung stehen – beispielsweise über einen Switch im Seitenfooter.

---

[5] Als *Call to Action* wird ein Button bezeichnet, wenn dieser häufig auf Landingpages zu finden ist. Er soll ins Auge des Nutzers stechen und eine klare Handlung vorgeben, z. B. die Bestellung eines Newsletters.

[6] Die Fingerspitzen eines Erwachsenen sind durchschnittlich 10 mm breit. Dementsprechend sollten auch Schaltflächen und Links angepasst werden. Häufig benutzte Links und Buttons sollten groß genug sein, selten verwendete Links und Buttons können dagegen kleiner dargestellt werden, jedoch ebenfalls mit genügend Abstand zum nächsten klickbaren Inhalt, z. B. 5mm oder 32 CSS-Pixel.

5.5 Suchfunktion

Viele Nutzer navigieren per Suchfunktion innerhalb einer Website. Der Suchschlitz sollte direkt erreichbar und als solcher erkennbar sein und der „Suchen“-Button könnte z.B. in Form einer Lupe dargestellt werden. Der Suchschlitz sollte über eine Vorschlags-Funktion verfügen, mit der mögliche Anfragen bereits nach der Eingabe der ersten Buchstaben zur Auswahl angezeigt werden, um dem Nutzer möglichst viel Zeit und Aufwand zu ersparen. Auf Mobilgeräten kann die Eingabe in einem Suchschlitz außerdem über eine Audioeingabe erfolgen.

Die Trefferliste sollte nur die wichtigsten Informationen (Titel, Snippet) enthalten und direkt zum Ergebnis (in mobiler Ansicht) führen.

5.6 Geschwindigkeitsaspekte

Die Website-Geschwindigkeit muss optimiert und die Inhalte entsprechend reduziert werden. Das mobile Internet ist teilweise schnell geworden, doch nicht alle Nutzer verfügen über eine LTE oder 3G-Verbindung (UMTS, HSDPA) bzw. diese Technologien stehen nicht flächendeckend zur Verfügung. Im Idealfall lädt die mobile Website, also auch bei GPRS-Verbindungen, so schnell, dass sie mobilen Anforderungen, bei denen der Nutzer in der Regel wenig Zeit mitbringt, gerecht wird.

Wichtige Grundlagen bzw. Faktoren für eine gute Performanz sind:

- Kleine Bilder durch korrekte Formatierung und Komprimierung (siehe auch die Ausführungen dazu im vorherigen Abschnitt)
- Nur wirklich notwendige Bilder verwenden (Ausschmückung vermeiden, s. vorheriger Abschnitt)
- Verwendung von Bild-Sprites (siehe vorheriger Abschnitt)
- Verzicht auf das Rendering von (die ganze Seite) blockierenden Elementen. Alternativ können diese asynchron nachgeladen oder im HTML Code eingebettet werden
- Weiterleitungen vermeiden – da jede Weiterleitung zusätzliche HTTP-Requests erzeugt
- Verwendung nur von notwendigem JavaScript
- Nutzung von Lazy Loading (s. u.)
- Browser-Caching aktivieren, Zeitstempel (Last-Modified) setzen
- Komprimierung aktivieren
- Wichtige Seiteninhalte im sichtbaren Bereich („above the fold“) als erstes laden lassen

- Größe von HTML-, JavaScript/ECMAScript- und CSS-Code so gering wie möglich halten (Kompression, Minifizierung)
- Server sollten innerhalb von 200 ms antworten

Wenn gängige JavaScript-Bibliotheken oder Inhalte (Icons, Schriftarten etc.) genutzt werden (z.B. jQuery), sollten diese von gängigen Adressen („Hosted Libraries“[7], meist von Cloud-Anbietern) geladen werden, um die Chance auf Cache-Hits zu erhöhen.

Bei der Vermeidung unnötiger HTTP-Requests hilft das Box-Prinzip: Innerhalb einer Anfrage wird nicht ein einzelnes Objekt, sondern eine Sammlung gleichartiger Objekte verschickt. Wenn mehrere Icons benötigt werden, dann werden diese in Form einer Sprite-Grafik übertragen, die mehrere unterschiedliche Grafiken enthält (siehe vorherige Ausführung hierzu). JavaScript-Code wird nicht in Form weniger Methoden pro Datei geladen, sondern alle benötigten Funktionen zusammen. Kleine Grafiken können innerhalb der HTML-Seite als Inline-Grafik (Base64-Encoding) mitgeschickt werden.

5.6.1 Asynchrones Laden und Caching

Gerade bei mobilen Webseiten sollten Inhalte, die nicht direkt benötigt werden, verzögert nachladen („Lazy Loading“). Sobald der Nutzer z.B. nach unten scrollt, werden z.B. weitere Inhalte (einer Listenansicht) oder der Footer per AJAX nachgeladen.

Der Cache mobiler Browser ist häufig beschränkt. Ein iPhone stellt z.B. nur bedingt Speicher für einen Browser zur Verfügung, etwa 25 kB je Datei und 500 kB Cache im Browser insgesamt[8]. Daher sollten zu cachende Dateien klein gehalten werden (z. B. JavaScript minifiziert). HTML5 bietet weitere Möglichkeiten an (z.B. Localstorage).

5.7 Elemente zur Nutzerinteraktion

Mobilgeräte werden in der Regel über einen Touchscreen und eine Softwaretastatur bedient, während die Bedienung von Browsern von Desktopgeräten nach wie vor oftmals über Maus und externer Hardwaretastatur erfolgt. Beide Bedienkonzepte müssen bei einer großflächigen Abdeckung aller Geräte mit abgedeckt werden.

[7] Google Hosted Libraries: https://developers.google.com/speed/libraries/
[8] Selbstversuche zahlreicher Autoren

5.7.1 Interaktions- und Bedienelemente

Einige Aspekte wurden bereits im Abschnitt „Anzeige“ angesprochen. So sollten Bedienelemente, wie Button, Checkboxes und dergleichen, ausreichend groß sein, dass sie sich leicht mit den Fingern bedienen lassen. Für die Nutzung auf Smartphones sollte der Daumen stets in der Lage sein, jeden Winkel des sichtbaren Bereichs bzw. zumindest alle Navigationselemente zu erreichen. Das hängt allerdings auch von der Größe des Gerätes ab. Außerdem sollten interaktive Elemente dem Nutzer direktes Feedback darüber geben, dass sie als Interaktionselemente genutzt werden können bzw. welchen Zustand (z.B. „gedrückt“) sie bei der Nutzung haben. Hierzu sind die hierfür vorgegebenen CSS-Stilelemente entsprechend zu instrumentieren.

Auch wenn sich Nutzer mittlerweile an die Skalierung von Inhalten per Daumen und Zeigefinger gewöhnt haben („Pinch-to-Zoom“), sollten Websites ihre Webseiten automatisch so skalieren, dass eine Vergrößerung der Webseite gänzlich unnötig ist. Umgekehrt sollte evtl. sogar verboten sein, dass die Webseite durch eine solche Geste versehentlich so weit vergrößert wird, dass in der Breite all ihre Inhalte gar nicht mehr auf dem Display angezeigt werden. Dies kann z.B. durch Unterbinden der manuellen Skalierung erreicht werden, z.B. mit

<meta name="viewport" content="width=device-width, **user-scalable=no**">

Gesten haben sich bei den zahlreichen mobilen Anwendungen noch zu keinem echten Standard entwickelt, bereichern an vielen Stellen aber das Nutzererlebnis, z. B. bei der Vor- und Zurücknavigation in einer Bildergalerie mittels Wischgeste. Die Nutzung gängiger Gesten trägt zur mobilen Nutzerfreundlichkeit einer Website bei.

Zur Implementierung von Gestensteuerungen ist in der Regel die Verwendung von JavaScript nötig, da ihre Bedeutung je nach Bedienelement unterschiedlich ausfallen kann. Browser unterstützen aber auch eine Reihe von Standardsteuerungen, die – falls sie vom Benutzer unabsichtlich durchgeführt werden – zu ungewollten Effekten führen können. Um dies zu vermeiden, sollten auf bestimmten Elementen diese Standardfunktionen per CSS deaktiviert werden.

Einige über CSS beeinflussbare Standardgesten sind:

touch-action: auto	Der Browser führt die eingebauten Touchgesten aus (siehe andere Optionen)
touch-action: none	Hiermit wird jede Touchunterstützung auf einem Element ausgeschaltet
touch-action: pan-x	Horizontales Scrolling ist als Touchgeste erlaubt

touch-action: pan-y	Vertikales Scrolling ist als Touchgeste erlaubt
touch-action: manipulation	Scrollen in beide Richtungen und Pinch Zooming ist erlaubt

Zur vollständigen Unterstützung aller Gerätetypen sollte die Webanwendung sowohl eine Tastatur, eine Maus, als auch eine Touchscreen-Bedienung realisieren.

Für die Bereitstellung komplexerer Interaktionsmechanismen, die über JavaScript realisiert werden, ist es vorteilhaft, für die Programmierung hierfür Baustein-Bibliotheken mit wiederverwendbaren Komponenten zu verwenden. Diese sind in der Regel bereits so implementiert, dass sie alle Bedienphilosophien implementieren. Außerdem berücksichtigen hochqualitative Komponenten auch Aspekte wie Barrierefreiheit. Für eine einheitliche Nutzererfahrung bieten sich dann Bibliotheken an, die bzgl. der Bedienelemente eine standardisierte Bedienphilosophie und Designsprache implementieren, wie z.B. das Google Material Design, dem die UI-Elemente des Android Betriebssystem und von iOS folgen. Die Polymer Bibliothek stellt hierfür z.B. wiederverwendbare Komponenten für Webanwendungen bereit.

5.7.2 Formulare

Formulare zum Ausfüllen sollten so benutzerfreundlich wie möglich gestaltet werden, da gerade das Eintippen von Daten auf den Smartphones für viele Nutzer sehr lästig werden kann. Nutzer sollten z.B. nicht mehrere Aktionen ausführen müssen, um an ihr Ziel zu gelangen, wenn sich die Eingabe in einem Schritt erledigen lässt. Formulare sollten sich Eingaben merken und gegebenenfalls Vorschläge zum Ausfüllen unterbreiten (dies wird teilweise bereits von Browsern unterstützt).

Die Bedienelemente zur Eingabe, wie Checkboxen, Auswahllisten, boolesche Schalter, Textfelder etc. sollten ebenfalls mobil-konform arbeiten (siehe hierzu auch den vorherigen Abschnitt), d.h.: die Bedienelemente müssen auf Mobilgeräten ebenfalls groß genug dargestellt werden, damit sie mit den Fingern bedient werden können. Des Weiteren sollten zur Unterstützung aller Geräte eine Tastatur, eine Maus und ein Touchscreen unterstützt werden.

Für die Eingabe von Text sollten verschiedene Tastatur-Layouts (z.B. Nummern-Tastatur), inhaltsspezifisch für unterschiedliche Typen von Texteingabefeldern, genutzt werden. Eine weitere Option wäre das Zulassen von Texteingaben über ein Mikrofon eines Mobilgerätes. Die Auswahl vorgegebener Inhalte (wenn z.B. nur einige wenige Auswahloptionen möglich sind) sollen Freitextfeldern vorgezogen werden. Die Eingabe in Freitextfeldern sollte durch Vorschläge unterstützt werden.

Bei der Eingabe von Zahlen, Datumsangaben oder Zeitangaben können Formulare ergonomischer gestaltet werden, wenn hierfür komplexere Komponenten verwendet

werden, die den Nutzer optimal auf dem jeweiligen Endgerät unterstützen (z.B. eine Kalenderansicht zur Auswahl eines Datums) oder eine Komponente, die ähnlich einer digitalen Uhr die Uhrzeit über Stunden und Minutenauswahlmechanismen einstellen lässt. Hierfür bietet es sich wieder an, entsprechende Komponentenbibliotheken zu verwenden.

HTML5 unterstützt aber bereits eine Reihe von Inputformaten, die die Eingabe von Daten über/durch die Geräte bereitgestellten nativen Eingabemechanismen erlauben. Diese vorgegebenen Inputformate sollten dann im HTML instrumentiert sein. Beispiele sind:

url	zur Eingabe von URLs
tel	zur Eingabe von Telefonnummern
email	zur Eingabe von Emailadressen
search	zur Eingabe von Suchausdrücken
number	zur Eingabe einer Zahl
range	Zahl über Sliderstellung eingeben
datetime-local	lokale Datum und Zeit eingeben
date	Eingabe eines Datums
time	Eingabe einer Zeit
week	Eingabe einer Woche
color	Eingabe eines Farbwertes

Über ein <datalist>-Element kann darüber hinaus an ein Textfeld eine Liste von möglichen Ausfüllvorschlägen hängen. Browser speichern mögliche Vorschlaglisten für Eingabefelder, die sie aus bereits ausgefüllten Formularen gewinnen. Um diese Vorschläge nutzen zu können, sollten die Namen von Eingabefeldern gängigen Bezeichnern folgen (z.B. *email* oder *address*) bzw. über das autocomplete-Attribut dem Browser ein Hinweis gegeben werden, welche Art von Information in dem Eingabefeld erwartet wird. Des Weiteren sind Ausfüllvorschläge für Textfelder sinnvoll (placeholder-Attribut), die dem Benutzer anzeigen, von welcher Struktur die anzugebende Information sein soll.

Damit ein Benutzer nicht beim Abschicken eines Formulars durch eventuell falsche Eingaben frustriert wird, sollte eine Validation von Eingaben bereits auf der Clientseite, direkt bei der Eingabe und nicht erst nach Absenden des Formulars auf der Serverseite, erfolgen. Hierzu gibt es bereits in HTML5 Datenvalidierungsattribute wie pattern, min, max, step etc., die genutzt werden können. Eine komplexere

Validierung kann durch JavaScript-Validierungsbibliotheken instrumentiert werden. Um Eingabefehler direkt am entsprechenden Eingabeelement anzuzeigen, gibt es eine Reihe von CSS3-Pseudo-Stilklassen, die für eine solche Anzeige instrumentiert werden können (z.B. „:valid“, „:invalid“ oder „:out-of-range“).

5.7.3 Verknüpfungen

Nutzer von Tablets oder Smartphones wollen oft zügig Rufnummern herausfinden und dann anwählen oder eine Anfahrt recherchieren und dann die Navigation zu dem Ort aufrufen. Für solche Verknüpfungen gibt es mittlerweile spezielle Links (sogenannte Action-URLs), die einen Aufruf, einer speziellen App, auf einem Mobilgerät triggern. Mit solchen Verknüpfungen lassen sich bestimmte Aktionen, die den Nutzern angeboten werden sollen, schnell anbieten, ohne dass der Benutzer erst eine Information kopieren, dann eine App aufrufen und schließlich die gespeicherte Information in die App kopieren, muss.

6. Regeln für Autoren von Inhalten

Die Aufmerksamkeitsspanne von mobilen Besuchern bzw. ihre Bereitschaft zum Verweilen auf einer Seite ist deutlich geringer als die der Nutzer von Desktop-Computern. Der ohnehin schon kleine Platz von Smartphone- oder Tablet-Displays sollte sinnvoll genutzt werden – nicht zu lange Texte, da der mobile Besucher diese nur in den wenigsten Fällen komplett durchliest.

Benutzerstudien haben gezeigt, dass Nutzer typischerweise nur 20 % - 28 % einer Webseite lesen. Die ersten Sätze einer Webseite (die ersten 4 Paragraphen mit einem Maximum von ca. 70 Wörtern) werden dabei am häufigsten gelesen. Daher ist es insbesondere bei einem Fokus auf mobile Seiten wichtig, dass die für den Nutzer wichtigste Information direkt am Anfang der mobilen Webseite gezeigt wird, ohne dass ein Scrollen erforderlich ist. Eine visuelle Auszeichnung der wichtigsten Informationen (z.B. der Öffnungszeiten für einen Laden) unterstützt dabei zusätzlich die Auffindbarkeit.

Die Benutzerstudien haben weiter gezeigt, dass die Nutzer einfache Sätze mit kurzen Worten und einfacher Struktur bevorzugen. Komplexe Nebeninformationen, die nicht unmittelbar zur wirklich wichtigen zu vermittelnden Information gehören, sollten daher vermieden werden. Die einfache journalistische Regel, den Text für ein 11-jähriges Kind zu schreiben, kann hier helfen. In Stichpunkten sollten textuelle Inhalte die folgenden Regeln beachten:

- Halte den Text einfach (lesbar für ein 11-jähriges Kind)

- Entferne Inhalt, der zur Sache nichts beiträgt
- Bringe die wichtigen Informationen auf den Punkt
- Halte die Texte so kurz wie möglich (ohne essentiellen Inhalt zu unterdrücken)
- Reduziere die Anzahl der für den Inhalt notwendigen HTML-Elemente (vermeide zu komplexe und wechselnde Textauszeichnungen)

Ähnliche Regeln gelten auch bei der Verwendung von Bildern und anderen Medien:

- Vermeiden Sie Bilder und Medien, die nichts zur Sache beitragen.
- Wählen Sie Medien so aus, dass sie den beabsichtigten Inhalt auf den Punkt bringen.
- Das wichtige Topic sollte dabei im Vordergrund sein (im Fokus der Kamera und von der Größe gut sichtbar oder den wesentlichen Teil des Bildes ausmachen).
- Der wesentliche Inhalt sollte bei verschiedenen Gerätegrößen jeweils gut sichtbar sein.
- Verwenden Sie bei Inhalten Bilder, nicht nur als Schmuckelemente (z.B. für einen Hintergrund etc.). Viele Bilder in einer Seite kosten nicht nur Performanz, sondern beim Laden auch essentielle Energie des Gerätes, was die Akkulaufzeit reduziert
- Benutzen Sie SVG für Grafiken, da SVG besser auf verschiedene Größen skaliert und direkt mit der Webseite geladen wird

Für komplexere Inhalte wie Tabellen sollte beachtet werden, dass diese über verschiedene Geräte hinweg skalierbar bleiben, ohne dass die Lesbarkeit beeinträchtigt wird. Hier wäre es wünschenswert, dass das technische System hierzu intelligente Komponenten bereitstellt, die komplexe Inhalte auf verschiedene Gerätegrößen optimieren können (responsive Inhaltskomponenten).

7. Rechtliches

Für App-Entwickler und App-Anbieter wird auf die Orientierungshilfe zu den Datenschutzanforderungen verwiesen:

http://www.baden-wuerttemberg.datenschutz.de/wp-content/uploads/2013/02/OH_Apps_20140616.pdf

8. Mobile Websites und Suchmaschinen

Um auch Nutzer von Suchmaschinen auf die Inhalte mobiler Websites zu führen, müssen auch diese für Internet-Suchmaschinen crawlbar sein, d.h. von deren Bots gefunden werden können. Crawler dürfen nicht per robots.txt oder robots="noindex" von Seiten ausgesperrt werden, die über die Suchmaschine zugreifbar sein sollen. Da mobile Seiten häufig mit JavaScript und CSS optimiert sind bzw. Inhalte teilweise dynamisch in die Seite geladen werden, sollte der Crawler einer Suchmaschine auch Zugang zu den JavaScript und CSS-Dateien haben, da er diese bei der Analyse berücksichtigt.

Wenn eine Unterscheidung der Clients (Variante 2) bzw. eine automatische Weiterleitung mobiler Geräte auf eine mobile Landing Page (Variante 3) stattfindet, muss der Mobile-Bot ebenfalls die mobilen Inhalte bekommen bzw. auf die mobile Ausgabe weitergeleitet werden.

Für Webseiten, die je nach Gerätetyp unterschiedliche Webseiten ausliefern, sollte der **Vary** HTTP-Header gesetzt werden, damit der Crawler weiß, dass diese Webseite geräteabhängig andere Webseiten ausliefert.

```
HTTP/1.1 200 OK
Content-Type: text/html
Vary: User-Agent
Content-Length: 5710
```

9. Tests und Qualitätssicherung

Wie auch bei Desktop-Versionen gilt es zu testen, welche Texte, Buttons, Formatierungen etc. funktionieren, z.B. mit multivarianten Tests[9] oder A/B-Tests[10].

Die Google Webmaster Tools sind ein wertvoller Werkzeugkasten, gerade auch in Bezug auf mobile Websites. Sie bieten unter anderem eine Sicht auf die Website, die der einer Suchmaschine entspricht, jedoch auch weitere Perspektiven. Unter „Crawling-Fehler“ gibt es einen Unterpunkt „Smartphone“. Die dort aufgeführten Fehler sollten behoben werden, um eine funktionierende mobile Website zu gewährleisten. Seit einiger Zeit existiert unter „Suchanfragen“ der Unterpunkt „Benutzerfreundlichkeit auf Mobilgeräten“. Dort werden von Google Fehler wie „fehlender Darstellungsbereich“, „Kleine Schriftgröße“ usw. aufgeführt – eben alle

[9] https://www.optimizely.com/de/resources/multivariate-testing/
[10] http://abtests.de/was-ist-ab-testing

elementaren Faktoren, die ein gutes Nutzererlebnis verhindern. Daher sollte auch dieser Punkt regelmäßig gesichtet werden.

Es können auch kommerzielle Tools (z.B. siteimprove) genutzt werden, um u.a. Dateigrößen, Barrierefreiheit, nicht valide Formatierungen etc. einfach und effizient zu überprüfen.

Eine Optimierung, speziell für Tablets, gestaltet sich häufig schwierig, da viele Geräte (auflösungsabhängig) für die Darstellung der Desktop-Version geeignet sind. Die Zwischengröße von Tablets ist ein weiterer guter Grund für die Nutzung eines responsiven Webdesigns[11].

Eine Checkliste zur Durchführung von Tests und zugehörige Werkzeuge enthält die folgende Tabelle:

1	**Erste grobe Prüfung ausgewählter Webseiten der Website auf Mobilkonformität unter Nutzung der Google Search Console**
URL	**https://search.google.com/search-console/mobile-friendly**
Dieser schnelle Check von Webseiten überprüft das Vorhandensein der für Mobilkonformität wesentlichen Metadatenangaben für mobile Webseiten, wie die viewport-Angabe. Mögliche Ausgaben sind z.B. – Darstellungsbereich nicht festgelegt (viewport-Angabe fehlt) – Text ist zu klein zum Lesen (Text skaliert nicht passend zum Mobilgerät) – Anklickbare Elemente liegen zu dicht beieinander (Abstände zu gering für Bedienung mit Fingern) –	
2	**„Google Page Speed Insights“-Webseite aufrufen und Webseiten in Bezug auf Performanz untersuchen**
URL	**https://developers.google.com/speed/pagespeed/insights/**
Die „Google Page Speed Insights“-Webseite erlaubt es, Webseiten in Bezug auf die Performanz des Ladens verschiedener Teile der Webseite in den Webbrowser zu untersuchen. Die Performanz-Resultate werden auch in Hinblick auf den mobilen Zugriff auf die Webseite bewertet. Zusammen mit den Analyseergebnissen werden Tipps präsentiert, wie die Webseite auf mobile Nutzung optimiert werden kann. Mit diesen Tests kann überprüft werden, ob die eigene Webseite Medien-, HTML-, CSS- und JavaScript-Dateien so bereitstellt, dass sie für geringe Bandbreiten optimiert sind. Es werden auch Tipps gegeben, wie die Seite für den mobilen Betrieb bzgl. der Performanz optimiert werden kann.	
3	**Google Web Master Tools für eigene Website instrumentieren**
URL	**https://www.google.com/webmasters/tools/home?hl=de**

[11] Falls statt Responsive Design die Konfigurations-Variante mit „gerätespezifischen Inhalten“ verwendet wird, sollte für Tablets eher die Desktop-Version statt einer mobilen Ansicht ausgeliefert werden.

Die Google Web Master Tools führen ebenfalls die Überprüfungen aus 1), 2) durch, ermöglichen aber darüber hinaus die automatisierte Analyse der kompletten Website (aller Seiten) im Rahmen des Crawlings der Website. Außerdem erkennt der Crawler gegebenenfalls weitere Probleme, wie z.B. fehlerhafte Links auf Ressourcen, die nicht geladen werden können, oder Sicherheitsprobleme der Website.

4	**Testen der Mobilkonformität über mobiReady**
URL	**https://ready.mobi**

Das mobiReady Testtool für mobile Webseiten beinhaltet eine große Anzahl an Tests, die nicht nur die HTML-Struktur, sondern auch CSS-Dateien einer größeren Anzahl von Prüfungen unterziehen. U.a. erkennt mobiReady, ob Größenangaben in CSS Dateien absolut durchgeführt werden und damit auf verschiedenen Geräten die Styles nicht sauber skalieren. mobiReady erkennt auch das Fehlen wichtiger Meta-datenangaben, wie z.B. viewport-Angaben.
Des Weiteren wird die Ladezeit der Webseite analysiert und in Bezug auf Mobilkonformität bewertet. Eine Darstellung der Webseite für unterschiedliche Geräte im oberen Bereich vermittelt sofort einen Überblick, ob das Design der Webseite responsiv ist, oder ob das Design bei einigen Displaygrößen fehlerhaft dargestellt wird. Ein errechneter Index bestimmt, wie mobilkonform die Webseite insgesamt ist.

5	**Testen mit „Think with Google”**
URL	**https://testmysite.thinkwithgoogle.com/**

Das Testtool „Think with Google“ überprüft die Webseite ebenfalls auf mobile Konformität. Es wird jeweils eine Punktzahl für Mobilkonformität sowie für Mobilgeräte-konforme Performanz beim Laden der Webseite ausgegeben.
Zu den Punktzahlen gibt es weitere Informationen, die erläutern, welche Eigenschaften mobilkonform sind bzw. welche Probleme behoben werden müssen.
Ein detaillierter Report kann per Email zugesendet werden.

6	**Chrome Web Developer Tool im Chrome Browser installieren und „responsives Verhalten“ einer Webseite über Geräteemulator testen**
URL	**Installierbar über den Chrome Erweiterungskatalog im Browser**

Die Chrome Web Developer App für den Chrome Webbrowser erlaubt es Entwicklern, das Verhalten von Webseiten detailliert zu untersuchen. Die Geräteemulation der Anwendung erlaubt es Anwendern dabei, das responsive Verhalten von Webseiten direkt im Browser durch eine Vorschau, die für verschiedene Gerätetypen generiert werden kann, zu verifizieren. Hierzu ist eine Webseite bei installierten Plugin zu laden und dann im Kontextmenü der Seite (Rechtsklick auf die Seite) der Menüpunkt „Untersuchen“ aufzurufen. Die Geräteemulator-Vorschau im obigen Bereich zeigt dann die Darstellung auf dem ausgewählten Gerätetyp. Standardmäßig ist der Gerätetyp „responsives Gerät“ eingestellt, bei dem die Darstellung mit der Maus beliebig skalieren und hiermit die Responsitivität bei verschiedenen Browsergrößen getestet werden kann.
Die Web Developer App zeigt des Weiteren im unteren Bereich den Sourcecode der

Website, der bei Bedarf auf das Vorhandensein einer viewport-Metadatenangabe überprüft werden kann.

7 Manuelles Testen mit verschiedenen Geräten und Browsern

URL **-**

Trotz der Tests mit den oben beschriebenen Werkzeugen können diverse Details der Webseitendarstellung nur manuell unter Benutzung verschiedener Geräte und Browser getestet werden. Hierbei sollten kritische Funktionalitäten, wie z.B. spezielle Darstellungen (Tabellen, Kartendarstellungen, etc.) sowie die korrekte Funktionsweise von Navigationselementen stichpunktartig mit verschiedenen Geräten durchprobiert werden, um zu garantieren, dass die Website vollständig erreichbar ist und die Darstellungen auf den verschiedenen Geräten für die Nutzer akzeptabel ist.

10. Referenzen

„Checkliste für die mobile Website – Google pocht auf Optimierung“
https://votum.de/fokus-e-commerce/mobile-website-checkliste/

Google: “Home page and site navigation”
https://developers.google.com/web/fundamentals/getting-started/principles/site-and-page-navigation?hl=en

Google: „Mobile Friendly Websites”
https://developers.google.com/webmasters/mobile-sites/

Keller, Alexander: “Mobile Nutzung – Homepage umbauen”
https://www.google.de/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&ved=0ahUKEwiN9NnYhfDNAhVlGsAKHV_eAXAQFghCMAQ&url=http%3A%2F%2Fwww.handwerk-magazin.de%2F%3Fevent%3Dcmp.cst.downloads.getdownloadfile%26pk%3D736&usg=AFQjCNGSoLLIZrtZIeAN_ck0t9XP2N7MYQ&sig2=l9UkaeKCAeII5M4v6alWuQ&cad=rja

Priebe, Matthias: „Die 5 wichtigsten Anforderungen an mobile Webseiten“
http://www.experto.de/kommunikation/unternehmenskommunikation/online-pr/die-5-wichtigsten-anforderungen-an-mobile-webseiten.html

Ranksider: „15 Punkte für Ihre mobile Landing Page“
http://www.ranksider.de/talk/15-punkte-fur-ihre-mobile-landing-page

Röckel, Holger: „10 wichtige Faktoren für bessere mobile Websites“
https://www.seonative.de/mobile-website-optimierung/

Thibaut, Jasper: „Inbound Marketing: Mobile Usability“
https://www.ranking-check.de/blog/inbound-marketing-mobile-usability-inkl-checkliste/

“4 Mobile Performanz-Regeln”
http://www.omkt.de/4-mobile-performance-regeln/

„Website-Relaunches – der mobile Faktor in 2015“
http://primweb.de/website-relaunches-mobile-2015/