BITV reloaded

Vier Jahre mit der BITV – eine Bestandsaufnahme

Mit dem Inkrafttreten des Gesetzes zur Gleichstellung behinderter Menschen ist Barrierefreiheit nicht mehr nur eine nette Geste sondern für die Behörden des Bundes Pflicht. Seit Juli 2002 gibt es die Barrierefreie Informationstechnik-Verordnung des Bundes (BITV), die in der Zwischenzeit in vielen Bundesländern in entsprechende Verordnungen übernommen wurde. Beinahe genau so alt ist unsere ursprüngliche Serie ›BITV für Alle‹, mit der wir versucht haben, die einzelnen Bedingungen und Anforderungen in umsetzbare Techniken für Konzeptioner, Designer und Techniker zu übersetzen.

Mittlerweile sind sämtliche in der Bundes-BITV gesetzten Fristen abgelaufen; das Web hat sich in den vergangenen Jahren dramatisch weiterentwickelt. Grund genug, sich die Verordnung nochmals genau anzusehen und zu überlegen, was sich bewährt hat und was nicht. Dazu werden wir hier in einer neuen Serie die einzelnen Anforderungen der BITV erneut betrachten, ihre Umsetzbarkeit bewerten und leicht nachvollziehbare Beispiele von typischen Barrieren und Techniken zu deren Vermeidung vorstellen. Damit Sie die Vorgaben der Verordnung besser in Arbeitsabläufe integrieren können haben wir zusätzlich eine tabellarische Übersicht der Zuständigkeiten erstellt.

Was kommt, was geht?

Bei aller Kritik an einzelnen Details der BITV **– sie ist als politische Richtlinie unentbehrlich.** Weniger als technisch umsetzbare Richtlinie für Webentwickler, sondern viel mehr als Signal, dass auch im World Wide Web Menschen mit Behinderung an der allgemeinen Entwicklung teilhaben wollen.

Versetzen wir uns einen Moment zurück in die 90er Jahre. Wenn damals die Sprache auf das Thema ›Barrierefreies Internet‹ kam, war die übliche Antwort: »Wie, Blinde? Was machen die denn im Internet?« *Das* hat sich zum Glück mittlerweile geändert, und *das* ist sicher in weiten Teilen der Existenz der BITV zu verdanken. Sie hat das Thema in den Fokus der Fachleute gerückt, in deren Verantwortung es liegt, den Zugang und die Nutzung für alle potenziellen Nutzer sicherzustellen. Der Zwang, sich mit Qualitätsstandards zu befassen, hat zu einer deutlichen Verbesserung der Angebote des Bundes und vieler Länder geführt.

Bei einzelnen Anforderungen der Verordnung hingegen kann man kein einheitliches Urteil zu deren Sinnhaftigkeit und Umsetzbarkeit fällen. Natürlich sind die grundlegenden Prinzipien eines barrierefreien Angebotes immer gleich, egal welche Versionsnummer die BITV oder auch deren Vorlage in Form der internationalen *Web Content Accessibility Guidelines* (WCAG) tragen mag. Um die kommenden WCAG 2.0 zu zitieren:

<zitat>

- Inhalte müssen wahrnehmbar sein
- Benutzerschnittstellen im Inhalt müssen bedienbar sein
- Inhalte und Bedienelemente müssen verständlich sein
- Inhalte sollten robust genug sein, um mit aktuellen und zukünftigen Benutzeragenten zu arbeiten (inklusive assistiver Hilfsmittel)

Hierbei handelt es sich um die universellen Prinzipien der barrierefreien Informationstechnik – diese **müssen** erfüllt sein, wenn ein Angebot als barrierefrei bezeichnet wird. Der aktuelle Stand der WCAG 2.0 weist leider in weiten Teilen in eine Richtung, die an ihrer Umsetzbarkeit in eine eventuelle BITV 2.0 erhebliche Zweifel aufkommen lassen.

Bleiben also nur die bestehenden Vorgaben der BITV bzw. der WCAG 1.0: ein Teil der Vorgaben wurde von der technischen Entwicklung eingeholt. Im Original finden sich eine Reihe von Checkpunkten, die als Interims-Lösungen gekennzeichnet sind (»Until user agents…« im englischen Text bzw. »bis Benutzeragenten…« in der deutschen Übersetzung), so zum Beispiel die Richtlinien 7 und 10.

In der BITV fehlt diese Einschränkung aus formaljuristischen Gründen, dafür hat der Gesetzgeber als Ersatz die 3 Jahre / 5 %-Regel in der Begründung zur BITV eingeführt. Diese besagt:

<zitat>

»Die Sicherstellung der Verwendbarkeit assistiver Technologien und Browser ist insbesondere dann unverhältnismäßig, wenn die assistiven Technologien [*sic*] und Browser älter als drei Jahre sind und der Verbreitungsgrad in der einschlägigen Benutzergruppe unter 5 % liegt.«

Nur – wer legt fest, ob diese Werte bei problematischen assistiven Hilfsmitteln erreicht sind? Die Verbreitung und die Marktanteile assistiver Hilfsmittel wie z. B. Screenreader kennen wir nicht; falls Sie eine Statistik kennen, freuen wir uns über eine kurze Nachricht.

Die Streichliste

Im Anhang D des Entwurfs der WCAG 2 findet sich eine Auflistung der alten *»until user agents…«*-Klauseln, die nach einhelliger Expertenmeinung einen Stand erreicht haben, an dem sie als erledigt gelten können: Beispiele hierfür sind die Checkpunkte 1.5 (Textäquivalente für Imagemaps), 10.2 (explizite Labels), 10.3 (Textfluss in Layouttabellen), 10.5 (Trennzeichen zwischen Links) und 10.4 (Platzhalter in Formularen).

Andere sind deutlich abgeschwächt worden, wie die Checkpunkte 3.4 (skalierbare Schriften), 7.3 (Einfrieren von Bewegungen), 13.6 (Skip Links für alles und jedes) und 3.2 (Validierung). Einen Vergleich der BITV- bzw. WCAG 1.0-Anforderungen mit den Erfolgskriterien der kommenden WCAG 2.0 sowie den BIENE-Prüfschritten finden Sie in »BITV, WCAG & BIENE – Die Matrix«

Die Serie: BITV für Alle, Reloaded

Somit ist es an der Zeit, die Serie »BITV für Alle« grundlegend zu renovieren. Wie schon in der ursprünglichen Serie schlüsseln wir die Anforderungen und Bedingungen der BITV **nicht** nach Prioritäten auf. In beiden Prioritätsstufen der Verordnung gibt es Dinge, die bei Nichterfüllung eine unüberwindbare Hürde für manche Nutzer darstellen würden. Eine weitere Aufspaltung in »Das sollten Sie unbedingt machen« und »Das können Sie auch noch erledigen, wenn Sie noch Lust dazu haben« würde diese Spaltung in ein technisch zugängliches und ein inhaltlich ***un****zugängliches* Web noch weiter fortschreiben.

Die Serie verlinkt zu den jeweils passenden Prüfschritten aus dem Testverfahren der BIENE 2006 sowie, wenn möglich, zu entsprechenden Erfolgskriterien der Web Content Accessibility Guidelines 2.0. Bitte beachten Sie, daß sich die WCAG 2.0 immer noch in der Phase des Entwurfes befinden und auch in 2006 entgegen der Zusagen noch nicht fertiggestellt wurden. Daher sind eventuell übernommene Strukturen und Inhalte bis zur Fertigstellung der Richtlinie nur vorläufiger Natur.

Ihre Einkaufsliste:

Die in der Serie durchgeführten Tests funktionieren leider nicht mit jedem Browser. Nur Browser, die auf einer modernen Codebasis beruhen und z. B. das W3C DOM vollständig unterstützen, können überhaupt die notwendigen Aktionen in den Seiten durchführen. Daher haben wir uns weitestgehend auf die kostenlos erhältlichen Browser Mozilla/Firefox und Opera beschränkt.

Browser:

- Mozilla / Firefox
- Opera

Browser-Erweiterungen (Firefox):

- Web Developer Extension
- Web Developer Extension (deutsche Version)
- Accessibility Extensions for Mozilla/Firefox
- FireBug
- JAWS-Simulator Fangs
- Rendered Source Chart
- Colour Contrast Analyser Firefox Extension
- WAVE 3.0 Accessibility Tool

Lokalisierte Fassungen der wichtigsten Firefox-Erweiterungen finden Sie bei www.erweiterungen.de.

Browser-Erweiterungen (Opera):

- Web Developer Toolbar & Menu for Opera
- Opera W3-Dev Menu

Browser-Erweiterungen (Internet Explorer):

- Web Accessibility Toolbar (deutsche Version)
- WEB for ALL - AIS-Toolbar
- Internet Explorer Developer Toolbar
- WAVE 3.0 Accessibility Tool

Browser-Erweiterungen (Safari):

- Safari Tidy plugin
- WebDevAdditions 1.0b18
- Drosera
- Web Inspector
- SafariStand

Eigenständige Anwendungen und Online-Tools:

- Photosensitive Epilepsy Analysis Tool (PEAT)
- Web Accessibility Inspector
- Java Accessibility Helper Early Access
- Validome

Weitere Links zu Testwerkzeugen finden Sie im EfA-Blog unter den Tags ›Browser‹, ›Hilfsmittel‹, ›Testen‹ oder ›Werkzeuge‹. Die Web Accessibility Initiative führt eine eigene Liste mit Testwerkzeugen, in der Sie nach Einsatzzweck, Sprachversion etc. filtern können.

Falls Sie Kontakt zu Nutzern assistiver Werkzeuge suchen gibt es hierfür eine Reihe von Institutionen, die über sehr erfahrene Tester verfügen. Eine weitere gute Möglichkeit zum Austausch mit betroffenen Experten sind Mailinglisten wie die WAI-DE-Liste des deutschen W3C-Büros am Fraunhofer-Institut.

Ein paar grundlegende Definitionen

- Die BITV ist nur für den Zuständigkeitsbereich des Bundes gültig (»Behörden der Bundesverwaltung«). Mittlerweile verfügen die meisten Bundesländer aber über eigene Landesgleichstellungsgesetze, in denen oftmals von der Bundes-BITV abgeleitete Verordnungen bestimmt sind. Auch ohne dem gesetzlichen Zwang zur Barrierefreiheit zu unterliegen ist es für viele andere Anbieter sinnvoll, die Richtlinien umzusetzen.
- Wenn wir in dieser Serie von optionalen Formaten sprechen, so sind damit alle Inhalte einer Webseite gemeint, die nicht HTML oder Text sind. Dazu zählen so offensichtliche Dinge wie JavaScript, Java, Flash, Videos und Sounds (inklusive Podcasts), aber auch Bilder und Grafiken, CSS und hartcodierte Formatierungen im HTML selbst. Letzlich alles, was sich in irgendeinem Browser abschalten lässt, oder was in bestimmten assistiven Werkzeugen nicht unterstützt wird, ist somit im Sinne der BITV und damit für die folgenden Tests ein optionales Format.
- Die häufige Nennung bestimmter Browser stellt natürlich keine Wertung dar. Wenn Sie einen anderen Browser in Ihrem täglichen Umgang mit dem Web einsetzen, so können Sie dies auch gerne weiter tun. Wir beschränken uns hier auf bestimmte Browser, weil diese über Features verfügen, welche die beschriebenen Tests vereinfachen oder überhaupt erst ermöglichen.

Die gesamte Serie auf einen Blick:

Start der Serie »BITV reloaded«

Zum Beginn der Serie stehen einige grundsätzliche Überlegungen zu dieser Verordnung...

BITV-Anforderung 1: Äquivalente Alternativen

In der ersten Anforderung der BITV geht es um einige grundlegende Dinge, die schon immer zum »Guten Ton« einer Website gehörten…

BITV-Anforderung 2: Farben und Kontraste

Zirka zehn Prozent Ihres männlichen Publikums ist farbfehlsichtig und kann mit der Anweisung »Zum Bestellen drücken Sie bitte auf den grünen Knopf« nichts anfangen…

BITV-Anforderung 3: Sauberer Code

Sauberer Code ist nicht nur der Zeitvertreib von HTML-Puristen, sondern legt das Fundament für die Barrierefreiheit…

BITV-Anforderung 4: Sprachliche Besonderheiten

Die Kennzeichnung von Besonderheiten wie Sprachwechsel und Abkürzungen erleichter das Verständnis…

BITV-Anforderung 5: Tabellen

Layout-Tabellen haben mittlerweile ihr Verfallsdatum deutlich überschritten…

BITV-Anforderung 6: Neuere Technologien

Ein Blick in die Statistik Ihrer Website enthüllt die Vielzahl der unterschiedlichsten Browser aus verschiedenen Generationen, die auf Ihre Website zugreifen…

BITV-Anforderung 7: Dynamik und Bewegung

Das Web ist weit mehr als nur ein Sammlung wohlstrukturierter statischer Texte. Interaktive Elemente sind oft der bessere Weg, um manche Nutzergruppen zu erreichen…

BITV-Anforderung 8: Skripte und Applets

In dieser Folge geht es um die Zugänglichkeit der Inhalte, die über das bloße HTML oder Bilder hinausgehen und eigene Schnittstellen zur Ausgabe und Bedienung haben…

BITV-Anforderung 9: Geräteunabhängigkeit

Wenn Ihre Seiten für Sprachausgaben, Braillezeilen und andere assistive Werkzeuge zugänglich sind, werden sie auch für Kiosksystemen ohne Tastatur oder mobile Endgeräten bedienbar sein…

BITV-Anforderung 10: Abwärtskompatibilität

Nicht jedes Ausgabegerät verfügt über die selben Möglichkeiten wie der Rechner, auf dem die Website entwickelt wurde…

BITV-Anforderung 11: Webstandards

Die Verwendung von offenen Standards garantiert nicht nur die Interoperabilität Ihres Internetangebots, sondern spart bei Wartung und Pflege bares Geld.....

BITV-Anforderung 12: Orientierung

Vielen Benutzern assistiver Werkzeuge wird der Zugang bereits durch die mangelnde Übersichtlichkeit einer Website verwehrt…

BITV-Anforderung 13: Bedienung

Klassische Usability-Faktoren haben ebenso einen starken Einfluß auf die Barrierefreiheit eines Angebotes…

BITV-Anforderung 14: Verständlichkeit

Selbst wenn die Prüfprogramme auf keine groben Schnitzer hinweisen, ist eine Website damit noch lange nicht für alle Besucher tatsächlich auch benutzbar…

Anforderung 1: Aquivalente Alternativen

»Für jeden Audio- oder visuellen Inhalt sind geeignete äquivalente Inhalte bereitzustellen, die den gleichen Zweck oder die gleiche Funktion wie der originäre Inhalt erfüllen.« <zitat>

Was heißt das?

Gleich zu Beginn der BITV geht es um grundlegende Dinge, die seit jeher zum ›Guten Ton‹ einer Website gehören. Kurz gesagt: **alles was sich in Form von Text ausdrücken lässt sollte auch in Textform vorhanden sein.** Reiner Text bietet als einziges Format die Möglichkeit, maschinell in andere, alternative Ausgabeformate übertragbar zu sein. Texte können über den Lautsprecher ausgegeben oder in Braille umgewandelt werden, akustische Informationen können zusätzlich in Textform dargestellt werden und vieles mehr. Viele Menschen mit sensorischen Behinderungen sind auf die Übertragung in eine Textalternative angewiesen, da versteht es sich von selbst, dass diese Alternativen auch einen adäquaten Ersatz für die in Nicht-Textform angebotenen Inhalte bieten müssen.

Eine ganze Reihe Bedingungen der BITV erfüllt man fast automatisch, wenn man sich an die jeweiligen Standards hält. Die Anforderung 1 ist einer der Punkte der BITV, deren *technische* Umsetzung eigentlich ein Kinderspiel sein sollte, deren *inhaltliche* Umsetzung allerdings Aufwand bedeutet. Mehr dazu in den einzelnen Bedingungen:

Anmerkung
Wenn Sie die Tests zu dieser Anforderung an Ihren eigenen Seiten nachvollziehen wollen, finden Sie in der Einleitung zu dieser Serie eine Liste mit den benötigten Tools und Adressen, wo Sie diese herunterladen können.

Bedingung 1.1: Alternativtexte (Prio. 1)

»Für jedes Nicht-Text-Element ist ein äquivalenter Text bereitzustellen. Dies gilt insbesondere für: Bilder, graphisch dargestellten Text einschließlich Symbolen, Regionen von Imagemaps, Animationen (z. B. animierte GIFs), Applets und programmierte Objekte, Zeichnungen, die auf der Verwendung von Zeichen und Symbolen des ASCII–Codes basieren (ASCII–Zeichnungen), Frames, Scripts, Bilder, die als Punkte in Listen verwendet werden, Platzhalter–Graphiken, graphische Buttons, Töne (abgespielt mit oder ohne Einwirkung des Benutzers), Audio–Dateien, die für sich allein stehen, Tonspuren von Videos und Videos.« <zitat>

Was heißt das?

In dieser Anforderung geht es um die Überprüfung des Vorhandenseins und die Qualität von alternativen Texten zu Bildern und Imagemaps, aber auch von anderen Nicht–Text–Elementen wie `<OBJECT>` und `<SCRIPT>`. Zunächst ganz einfach, die meisten Editoren sehen separate Eingabefelder beim Einfügen von Bildern und anderen Objekten vor. Zumal das `alt`-Attribut seit HTML 4 für die Elemente `IMG` und `AREA` verbindlich vorgeschriebenen ist und optional für die Elemente `INPUT` und `APPLET` ebenfalls vegeben werden *kann*. Aber bei der Frage, *was* ein »äquivalenter Text« ist, scheiden sich die Geister.

Die Vorgabe beschränkt sich nicht nur auf alternative Texte zu grafischen Elementen. Wenn man den Text genau liest, stellt man fest, dass mit einer Universalklausel zu sämtlichen Formaten, die nicht im Text direkt beschrieben werden, alternative Texte eingefordert werden. Streng genommen muss also der *gesamte akustische und visuelle **Inhalt*** einer Webseite nochmals in Textform hinterlegt oder beschrieben werden. Im Detail bedeutet dies:

Bilder, graphisch dargestellten Text einschließlich Symbolen, [...]
Dies betrifft: sämtliche Bilder, Grafiken, Icons, Diagramme, grafisch umgesetzten Text (inklusive bestimmter Image Replacement-Techniken), die aus externen Ressourcen in eine Seite eingebettet werden.

[...] **Regionen von Imagemaps,** [...]
Dies betrifft: sämtliche (grafischen) Formate, bei denen die Interaktion über Koordinaten oder Teilbereiche des Objektes geschieht. Das können Bilder mit nur sehr wenigen Teilbereichen sein, im Ernstfall aber auch Karten, Diagramme oder ähnliches mit dutzenden oder hunderten eigenständigen aktiven Links.

[...] **Animationen (z. B. animierte GIFs),** [...]
Dies betrifft: nicht nur die beispielhaft in der Verordnung angeführten animierten GIFs, sondern sämtliche Formate, mit denen Animationen hinterlegt werden können, also auch Applets, Objekte wie Flash und vieles mehr.

[...] **Applets und programmierte Objekte,** [...] **Scripts,** [...]
Dies betrifft: in HTML 4 ist die Notierung des `alt`-Attributs für die Elemente `INPUT` und `APPLET` optional, hier wird sie insbesondere für `INPUT`-Elemente vom Typ `image` gefordert, aber auch für Applets (Java etc.) und andere programmierte Objekte.

[...] **Zeichnungen, die auf der Verwendung von Zeichen und Symbolen des ASCII-Codes basieren (ASCII-Zeichnungen),** [...]

Dies betrifft: streng genommen jede Form von ASCII-Art, also auch Smileys wie :-) – der gesunde Menschenverstand sagt einem aber, dass diese sicher nicht gemeint sind. »Zeichnungen, die auf der Verwendung von Zeichen und Symbolen des ASCII-Codes basieren« heißt natürlich nicht, dass ASCII-Art dann barrierefrei ist, sobald man eine andere Kodierung wie UTF-8 oder ISO 8859-1 verwendet. Gemeint sind hier eher waghalsige Konstruktionen aus den Urzeiten des Webs: damals beherrschten noch nicht alle Browser HTML-Tabellen und manche Anbieter versuchten diese Tabellen durch eine Folge von Zeichen wie _, l, - nachzubilden.

[...] **Frames** [...]

Dies betrifft: Frames. Der alternative Inhalt von Frames wird nicht in einem alt-Attribut, sondern im separaten `NOFRAMES`-Element gesetzt, das sowohl Teil eines Framesets sein kann als auch eines HTML 4 transitional-Dokumentes. Weitergehende Vorgaben zur Umsetzung von Frames werden weiter hinten bei Bedingung 12.2 gemacht.

[...] **Bilder, die als Punkte in Listen verwendet werden,** [...]

Dies betrifft: eine veraltete Technik, Listen mit künstlichen Einrückungen und Listenzeichen zu simulieren. Da die Verwendung der korrekten HTML-Listenelemente in Bedingung 3.6 verlangt wird ist dieser Punkt hier zu vernachlässigen.

[...] **Platzhalter-Graphiken,** [...]

Dies betrifft: sogenannte Spacer- oder Leer-GIF. Ebenfalls ein Relikt aus vergangenen Tagen, das aber leider immer noch häufig anzutreffen ist. Bei korrekter Anwendung von Cascading Style Sheets (CSS, s. Bedingung 3.3) obsolet.

[...] **graphische Buttons,** [...]

Dies betrifft: Formularelemente vom Typ `<input type="image">` und grafische Elemente innerhalb des `BUTTON`-Elements.

[...] **Töne (abgespielt mit oder ohne Einwirkung des Benutzers), Audio-Dateien, die für sich allein stehen, Tonspuren von Videos und Videos.** [...]

Dies betrifft: sämtliche Audiosignale, die bei der Betrachtung oder Bedienung einer Webseite oder -anwendung von Bedeutung sind und die echte Inhalte transportieren. Dies schliesst auch Podcasts ein, die ohne Mitschrift insbesondere für gehörlose Menschen unzugänglich sind.

Wie können Sie das testen?

In dieser Bedingung verstecken sich eine ganze Reihe von verschiedenen Formaten. Und wie so oft bei Tests auf BITV-Konformität hilft hier das Schweizer Taschenmesser des Webdesigns, die Web Developer Extension für Firefox. In dieser lassen sich sämtliche in BITV Bedingung 1.1 aufgelisteten Formate abschalten und durch die hinterlegten Textalternativen ersetzen.

Die Option zum Ausblenden von Bildern und zur Anzeige der Alternativtexte finden Sie in der Toolbar im Menü »Images«. Wählen Sie dort die Option »Replace Images With Alt Attributes«. Durch an- und abschalten dieser Option und Vergleich der sichtbaren Inhalte können Sie kontrollieren, ob sich ohne Bilder noch **der komplette Inhalt und alle Funktionen** der Seite erschließen. Sollte dies der Fall sein, so können Sie diesen Punkt abhaken. Sie sehen nämlich genau die Inhalte, die auch ein Screenreader oder Sprachbrowser vorlesen würde.

Hier wird deutlich, warum man dekorative Elemente oder sog. Spacer, die selbst keine Inhalte transportieren, mit einem leeren `alt=""` verstecken sollte. In diesem Fall versucht der Browser das Bild (bzw. den Platz, den es einnehmen würde) nämlich erst gar nicht zu rendern, so dass der Lesefluss nicht gestört wird. Auch hierzu gibt es den passenden Menübefehl in der Web Developer Toolbar: mit »Outline Images With Empty Alt Attributes« werden alle Bilder mit leerem `alt`-Attribut mit einem rotem Rahmen hervorgehoben.

Was ist wirklich Äquivalent?

Um es kurz zu fassen: so kurz wie möglich und so lang wie nötig. Daher sprechen die Web Content Accessibility Guidelines immer nur von wesentlichen Informationen und für den jeweiligen Kontext relevanten Bildinhalten, für die Alternativtexte hinterlegt werden müssen (»Text-Äquivalente müssen so geschrieben werden, dass sie alle wesentlichen Informationen vermitteln«).

Bei der Bewertung der Qualität von Alternativtexten ist der Kontext zu berücksichtigen – was ist im aktuellen Sinnzusammenhang wirklich nötig? Es ist möglich, dass ein und dasselbe Bild in unterschiedlichen Zusammenhängen einen Alternativtext benötigt oder ein `alt=""` angemessen ist.

Die bei Online-Ausgaben von Angeboten der klassischen Medien häufig anzutreffenden Bildunterschriften können eine äquivalente Alternative sein. Sie haben den Vorteil, dass sie für alle Nutzer wahrnehmbar sind und so auch an ihre individuellen

Bedürfnisse stark sehbehinderter Menschen anpassbar sind, die keinen Screenreader nutzen. In diesem Fall kann das `alt`-Attribut des mit einer Bildunterschrift versehenen Bildes auch leer bleiben, da eine Wiederholung des identischen Textes störend wirkt.

Positive Beispiele:

Ein einfach nachzuvollziehendes Beispiel für sinnvolle Alternativtexte ist das Logo der Aktion Mensch in der Fußzeile dieser Seiten. In grafischen Browsern erscheint dort der Hinweis »›Einfach für Alle‹ ist eine Initiative der« gefolgt von einer Grafik mit dem Logo der Aktion Mensch.

Üblicherweise würde der alternative Text für die Grafik nun heißen: »Logo der Aktion Mensch« – allerdings würde dann ein Screenreader genau das vorlesen: »›Einfach für Alle‹ ist eine Initiative der Logo der Aktion Mensch«.

Der folgende Screenshot zeigt die Darstellung des an Stelle dessen verwendeten alternativen Textes, der den tatsächlichen Inhalt des Logos, nämlich den Text »Aktion Mensch« wiedergibt. Hier wurde zur Kontrolle der Qualität der `alt`-Texte das Laden der Bilder im Browser unterbunden, das Ergebnis ist lesbar und diesmal auch sinnvoll.

In modernen `Browsern` können Sie die Schrifttype für die alternativen Texte per CSS bestimmen, so dass diese nicht in der voreingestellten Standardschrift erscheinen. Die einfache Anweisung `img{font-family:"Helvetica",sans-serif}` in Ihrem Style Sheet bringt den Browser auf den richtigen Kurs, so dass auch Besucher, die keine Bilder laden, in den vollen Genuss Ihrer Gestaltung kommen.

Negative Beispiele:

Überlange Alternativtexte

In der Liste der HTML 4-Attribute steht als Bedeutung des `alt`-Attributs »short description« – eine kurze Beschreibung – damit ist eigentlich alles Wesentliche gesagt. Wenn Sie eine längere Beschreibung hinterlegen müssen, so tun Sie dies im begleitenden Text oder benutzen Sie das hierfür vorgesehene `longdesc`-Attribut (Beispiel: `<img src="verkaufszahlen.png" alt="Verkaufszahlen 2006" longdesc="verkaufszahlen-2006.html">`).

Falsche Alternativtexte

Ein häufig anzutreffender Fehler bei der alternativen Betextung von Bedienelementen. Zur Verdeutlichung ein einfaches Beispiel: in einem Shopping-Angebot sind die Buttons grafisch umgesetzt und enthalten den Text »Bestellung abschicken«. Der Alternativtext des `INPUT`-Elementes ist davon abweichend »Hier Klicken, um zur Kasse zu gehen«.

Wenn ein Nutzer mit motorischer Behinderung, der auf eine Sprachsteuerung angewiesen ist, versucht, den Kaufvorgang abzuschliessen, so wird er an dieser Stelle mit einer Barriere konfrontiert. Er spricht den Text des Buttons (»Bestellung abschicken«), die Software zur Sprachsteuerung kann aber naturgemäß den Text des grafischen Buttons nicht erkennen und sucht nun im Quelltext der Seite vergebens nach dem passenden Text.

Alternativtexte sind übrigens auch kein guter Platz für Urheberrechtsvermerke (alt="© 2007") und ähnliches.

Linkziele im Alternativtext

Ebenfalls ein häufig anzutreffender Fehler. Vielfach wird bei verlinkten Bildern gleichzeitig auch das Linkziel im Alternativtext des Bildes beschrieben. Die Beschreibung des Linkziels ist jedoch (wenn überhaupt) eine Funktion des Links (also des `A`-Elements bzw. eines optionalen `title`-Attributs für den Link), *nicht* des Bildes. Der Vollständigkeit halber hier ein *korrektes* Beispiel: `<a href="http://www.aktion-mensch.de/" title="Zum Portal der Aktion Mensch"><img src="logo-am.png" alt="Aktion Mensch" width="96" height="25"></a>`.

Alternativtexte als Tooltip

Eng verwandt mit dem vorhergehenden Fehler und genauso falsch. Im Microsoft Internet Explorer werden die Werte des `alt`-Attributs als Tooltip über dem Bild dargestellt, wenn man mit der Maus darüberfährt. Dies ist streng genommen ein Fehler in diesem Browser – Alternativtexte dienen als *Alternative* zum Bild und **nicht** als Zusatzinformationen. Sie sollen nur

dargestellt werden, wenn der Benutzer das ursprüngliche Bild nicht sehen kann. Dieses Fehlverhalten lässt sich im IE übrigens abschalten, indem man ein leeres `title=""` definiert.

Bedingung 1.2: Textlinks für serverseitige Imagemaps (Prio. 1)

»Für jede aktive Region einer serverseitigen Imagemap sind redundante Texthyperlinks bereitzustellen« <zitat>

Was heißt das?

Die ursprünglich mit dieser Bedingung gemeinten serverseitigen Imagemaps waren kurzfristig gegen Mitte der neunziger Jahre des letzten Jahrhunderts in Mode, als manche Browser (z. B. Netscape 1.0) die gerade erfundenen clientseitigen Imagemaps noch nicht verstanden. Serverseitige Imagemaps können für `IMG` sowie `INPUT` mit dem Typ `"image"` definiert werden. Der Browser schickt die pixelgenauen Koordinaten eines Klicks auf diese Elemente zur weiteren Verarbeitung an den Server; dieser entscheidet dann mit einem serverseitigen Skript, welche Aktion auf diesen Klick folgen soll. Dies kann zum Beispiel der Aufruf einer neuen bzw. geänderten Seite oder die Übernahme von Formulardaten sein.

Diese einfachen serverseitigen Imagemaps sind kaum noch anzutreffen. Die Bedingung lässt sich jedoch so interpretieren, dass von ihr sämtliche Techniken erfasst werden, bei denen die weitere Verarbeitung des genauen Orts einer Interaktion eine Rolle spielt. In diesem Fall würde die Bedingung auch auf alle webbasierten Applikationen zutreffen, die in irgendeiner Form Landkarten oder andere Geodaten benutzen (wie Yahoo! Maps oder Google Maps). Da in Zeiten von AJAX & Co. jedoch nicht mehr trennscharf zu unterscheiden ist, ob es sich um eine server- oder eine clientseitige Applikation handelt haben wir die Diskussion zu diesem Punkt der BITV in Bedingung 1.5 (clientseitige Imagemaps) zusammengefasst.

Bedingung 1.3: Audiodeskription für Videos (Prio. 1)

»Für Multimedia-Präsentationen ist eine Audio-Beschreibung der wichtigen Informationen der Videospur bereitzustellen.« <zitat>

Was heißt das?

Audio-Beschreibungen, oder um beim eigentlichen Fachterminus zu bleiben, Audiodeskriptionen sollten angeboten werden, wenn sich Sinnzusammenhänge und Inhalte nicht alleine aus den akustischen Informationen eines Videos erschließen. In dieser Bedingung geht es im Gegensatz zur folgenden Bedingung 1.4 ausschliesslich um stark sehbehinderte und blinde Nutzer, denen unter Umständen wesentliche Informationen vorenthalten bleiben, wenn diese rein visuell dargestellt werden.

Beispiele:

In den klassischen Medien werden Audiodeskriptionen auf einer zusätzlichen Tonspur z. B. in Spielfilmen eingesetzt, um szenische oder atmosphärische Informationen zu transportieren. Ein Erzähler führt den Hörer durch den Film und versorgt ihn mit Informationen, die bei rein auditiver Wahrnehmung verborgen blieben. Die Schwierigkeit besteht hier, diese so mit den bewegten Bilder zu synchronisieren, dass eventuell gesprochener Text nicht überlagert wird. Die weitaus meisten Videos im Web fallen sicher nicht in die Kategorie »abendfüllender Spielfilm«, sondern sind eher kurze Stücke, in denen üblicherweise alles wesentliche gesagt wird – diese Videos sind somit zumindest für Blinde und Sehbehinderte allgemein gut zugänglich.

Anders sieht es im Bereich der webbasierten Lerninhalte (e-Learning) aus. Immer mehr Universitäten bieten Ihre Vorlesungen auch im Netz an. Wenn nun z. B. Vortragsfolien, Animationen, aber auch Textinhalte in der Multimedia-Präsentation so umgesetzt sind, dass die Inhalte zwar zu sehen, aber nicht zu hören sind, besteht Handlungsbedarf im Sinne dieser Bedingung.

Wie können Sie das testen?

Die Art des Tests hängt von den jeweiligen Inhalten und deren Quellen, der verwendeten Technik und dem verwendeten Format bzw. Abspielprogramm (*Player*) ab. Die separate Tonspur ist in vielen Playern optional an- und abschaltbar, diese setzen jedoch meist auf eine proprietäre Technik und schliessen sich gegenseitig aus. Das wiederum vom W3C standardisierte SMIL-Format für die Integration von synchronisiertem Multimedia wird nur von einem Teil der Player unterstützt. Microsoft hingegen hat mit SAMI eine eigene Spezifikation zur Erstellung von Audiodeskriptionen und Untertiteln eingeführt.

Ein Problem besteht in der Abgrenzung der Inhalte und der Definition des Begriffs *»Videospur«*. Einige Video- und Multimediaformate können beliebige andere Quellformate einbetten; so können in QuickTime HTML- oder RTF-Texte geladen werden, in einen Windows Media Stream können Powerpoint-Präsentationen aus externen Quellen eingebettet werden, und in beiden Formaten kann die Synchronisation und Interaktion über Skripte und Sprungmarken gesteuert werden. Der RealPlayer nutzt hierfür den SMIL-Standard des W3C, diese Dateien sind dann auch mit dem Windows Media Player ab Internet Explorer 5.5 benutzbar.

Ob und wie man an die Audiodeskriptionen herankommt ist von Player zu Player unterschiedlich; in Apples QuickTime Player sind diese nur in der kostenpflichtigen Pro-Version verfügbar. Dazu kommen die Möglichkeiten von Flash, das sowohl als einfacher Container für Videos dienen kann, aber auch eigenständige multimediale Inhalte ermöglicht, bei deren Gestaltung keine Grenzen gesetzt sind. Aufgrund der Vielzahl von Variablen ist es kaum möglich, eine knappe und bündige Testanleitung zu schreiben.

Bedingung 1.4: Multimedia mit Untertiteln (Prio. 1)

<zitat>

»Für jede zeitgesteuerte Multimedia-Präsentation (insbesondere Film oder Animation) sind äquivalente Alternativen (z. B. Untertitel oder Audiobeschreibungen der Videospur) mit der Präsentation zu synchronisieren.«

Was heißt das?

In Ergänzung zur vorhergehenden Bedingung hier eine zweiteilige Vorgabe zur Synchronisation, d. h. zum zeitgleichen Erscheinen von Alternativen zu den angebotenen visuellen und akustischen Inhalten. Zum einen betrifft die Bedingung wie schon in 1.3 blinde und sehbehinderte Menschen, aber zusätzlich auch Menschen mit Lernbehinderung und spezifischen Leseschwächen. Im Gegensatz zu 1.3 ist hier von Bedeutung, ob der *Zeitpunkt* des Auftretens einer Information relevant ist – in diesem Fall *müssen* die Alternativen synchronisiert werden, sonst gilt 1.3.

Zum anderen betrifft diese Bedingung durch die Forderung nach Untertiteln gehörlose oder schwerhörige Menschen, die zum Teil auf eine schriftliche Alternative zu den akustischen Informationen angewiesen sind. Sie können sich beim Einsatz von Audio–Elementen (so zum Beispiel in Flash) nicht darauf verlassen, dass am Zielrechner ein Lautsprecher vorhanden ist oder ob die Anwendung vielleicht gerade von einem gehörlosen oder schwerhörigen Menschen genutzt wird. Wenn Sie wichtige Inhalte mittels Audio transportieren, sollten Sie auf jeden Fall Textalternativen anbieten, um Menschen mit Hörbehinderungen den vollen Zugang zu ermöglichen.

Bei click.tv finden Sie eine Reihe Videos, in denen man anhand des annotierten Textes zu bestimmten Ausschnitten springen kann.

Hierbei ist jedoch zwischen schwerhörigen und gehörlosen Nutzern deutlich abzugrenzen. Für schwerhörige Menschen sind zusätzliche, optional einschaltbare Transskripte oder Untertitel von multimedialen Inhalten sicher von Vorteil. Für gehörlose Menschen, die in Deutscher Gebärdensprache (DGS) kommunizieren, reicht diese Methode nicht aus, um einen gleichwertigen Zugang zu den Informationen zu gewährleisten.

Da DGS die primäre Sprache vieler gehörloser Menschen ist, bleibt die Schriftsprache die erste erlernte ›Fremdsprache‹ und ist keine »äquivalente Alternative« im Sinne der Bedingung 1.4. Sinnvoller als eine Untertitelung ist hier die Übersetzung in DGS. In erster Linie geschieht dies mit Gebärdendolmetschern oder gehörlosen Darstellern, in Einzelfällen werden auch Avatare eingesetzt.

Diese Programme zur automatischen Generierung eines computergenerierten Darstellers sind für die Übersetzung einfacher Sachverhalte geeignet und lassen sich zudem einfach mit Untertiteln synchronisieren. Entsprechende Beispiele finden Sie unter gebaerden.hamburg.de. Die Übersetzung komplexerer Informationen in computergenerierte Gebärdensprache mit Programmen wie Sign Smith Studio erfordert jedoch einen enormen Aufand, sodaß echte Darsteller auf absehbare Zeit sicher die bessere Alternative sind.

Beispiele:

Eine Auswahl von untertitelten Videos verschiedenster Art finden Sie bei Google Video. Eine Übersicht vorbildlicher Videos in Gebärdensprache finden Sie beim Deutschen Gehörlosen Bund im Menüpunkt «Gebärdensprach-Filme«. Sehr gut gemacht sind auch die DGS-Videos, mit denen die Deutsche Bundesbank Themen wie Geldpolitik, die Geschichte des Geldes oder die Euro-Einführung erklärt.

Werkzeuge:

- Captionate ist eine Windows-Anwendung, mit der Untertitel in Flash Video-Dateien (FLV) eingebettet werden können. Das Erscheinungsbild ist über FLVPlayback Skins anpassbar.
- Hi-Caption Studio ist ein Werkzeug zur Erstellung von SAMI-Dateien mit Untertiteln für Windows Media Player und RealPlayer
- Media Access Generator (MAGpie) hilft bei der Erstellung von Untertiteln und Audiodeskriptionen. Die Java-Anwendung unterstützt QuickTime, RealPlayer und Windows Media Player
- Die Firma Apple hat eine ganze Reihe von Tutorials zur Erstellung von SMIL-Dateien für QuickTime veröffentlicht.

Bedingung 1.5: Textlinks für clientseitige Imagemaps (Prio. 2)

»Für jede aktive Region einer clientseitigen Imagemap sind redundante Texthyperlinks bereitzustellen.« <zitat>

Was heißt das?

Wie bei den in Bedingung 1.2 genannten serverseitigen Imagemaps geht es auch hier um die Verarbeitung einer Interaktion mit pixelgenauen Koordinaten bei `IMG`-Elementen. In dieser Bedingung geht es um die *clientseitigen* Imagemaps, die unmittelbar im Browser ausgewertet werden, ohne dass dazu ein entfernter Server befragt werden muss. Der Browser entscheidet aufgrund der im HTML der Seite oder Anwendung per `MAP` hinterlegten Koordinaten (`AREA`), welche Aktion auf einen Klick folgen soll.

Im Gegensatz zu den serverseitigen Varianten bieten clientseitige Imagemaps aus Sicht der Barrierefreiheit den Vorteil, per Tastatur bedienbar zu sein, da die aktivierbaren Regionen einer Grafik oder eines Bildes echte, im HTML-Dokument enthaltene Hyperlinks sind. Um jedoch in den unterscheidlichen Ausgabeformen wahrnehm- und bedienbar zu sein, benötigt man zusätzliche Angaben, *was* in der ausgewählten Region dargestellt wird, bevor der Nutzer den Link aktiviert. Hierzu ist in HTML das `alt`-Attribut für das Element `AREA` zwingend vorgeschrieben.

Da man jedoch nicht davon ausgehen kann, dass alle Nutzer einen direkten Zugriff auf diese (versteckten) Angaben haben, verlangt die BITV hier die Bereitstellung von redundanten Texthyperlinks. Hierdurch soll sichergestellt werden, dass die Funktionen auch dann verständlich und bedienbar sind, wenn eine unmittelbare Interaktion mit der Imagemap selbst nicht möglich ist, da zum Beispiel ein User Agent die Alternativtexte der aktiven Regionen (vgl. Bedingung 1.1) nicht ausgibt.

Beispiele:

Ein einfaches Beispiel ist eine Weltkarte, in der die einzelnen Kontinente als aktive Links definiert sind. Hierbei gibt es einen begrenzten Satz an Auswahlmöglichkeiten (nämlich genau sieben) – diese lassen sich ohne weiteres als zusätzliche Textlinks in der Nähe der Weltkarte hinterlegen.

Wenn wir in dieser Karte auf Länderebene hinuntergehen, so stoßen wir schnell an die Grenzen des Konzept der redundanten Texthyperlinks:

- Die Vereinten Nationen haben zurzeit 192 Mitgliedsländer – diese müssten neben einer klickbaren Imagemap allesamt in einer zusätzlichen Liste verzeichnet werden.
- Die Kartenausschnitte der bekannten Anwendung Google Maps belegen bei einer Fenstergröße von 800 × 600 Pixel eine Fläche von ca. 450 × 500 Pixel. Da jedes dieser Pixel eine aktivierbare Region darstellt, verfügt eine solche Karte über 225.000 »aktive Regionen«. Zur Erfüllung der Bedingung 1.5 müssten diese allesamt redundant nachgebildet werden – ein schier unmögliches Unterfangen, dass zudem keinem Nutzer helfen würde.

Bei komplexen Imagemaps, insbesondere wenn es sich um Landkarten handelt, ist diese Bedingung der BITV nur sehr begrenzt erfüllbar. So meint das eigentlich zur strikten Einhaltung der BITV verpflichtete Bundesamt für Kartographie und Geodäsie (BKG) zum Thema Barrierefreiheit: »Bei der Erstellung dieser Seiten wurden die Vorgaben der BITV […] so weit wie möglich umgesetzt. Grenzen der Barrierefreiheit liegen beispielsweise bei der Darstellung von Landkarten.«

Wie können Sie das testen?

Ohne den Blick in den Quelltext oder das Durchklicken einer Seite lässt sich kaum feststellen, ob Imagemaps Verwendung finden. Hier hilft die Web Developer Toolbar für Firefox: im Menü »Outline« finden Sie den Befehl »Outline Custom Elements« – wenn Sie hier `MAP` eintragen, werden Imagemaps mit einer frei definierbaren Farbe umrandet und sind so leichter aufzufinden. So lässt sich überprüfen, ob es zu den Regionen der Imagemap zusätzliche Texthyperlinks gibt. Übrigens können alle diese Einstellungen dauerhaft vorgenommen werden, sodaß Sie sich in der Toolbar ihre eigene Testumgebung konfigurieren können, die dann auf alle neu geladenen Seiten angewendet wird.

Anforderung 2: Farben und Kontraste

»Texte und Graphiken müssen auch dann verständlich sein, wenn sie ohne Farbe betrachtet werden.« <zitat>

Was heißt das?

Ein Teil Ihres Publikums kann mit der Anweisung »Zum Bestellen drücken Sie bitte auf den grünen Knopf« unter Umständen nichts anfangen. Diese (zum überwiegenden Teil übrigens männlichen) Nutzer sind farbfehlsichtig, das heißt sie verfügen über eine ungenügende Differenzierung von bestimmten Farben. Welche Farben bzw. Farbkombinationen dies sind, hängt von der Art der Farbfehlsichtigkeit ab, da es mehrere, höchst unterschiedliche Varianten gibt.

Diese Bedingung betrifft noch weitere Benutzergruppen: viele sehbehinderte Menschen haben zur Vermeidung von Blendeffekten benutzerdefinierte Farben eingestellt, die den Browser veranlassen, sämtliche Farbangaben in Ihrem CSS zu ignorieren. Blinde und viele sehbehinderte Menschen können ebenfalls mit Informationen nichts anfangen, die ausschließlich über Farbe transportiert werden (Beispiel: »Bitte beachten Sie den Rot hinterlegten Text«). Mehr dazu in den einzelnen Bedingungen:

Bedingung 2.1: Auch ohne Farben nutzbar (Prio. 1)

»Alle mit Farbe dargestellten Informationen müssen auch ohne Farbe verfügbar sein, z. B. durch den Kontext oder die hierfür vorgesehenen Elemente der verwendeten Markup–Sprache.« <zitat>

Was heißt das?

Die Formulierung »ohne Farbe verfügbar« sollte präzisiert werden. Treffender ist die Formulierung in den WCAG 1.0: »Verlassen Sie sich nicht auf Farbe allein«. Farben sind kein generelles Problem, sondern es bedarf zusätzlicher Mittel, um farblich dargestellte Informationen für alle Nutzergruppen zugänglich zu gestalten.

Diese Bedingung ist gleich auf eine ganze Reihe möglicher Fallstricke im Webdesign anwendbar. So könnte es sein, dass der Autor einer Seite die Überschriften als normale Textabsätze angelegt hat und lediglich per CSS, oder, behüte, per `FONT color` eingefärbt hat, statt die hierfür vorgesehenen Überschriftenelemente aus HTML zu nutzen (und diese dann per CSS zu formatieren).

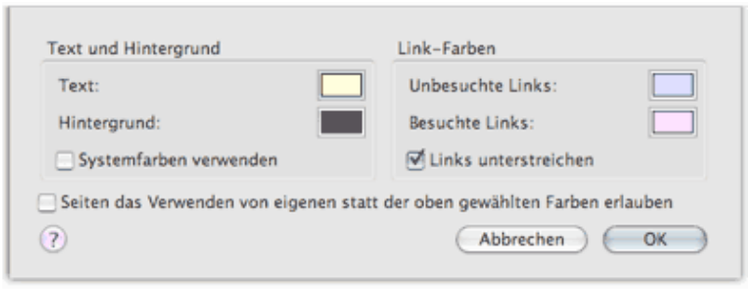
Wenn Links im Fließtext nicht unterstrichen sind, sondern sich nur durch die Farbe vom umgebenden Text unterscheiden, ist diese Bedingung nicht erfüllt. Formulare, in denen Pflichtfelder oder Fehleingaben ausschließlich fabrig gekennzeichnet sind müssen mit weiteren grafischen Elementen hervorgehoben oder im Text beschrieben werden. Diagramme, in denen Kurven oder Tortenstücke nur anhand ihrer Farben zu unterscheiden sind, können durch zusätzliche Schraffuren, unterscheidliche Linienarten und Strichstärken nutzbarer gestaltet werden.

Wie können Sie das testen?

Die Bedingung 1.2 lässt sich nicht maschinell prüfen, eine Sichtkontrolle ist notwendig. Zum einen muss der Text der Website oder Anwendung auf verdächtige Formulierungen abgesucht werden (Beispiel: Bestätigen Sie mit dem grünen Button), zum anderen müssen die Gestaltung und grafische Elemente visuell überprüft werden.

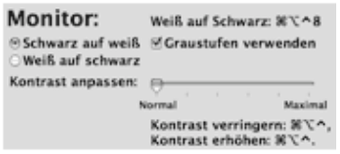
Es gibt spezialisierte Werkzeuge, die beim Aufspüren bestimmter Probleme helfen können; für manche Tests muss auch der Computer entsprechend bestimmter typischer Einstellungen sehbehinderter Menschen umkonfiguriert werden. Dabei muss klar sein, dass es sich immer nur um eine Annäherung handeln kann, da es eine große Vielfalt der individuellen Einstellungen gibt.

Einen ersten Kurztest können Sie mit Firefox vornehmen, indem Sie in den Einstellungen unter ›Inhalt‹ → ›Farben‹ ihr eigenes Farbschema definieren und Seiten untersagen, ihre eigenen Farben zu verwenden. Im folgenden Screenshot sehen Sie eine mögliche Einstellung mit dunkler Hintergrundfarbe und hellgelbem Text sowie angepassten Linkfarben:



Laden Sie nun Ihre Website neu. Wenn sie jetzt noch alle Inhalte wahrnehmen können und alle Funktionen ausführen können, haben Sie diesen Test schon fast bestanden. Der Vollständigkeit halber sollten Sie noch in der Web Developer Toolbar das Laden von CSS unterbinden. So können Sie auf einen Blick feststellen, ob für die Struktur der Seite die korrekten

Strukturelemente verwendet wurden oder ob diese doch nur mit `<div class="ueberschrift gross rot">` ausgezeichnet wurden.



Realistischer ist der Test mit einem der Kontrastschemata, die in Oberflächen wie Windows, Gnome oder Mac OS X frei einstellbar sind. Sie können davon ausgehen, daß stark sehbehinderte Menschen ihren Rechner an ihre ganz persönlichen Bedürfnisse angepasst haben, und dazu gehört mehr als nur der Browser. In den Systemeinstellungen von Windows lassen sich verschiedene Farb- und Kontrastschemata einstellen, die eine mit vielen Formen der Sehbehinderung verbundene Blendempfindlichkeit verhindern sollen. Mit einigen dieser Einstellungen wird das gesamte Betriebssystem invertiert dargestellt. Der Internet Explorer ignoriert unter Windows dann sämtliche per CSS als Hintergrundbilder realisierten Inhalte einer Webseite.

Weitere Informationen hierzu finden Sie in unserem Artikel »Image Replacement- Techniken nicht zugänglich für Sehbehinderte«. Eng verwandt ist die Problematik, dass freigestellter Text in Grafiken vor wechselnden Hintergründen unter Umständen nicht mehr erkennbar ist. Besonders in GIF- und PNG-Dateien mit Transparenzen kann die Farbinvertierung des Betriebssystems oder Browsers dazu führen, dass zum Beispiel freigestellter schwarze Texte oder grafische Elemente auf schwarzem Hintergrund stehen.

Der Prüfschritt zu Bedingung 2.1 könnte umformuliert heißen: »Testen Sie, ob sämtliche Inhalte auch bei benutzerdefinierten Farben wahrnehmbar sind.«

Wenn wesentliche Inhalte eines Angebots in anderen Formaten wie Flash oder PDF umgesetzt sind, dann ist diese Bedingung natürlich auch auf diese Inhalte anzuwenden, d. h. diese müssen in einen Test einbezogen werden. Die Möglichkeiten von Flash gehen mittlerweile über das hinaus, was man mit herkömmlichem HTML, CSS und JavaScript erreichen kann. So kann man in Flash nicht nur abfragen, ob auf dem Host-System ein Screenreader oder andere Hilfsmittel aktiv sind, sondern auch feststellen, ob der Nutzer unter Windows eines der Kontrastschemata zur Anzeige eingestellt hat. Prinzipiell ist dies also möglich und sollte dementsprechend in der Evaluierung einer Website bewertet werden.

Bedingung 2.2: Ausreichende Kontraste von Grafiken (Prio. 1)

»Bilder sind so zu gestalten, dass die Kombinationen aus Vordergrund- und Hintergrundfarbe auf einem Schwarz-Weiß-Bildschirm und bei der Betrachtung durch Menschen mit Farbfehlsichtigkeiten ausreichend kontrastieren.«

Was heißt das?

An der Formulierung kann man deutlich das Alter der Vorlage für die BITV, der Web Content Accessibility Guidelines, ablesen: Schwarz-Weiß-Bildschirme sind heutzutage wohl kaum noch im Einsatz und wenn, dann eher nicht an einen Rechner mit Internetzugang angeschlossen. Das ist aber auch nicht die Kernaussage dieser Bedingung, sondern sie ergänzt die vorhergehende Bedingung 2.1: Sie können sich *nicht* drauf verlassen, dass Farben ihren gedachten Zweck erfüllen.

Die Bedingungen 2.1 und 2.2 sind nach unserer Auffassung in der BITV sinnvoll priorisiert. Da Benutzer bei korrekter Anwendung von HTML & CSS weitgehende Freiheiten haben, die Präsentation von Inhalten an ihre speziellen Bedürfnisse anzupassen, sind die Vorgaben für Texte folgerichtig bei der Priorität 2 einsortiert. Da diese Möglichkeiten jedoch nicht ohne weiteres für eingebettete Bilder, Grafiken etc. bestehen ist es schlüssig, diese Formate mit höherer Priorität zu behandeln.

Da sich die Ursachen, Wirkungen und Tests jedoch kaum unterscheiden haben wir diesen Punkt in der verwandten Bedingung 2.3 zusammengefasst.

Bedingung 2.3: Wahrnehmung mit Farbfehlsichtigkeiten (Prio. 2)

»Texte sind so zu gestalten, dass die Kombinationen aus Vordergrund- und Hintergrundfarbe auf einem Schwarz-Weiß-Bildschirm und bei der Betrachtung durch Menschen mit Farbfehlsichtigkeiten ausreichend kontrastieren.«

Was heißt das?

Ignorieren wir den Schwarz-Weiß-Bildschirm und wenden uns dem Kern der Bedingung zu: wie in der Einleitung zu BITV-Anforderung 2 erwähnt sind ca. 8–10 Prozent der männlichen Bevölkerung farbfehlsichtig; bei Frauen ist der Anteil hingegen wesentlich geringer. Für diese Nutzer ist wichtig, dass Farben, die dem Verständnis oder der Bedienung dienen, ausreichend wahrgenommen und unterschieden werden können. Auch wenn es bestimmte Bereiche des Spektrums und gewisse Farbkombinationen gibt, die für Farbfehlsichtige grundsätzlich

problematisch sind, so sind diese nicht grundsätzlich zu vermeiden. Diese Bedingung bedeutet nicht, dass nur noch Layouts in Schwarz und Weiss möglich sind – im Gegenteil.

Die Frage ist hier, ob es zu Fehlbedienungen kommen kann oder ob Informationen verloren gehen, wenn bestimmte Farben oder Farbkombinationen nicht wahrgenommen werden können. Daher sollten Sie sich bei diesem Prüfschritt nicht auf Testwerkzeuge verlassen, die lediglich das Vorhandensein bestimmter Farben oder Farbkombinationen überprüfen, sondern die Farben immer in ihrem Kontext bewerten.

Welche der Farben denn problematisch ist wird Ihnen niemand mit absoluter Gewissheit sagen können, da man nicht messen kann, was Farbfehlsichtige tatsächlich sehen. Daher sind Empfehlungen oder sogar Berechnungen von Schwellwerten, ab denen diese Bedingung erfüllt ist, mit Vorsicht zu genießen. Wenn Sie weiter in diese Thematik einsteigen wollen empfehlen wir Ihnen hierzu das Buch *»Building Accessible Websites«* von Joe Clark, das auch als HTML-Version veröffentlicht wurde. In Kapitel 9 »Type and Colour« werden die verschiedenen Formen der Farbfehlsichtigkeit besprochen.

Komplementärkontraste sind grundsätzlich problematisch, besonders wenn es um grössere nebeneinander liegende Flächen geht. Hierunter versteht man Farben, die sich in einem Farbkreis genau gegenüberliegen und aufgrund der unterschiedlichen Wellenlängen ein dauerndes Umfokussieren des Auges verursachen. Dies führt zu dem bekannten Flimmern und Farbsäumen, die man zum Beispiel bei blauem Text auf orangem Hintergrund beobachten kann. In Halbtonbildern wie z. B. Fotos sind diese naturgemäß nicht zu vermeiden und sind somit auch nicht Gegenstand dieser Bedingung.

Wie schon in den vorhergehenden Bedingungen müssen hier noch weitere Punkte abgeprüft werden, da unzureichende Kontraste auch Nutzer betreffen, die Schwierigkeiten in der Erkennung subtiler Farb- oder Helligkeitsunterschiede haben. Mit zunehmenden Alter nimmt die Fähigkeit ab, schwache Farb- und Helligkeitskontraste zu differenzieren, die für jüngere Nutzer vollkommen unproblematisch sind.

Wie können Sie das testen?

Ein Teil dieser Bedingung ist mit den gängigen Mitteln der heutigen Betriebssysteme relativ einfach zu testen. Mac OS X lässt sich über die Systemeinstellung ›Bedienungshilfen‹ komplett auf Graustufen umstellen. Sie können auch einen Screenshot Ihrer Website in einem Grafikprogramm wie Photoshop in Graustufen konvertieren. Diese Methoden sind jedoch nur als erste grobe Tests zu verstehen und bieten keine verlässliche Aussage über tatsächlich problematische Farbkontraste.

Genauer als die reine Konvertierung nach Graustufen ist die Simulation von Farbfehlsichtigkeiten, wie sie bei www.vischeck.com möglich ist. Dieses kostenlose Online-Tool erstellt eine Grafikdatei Ihrer Website mit den Farben, wie sie Besucher mit den gängigsten Farbfehlsichtigkeiten sehen würde.

Noch genauer ist die Konvertierung, die unter colorfilter.wickline.org angeboten wird. Nach Eingabe der Adresse Ihrer Seite bekommen Sie keine Grafikdateien, sondern umgefärbte HTML-Seiten zurück und haben die Wahl zwischen einer Vielzahl unterschiedlichster Farbfehlsichtigkeiten.

Tipp: Beachten Sie, dass alle diese Tests, insbesondere die visuellen Überprüfungen in einer Position etwa eine Armlänge vom Monitor entfernt durchgeführt werden sollten (Normalsichtigkeit vorausgesetzt). Wenn Sie Inhalte nur noch erkennen können, indem Sie sich näher zum Monitor hinbewegen, werden dies auch viele Ihrer Benutzer machen müssen. Das bedeutet, dass Ihr Design zu kontrastarm ist oder die Inhalte zu filigran, um in normaler Sitzposition noch wahrgenommen zu werden. Besonders hilfreich sind Benutzertests mit Menschen mit Farbfehlsichtigkeiten.

Idealerweise führen Sie solche Tests bereits in der Designphase einer Website, also zu dem Zeitpunkt, wenn das Farbschema mit dem Kunden abgestimmt wird, durch. Mit CSS lassen sich Farbänderungen zwar in Sekundenschnelle durchführen, allerdings wird der Abstimmungsprozess mit dem Kunden verlängert. Und nichts ist schlimmer, als wenn Fehler erst nach dem Launch vom Kunden und dessen Besuchern gefunden werden. Also: frühzeitig testen!

Probleme bei der Bewertung

Spannend ist die Frage, ab welchem Wert Kontraste als ausreichend gelten können. Die BITV schweigt sich hier aus, da auch in den WCAG 1.0 eigentlich nur erklärt wird, wie man in CSS Farben festlegt, aber nicht welche man nehmen sollte und welche nicht.

Ein bekanntes Testwerkzeug für diese Bedingung ist der Colour Contrast Analyser des Web Accessibility Tools Consortiums, das auch in einer deutschen Übersetzung veröffentlicht wurde. Die Anwendung basiert auf einem vorgeschlagenen Algorithmus der nach wie vor in der Entwicklung befindlichen WCAG 2.0. So lange die neuen Richtlinien nicht fertiggestellt sind sollten die Ergebnisse mit Vorsicht behandelt werden.

Es ist fast unmöglich, Farbkontraste von geglätteten Schriften auf hochwertigen LC-Displays mit Subpixel-Rendering und Antialiasing verlässlich zu messen. Daher sollten sie sich *nicht* auf die Meßergebnisse von Werkzeugen wie der Firefox-Erweiterung ColorZilla verlassen.

Zur genauen Messung von Texten müssten Sie unter Umständen die Schriftglättung abschalten und den Text um ein vielfaches vergrößern. Das ist auch nicht unbedingt realistisch, da auf heutigen Computern die Schriftglättung ab Werk eingeschaltet ist und die meisten Nutzer, also auch die von dieser Bedingung Betroffenen, das Web ganz anders wahrnehmen als in diesem Testszenario.

Die bekannten Algorithmen ziehen nicht in Betracht, dass die möglichen Kontrastumfänge bei LC-Displays von 150:1 bei billigen Modellen bis zu 450:1 bei hochwertigen, bei Röhrenmonitoren von 350:1 bis 700:1 reichen, von unterscheidlichen Farbumfängen und Kalibrierungen ganz zu schweigen. Die subjektiv empfundenen Werte werden also höchst unterscheidlich sein und nicht unbedingt mit den Meßergebnissen übereinstimmen, zumal benachbarte Gestaltungselemente einen erheblichen Einfluß auf die subjektiv wahrgenommenen Kontraste haben können.

Ein weiteres Problem der für die WCAG 2.0 vorgeschlagenen Berechnungsmethode ist, dass sie zwar einen *Mindest*wert für Kontraste festlegt, jedoch keinen *Maximal*wert. Nach WCAG 2.0 erfüllt man die Priorität 2 mit einem Mindestkontrast, für einen noch höheren Kontrast erhält man die Höchstpunktzahl der Priorität 3. Eine maximale Obergrenze für Kontraste wird hingegen nicht gesetzt, wobei gerade für Menschen mit bestimmten Leseschwächen ein zu hoher Kontrast negative Auswirkungen auf die Lesbarkeit und das Textverständnis haben kann.

Anforderung 3: Sauberer Code

»Markup–Sprachen (insbesondere HTML) und Style Sheets sind entsprechend ihrer Spezifikationen und formalen Definitionen zu verwenden.« <zitat>

Was heißt das?

Sauberer Code legt ein wichtiges Fundament für die Vermeidung oder Beseitigung von Barrieren. Die Validierung der Dokumente schafft eine solide Basis für jedes Webangebot. Die saubere Trennung von Gestaltung und Inhalt verringert die Ladezeit beim Benutzer, Ihre Seiten werden offen für alle möglichen Ausgabeformen.

Die Zeiten, wo proprietäres Markup nötig war, um eine Webseite überhaupt lauffähig zu machen, sind längst vorbei. Denn auch die Browserhersteller haben den Trend erkannt und bringen seit Jahren nur noch Produkte auf den Markt, die dem Ideal der vollständigen Unterstützung der Standards immer näher kommen. Was es zu beachten gilt lesen Sie in den einzelnen Bedingungen:

Bedingung 3.1: Verzicht auf Schriftgrafiken (Prio. 1)

»Soweit eine angemessene Markup-Sprache existiert, ist diese anstelle von Bildern zu verwenden, um Informationen darzustellen.« <zitat>

Was heißt das?

Viele Texte verstecken sich in Schriftgrafiken: 9px großer Text in Navigationen, grafische Überschriften und sogar Fließtexte, aber auch Kleingedrucktes und die sogenannten CAPTCHAs. Für viele dieser Inhalte bieten CSS & HTML einen adäquaten Ersatz, wenn Webdesigner sie bestimmungsgemäß und bis zur Grenze des Machbaren einsetzen.

Die Umsetzung von Texten als echte Texte und nicht als Raster-oder Vektorgrafiken führt dazu, dass diese maschinell zu verarbeiten sind und für die Hilfsmittel von Menschen mit Behinderung direkt zugänglich sind, ohne dass man auf Alternativen zurückgreifen muss. Diese Texte lassen sich an individuelle Einstellungen von Farben (s. Bedingung 2.1) oder Schriftgrößen (s. Bedingung 3.4) anpassen, was mit Text in Rastergrafiken (.gif, .png oder .jpg) nicht ohne weiteres möglich ist.

Es gibt mittlerweile viele Markup-Sprachen: mit SVG, MathML oder CML lassen sich unterschiedlichste Einsatzgebiete abdecken. **Nur:** ab wann ist ihre Nutzng angemessen? Wenn sie von ein oder zwei exotischen User Agents unterstützt werden? In Bereichen, die sich an ein Fachpublikum wenden, kann man sicher erwarten, dass diese über eine entsprechend spezialisierte Software verfügen. Zumindest sollte eine hohe Bereitschaft vorausgesetzt werden können, diese zu installieren. Was ist aber, wenn es wie im Fall von SVG wesentlich universeller einsetzbare Alternativen wie Flash gibt, die nach heutigem Kenntnisstand ebenfalls barrierefrei eingesetzt werden können – die aber auch bei liberaler Auslegung keine Markup-Sprachen sind?

Wie können Sie das testen?

Hier hilft das Gelernte von Anforderung 1: Schalten Sie sämtliche Inhalte einer Seite ab, die nicht (X)HTML sind und überprüfen Sie, ob der Sinn und Zweck der Seite noch transportiert wird und der vollständige Inhalt noch vorhanden und wahrnehmbar ist und alle Funktionen bedienbar sind. Dies betrifft nicht nur Techniken wie Java, Flash oder JavaScript, auch in Bildern können sich wesentliche Inhalte verstecken, die sich **ohne** diese Bilder nicht mehr erschließen lassen, wenn keine adäquaten Alternativen bereitgestellt werden.

Bitte beachten Sie: um bewerten zu können, ob man ein vergleichbares Resultat mit HTML und CSS statt eines Bildes erreicht hätte sind mehr als Grundkenntnisse in diesen Sprachen unerlässlich.

Bei einer Sichtkontrolle ist es unter Umständen nicht möglich, auf Anhieb zwischen echtem Text und Schriftgrafiken zu unterscheiden. Auch hier helfen Werkzeuge wie WAVE oder die schon öfter erwähnte Web Developer Toolbar: der Befehl ›Outline Images‹ hebt sämtliche Bilder einer Seite hervor, ›Disable Images‹ hingegen lässt sie verschwinden. Der Befehl ›View Image Information‹ wiederum öffnet eine neue Seite mit einer tabellarischen Übersicht sämtlicher Bilder einer Seite inklusive ihrer Alternativtexte.

Bedingung 3.2: Valides Markup (Prio. 1)

»Mittels Markup-Sprachen geschaffene Dokumente sind so zu erstellen und zu deklarieren, dass sie gegen veröffentlichte formale Grammatiken validieren.«

Was heißt das?

Eine kurze Anforderung, die es in sich hat. Dies bedeutet, dass Ihre Inhalte mit gültigem, sauberem Code nach den jeweils gültigen Empfehlungen des W3C aufgebaut sein müssen. *Ohne wenn und aber.*

Das beginnt bei (X)HTML-Dokumenten bereits mit der sogenannten Document Type Definition (*DTD*), nach der sich ein HTML-Dokument als solches zu erkennen gibt – das ist mit »… und zu deklarieren …« gemeint. Es reicht nicht, HTML–Dokumente mit der Dateiendung .html zu speichern und den Rest dem Browser zu überlassen. Im Dokument steht als Erstes, nach welcher der verschiedenen (X)HTML-Geschmacksrichtungen sich Ihr Dokument richtet. Dies hilft bei der Fehlersuche, wenn sie Ihre Seiten durch eines der verschiedenen Prüfprogramme wie z. B. validator.w3.org oder Validome kontrollieren lassen.

Eine korrekte DTD bringt moderne Browser dazu, sich an die Regeln zu halten und die Angaben z. B. des CSS so umzusetzen, wie vom Seitenbauer gewünscht. Verlassen Sie sich nicht auf die DTDs, die von manchen Programmen eingesetzt werden – diese sind oftmals fehlerhaft. Eine Liste der korrekten DTDs finden Sie beim W3C.

Auch der Rest des HTML-Codes muss bestimmten Qualitätskriterien genügen. Wie Sie diese überprüfen können, erfahren Sie im nächsten Abschnitt.

Wie können Sie das testen?

Werkzeug der Wahl ist die Web Developer Toolbar für Firefox. Im Menü ›Tools‹ finden Sie einige Befehle, um die aktuell angezeigte Seite an den W3C-Validator zur Überprüfung zu schicken. Dabei muss dies nicht mal ein Dokument sein, dass auf einem aus dem Netz erreichbaren Server liegt. In neueren Versionen kann die Toolbar auch lokal gespeichertes HTML an den Validator schicken (›Validate Local HTML‹).

Alternativ können Sie auf lokale Syntax-Checker zurückgreifen, die in so gut wie allen Autorentools wie Dreamweaver oder BBEdit bereits eingebaut sind. Andere Editoren wie TopStyle bieten die Möglichkeit, den Code der erstellten Seiten an eines der Online-Prüfprogramme zu übergeben. Verweise und kurze Erläuterungen zu diesen Werkzeugen finden Sie im Artikel »Links zur Überprüfung Ihrer Website«.

Übrigens: auch RSS- & Atom-Feeds und andere auf XML basierende Formate sind mit »Markup-Sprachen geschaffene Dokumente« im Sinne der Bedingung 3.2 und sollten entsprechend gültig bzw. wohlgeformt sein.

Beispiele:

Dieser Link übergibt die aktuelle Seite an den HTMLValidator des W3C:

Dieser Link übergibt das Style Sheet der aktuellen Seite an den CSS-Validator des W3C:

Für welche der unterscheidlichen Varianten von HTML & XHTML Sie sich entscheiden ist letztendlich Geschmackssache. Wir haben uns beim letzten Relaunch nach reiflicher Überlegung wieder für HTML 4.01 strict entscheiden und die Beweggründe auch in der Dokumentation des Relaunches begründet.

Probleme bei der zukünftigen Bewertung

Während die Forderung nach validem Code in der BITV noch bei Priorität 1 einsortiert ist, wird dieser Punkt bei den in der Entwicklung befindlichen WCAG 2.0 weiterhin nur mit geringerer Priorität bewertet. Dass dieser Punkt auch in der Version 1.0 der WCAG nur bei der Priorität 2 (AA) aufgeführt war ist verständlich, wenn man dies im historischen Kontext betrachtet. Zur Entstehungszeit der WCAG 1.0 waren in der Tat die meisten Browser mit standardkonformem Code überfordert, sodaß man diese Richtlinie nicht als ›*muss*‹ durchsetzen konnte. Dies ist heute grundsätzlich anders, sodaß es eigentlich keinen vernünftigen Grund gibt, dass eine Arbeitsgruppe des W3C die Spezifikationen aller anderen Arbeitsgruppen für nicht so wichtig erklärt.

Wie zu erwarten war hat diese Entscheidung der Web Accessibility Initiative für Unruhe und Widerspruch in der Szene gesorgt:

- Auf der einen Seite steht das (nachvollziehbare) Argument, dass es im Netz gravierendere Barrieren als unkodierte Ampersands gibt. Das Validität nicht automatisch Barrierefreiheit bedeutet dürfte sich mittlerweile herumgesprochen haben,

sonst gäbe es ja auch keinen Bedarf an diesen Richtlinien und man könnte in der BITV kurzerhand auf die Empfehlungen zu HTML und ähnlichem verweisen.
- Auf der anderen Seite steht das (ebenfalls nachvollziehbare) Argument, dass man sich erst mit einer sauberen Grundlage an das Beseitigen dieser Barrieren machen kann, die Verarbeitung des Codes gerade auch für die Hilfsmittel behinderter Menschen vereinfacht wird und vieles mehr.

Einen Überblick über den Stand der Diskussion können Sie sich in unserem Weblog unter dem Tag WCAG verschaffen.

Bedingung 3.3: Style Sheets verwendet (Prio. 1)

»Es sind Stylesheets zu verwenden, um die Text- und Bildgestaltung sowie die Präsentation von mittels Markup-Sprachen geschaffener Dokumente zu beeinflussen.« <zitat>

Was heißt das?

Nichts anderes als die saubere Trennung von Inhalt und Verpackung und das damit implizierte Verbot von Layouttabellen. Das HTML Ihrer Seiten soll die Inhalte und deren Struktur festlegen, die Gestaltung für alle denkbaren Ausgabeformen wird hingegen mit Cascading Style Sheets (CSS) erreicht.

Konkret bedeutet das, dass Sie z. B. Ihre Überschriften im HTML-Code als solche auszeichnen (also mit `H1-H6`) und die Gestaltung dann über eine (in den meisten Fällen externe) CSS-Datei.

Übrigens: auch Flash unterstützt seit der Version MX 2004 einiges aus CSS 1, wie zum Beispiel Schriftgrößen, Farben, Einzüge und eine Reihe weiterer CSS-Properties. Da die Texte in einer Flash-Datei aus einer externen HTML- oder XML-Datei stammen können, sollten Sie auch hier CSS zur Formatierung einsetzen.

Wie können Sie das testen?

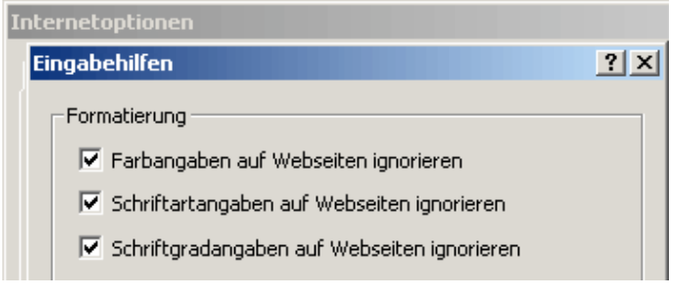
ToggleCSS von Tantek Çelik ist ein Bookmarklet, das sie in die Lesezeichenleiste Ihres Browsers legen können. Der Vorteil dieser Bookmarklets (oder Favelets) besteht darin, dass es simple JavaScripts sind, die bestimmte Aktionen mit Ihren Seiten ausführen. Im konkreten Fall werden durch einen Klick sämtliche Formatierungen, die aus einem Style Sheet kommen, aus der Seite entfernt. Was dann noch übrig bleibt, ist im Idealfall die pure Struktur, wie sie im HTML steht.

Weitere fleißige Helferlein dieser Art finden Sie im Netz beim W3C oder bei squarefree.com/bookmarklets.

Sie können in Ihrem Browser das Laden von CSS komplett unterbinden. Sinnvoll ist dies um z. B. einen ganzen Satz Dateien nacheinander zu testen, ohne jedes mal die bereits geladenen Styles wieder händisch zu entfernen. In Opera können Sie dazu in den Benutzermodus umschalten und in dem entsprechenden Dialog weitere Dinge festlegen, die der Browser anwenden oder ignorieren soll.

Erscheint Ihre Seite nun nicht in der Standardschrift des Webbrowsers (also höchstwahrscheinlich Times) oder sind immer noch andere Vorder- und Hintergrundfarben als schwarz und weiß zu sehen (blaue Links natürlich ausgenommen), so verstecken sich noch Formatierungsanweisungen im HTML–Code.

Das Pendant dazu im Internet Explorer finden Sie unter ›Extras‹ → ›Internetoptionen‹ → ›Eingabehilfen‹. Allerdings wird hier durch die Vorgabe ignorieren nicht die gesamte CSS-Gestaltung entfernt, sondern nur die Angaben zu Schriftarten und -größen sowie die Farbangaben. Die Positionierung der einzelnen Seitenelemente per CSS bleibt nach wie vor erhalten.



Falls Sie diese Seiten hier gerade in einem aktuellen Browser wie Mozilla oder Firefox lesen, schalten Sie doch einfach mal alle Style Sheets ab. Den Befehl dazu finden Sie im Menü Ansicht → Webseiten-Stil → Kein Stil. Schon erhalten Sie den reinen ungestalteten Text, der nach wie vor über seine eingebaute Struktur verwertbar ist.

Bedingung 3.4: Maßangaben variabel (Prio. 1)

»Es sind relative anstelle von absoluten Einheiten in den Attributwerten der verwendeten Markup-Sprache und den Stylesheet-Property-Werten zu verwenden.« <zitat>

Was heißt das?

Wenn Sie diese Frage im Jahr 2006 gestellt hätten, dann hätten wir eine eindeutige Antwort gewusst: alles was dem Benutzer dabei hilft, ein Dokument oder eine Anwendung an seine Bedürfnisse anzupassen, sollte flexibel bemaßt werden. Dazu zählen Schriftgrößen, die sich nach den Voreinstellungen der Nutzer richten sollen und Spaltenbreiten in mehrspaltigen Layouts, die sich per CSS ebenfalls flexibel anlegen lassen.

Übrigens: hierbei handelt es sich um eine fehlerhafte Formulierung der BITV bzw. der WCAG 1.0. Formaljuristisch betrachtet bedeutet diese Bedingung, dass Sie weiterhin alles in der Einheit `px` (Pixel) bemaßen dürfen: `px` ist laut der CSS-Spezifikation eine relative Einheit, da die Größe der Pixel abhängig von der Bildschirmgröße und -auflösung ist. Wir können uns aber denken was gemeint ist und ignorieren diesen Fehler.

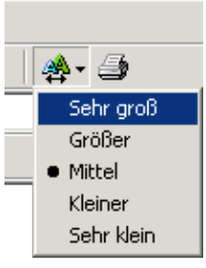
Sämtlichen Inhalte einer HTML-Seite können via CSS nicht nur Informationen über ihr Aussehen, sondern auch über ihre Größe und Position mitgegeben werden. Für Datentabellen können Sie natürlich weiterhin die Bemaßungen im HTML-Code angeben und trotzdem die BITV erfüllen. Sie sollten nur darauf achten, dass diese Bemaßungen nicht in starren Einheiten (also Pixel, Punkt, `cm`, `mm` oder Zoll), sondern flexibel (also `em`, `ex` oder `%`) gesetzt werden. Nur so passen sich Spalten oder Tabellen an die Größe des Browserfensters an.

Eingebettete Rasterbilder haben generell Pixeldimensionen; allerdings gibt es auch Methoden, die Breite von Bildern in `em` oder `%` anzugeben, so daß diese bei Größenänderungen ›gezoomt‹ werden – wenn auch bei starker Vergrößerung mit nicht immer optimalen Ergebnissen. Wenn es Ihre Gestaltung und die Inhalte zulassen können Sie auch andere eingebettete Inhalte wie z. B. Flash relativ bemaßen.

Ausnahmen kann und sollte es auch hier geben:

- Größenangaben für Gestaltungselemente wie Ränder, Linien etc. können Sie natürlich weiterhin in Pixel angeben.
- Im Style Sheet für die Druckausgabe ist es sinnvoller, die Angaben für Dimensionierung und Platzierung in `cm` oder `mm` zu setzen und die Angaben für Schriftgrade in typografischen Punkten (`pt`).

Aaaaber…



So einfach ist die Antwort aber in 2007 nicht mehr. Mit dem Erscheinen des Internet Explorer 7 hat sich die Grundlage für diese Bedingung in Luft aufgelöst. Sie war ursprünglich dazu gedacht Browsern auf die Sprünge zu helfen, die in `px` bemaßte Elemente (insbesondere Texte) nicht skalieren konnten. Hierzu gehörten insbesondere die Vorgänger des IE7 bis einschließlich zur Version 6. Streng genommen konnten auch diese schon jegliche Texte skalieren, die Option hierfür war nur a. werksseitig ausgeschaltet und b. nicht unmittelbar zu finden.

Mit dem Erscheinen des IE7 zieht Microsoft mit Opera gleich und bietet eine echte Zoom-Funktion, mit der sich ganze Seiten skalieren lassen. Eine ähnliche Funktion ist dem Vernehmen nach für die kommende Firefox-Version 3.0 in Planung, sie lässt sich aber auch jetzt schon mit der Erweiterung PageZoom nachrüsten. Zudem skaliert der Internet Explorer 7 auch in `px` bemaßte Texte, wie dies bisher schon sämtliche anderen Browser wie Mozilla, Firefox und Safari getan haben.

Welche Auswirkungen dies auf die verschiedenen Ansätze zur flexiblem Bemaßung haben wird, lässt sich zum Zeitpunkt der Veröffentlichung dieses Artikels noch nicht sagen – dafür sind die Konzepte noch zu neu und mögliche Lösungen noch zu wenig erprobt.

Bedingung 3.5: Strukturelemente für Uberschriften (Prio. 1)

»Zur Darstellung der Struktur von mittels Markup-Sprachen geschaffener Dokumente sind Überschriften-Elemente zu verwenden.«

<zitat>

Was heißt das?

Sie kennen das aus Ihrer Textverarbeitung: man kann Überschriften entweder als großen, fetten, farbigen Text anlegen, oder man kann eine Formatvorlage benutzen, um eine Überschrift als Überschrift 1, Überschrift 2 oder Überschrift 3 zu definieren. Beides sieht auf dem Monitor und im Ausdruck gleich aus, nur bedeutet der erste Weg wesentlich mehr Arbeit. Zudem kann Ihre Textverarbeitung echte Überschriften in der Dokumentstruktur als solche ausgeben und ihnen damit die Navigation im Dokument vereinfachen.

Nichts anderes geschieht im Web, wenn sie Ihre Überschriften mit den dafür vorgesehenen Tags `<h2>–<h6>` auszeichnen. Nicht nur Screenreader und Lupenprogramme können nun den Aufbau einer Seite besser wiedergeben und es dem Benutzer ermöglichen, sich in dieser Struktur von Überschrift zu Überschrift zu bewegen. Auch immer mehr grafische Browser können Seiten auf diese Art zusammenfassen. Dazu ist es aber zwingend erforderlich, dass diese strukturierenden Elemente auch als solche im Quelltext ausgezeichnet und nicht nur ›ungefähr so aussehen.‹

Wie können Sie das testen?

Hier kommen die zuvor erwähnten Eingabehilfen im Internet Explorer zum Einsatz. Schalten Sie alle Angaben zu Schriftgrad und –art ab und lesen Sie Ihren Text. Wenn die Überschriften genau so aussehen wie Ihr Fließtext, bedeutet dies, das diese nicht korrekt ausgezeichnet sind. Denn auch ohne Angaben aus CSS werden Überschriften in grafischen Browsern entsprechend Ihrer Bedeutung dargestellt, d. h. `h1` sehr groß und fett, `h4` oder `h5` dagegen in der Größe von normalem Fließtext.

Alternativ können Sie auch einen der Browser zu Rate ziehen, die bereits die Navigation per Überschriften beherrschen und so ihren Nutzern die Navigation erleichtern. Mit einigen Kniffen können dies alle grafischen Browser, von Mozilla und Firefox über Opera bis hin zu Exoten wie iCab.

Eine empfehlenswerte Firefox-Erweiterung für diesen Test ist Document Map. Nach der Installation sollten Sie sich das entsprechende Icon in die Symbolleiste des Browsers legen. Per Klick öffnet sich die hierarchische Überschriftenstruktur der aktuellen Seite in der sog. Sidebar. Versuchen Sie nun, nur über das Springen von Überschrift zu Überschrift zu navigieren – dieses Verhalten entspricht den Möglichkeiten von assistiven Programmen, Überschriften aus dem Kontext losgelöst darzustellen. Wenn die wesentlichen Inhaltsbereiche eines Dokumentes über diese Überschriftenstruktur erreichbar sind, dann haben Sie diesen Test bestanden.

Dabei ist es *unerheblich*, ob ein Dokument immer mit einer `<h2>` oder nur mit einer `<h3>` oder `<h4>` anfängt und sämtliche Hierarchieebenen immer und überall konsequent eingehalten werden (also grundsätzlich `<h2>` → `<h3>` → `<h4>` → …). Gerade in dynamisch erzeugten Seiten ist es oft unmöglich und letztendlich nicht hilfreich, diese strikte Ordnung immer herzustellen.

Der Kern dieser Bedingung 3.5 ist: wenn der Inhalt nach einer Überschrift verlangt, dann sollte diese entsprechend als Überschrift ausgezeichnet werden. Weitergehende Vorgaben zur Strukturierung Ihrer Inhalte finden Sie bei den Bedingungen 12.3 und 13.8.

Beispiele:

Bevor CSS starke Verbreitung fand, wurden Überschriften üblicherweise so ausgezeichnet:

```
<br><br>
<font size="7"><b>
 <font face="Helvetica">
  <font color="#ff0000">
   Großer fetter Text, der so aussieht wie eine Überschrift
  </font>
 </b></font>
<br></font><br>
```

Mit CSS und HTML sollte die wichtigste Überschrift Ihrer Seite nun so aussehen.

```
<h2>Wichtige Überschrift</h2>
```

Die Formatierung über CSS bringt nun das gewünschte Aussehen:

```
h1 {
  font  :  bold 2em/150% Helvetica, sans-serif;
  color :  #f00;
}
```

Bedingung 3.6: Strukturelemente für Listen (Prio. 1)

»Zur Darstellung von Listen und Listenelementen sind die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache zu verwenden.« <zitat>

Was heißt das?

Wie bei den meisten HTML-Elementen lassen sich auch Textauszeichnungen zu Zwecken missbrauchen, für die sie nicht vorgesehen sind. So kann man Listen und Aufzählungen durch Einrückungen, Zeilenschaltungen und Bilder als Listenzeichen erreichen. Diesen fehlt aber die eingebaute Logik, die richtige Listen mit den Elementen `UL` (unsortierte Liste), `OL` (sortierte Liste mit Ordnungszahlen oder -buchstaben) und `DL` (Definitionsliste) mit sich bringen.

Wie können Sie das testen?

Dieser Test ist dem vorhergehenden Test mit den Überschriften ähnlich: schalten Sie alle Formatierungen in Ihrem Browser ab und kontrollieren Sie, ob vorhandene Auflistungen noch eingerückt sind und mit einem Listensymbol (quadratisches oder rundes Bullet) oder einer Zahl versehen sind. Ist dies so, dann sind es echte Listen und Sie haben diesen Test bestanden.

Beispiele:

Beispiele für Definitionslisten finden Sie im Glossar zur BITV. Ein durchaus zulässiger Anwendungszweck für Listen sind sogenannte Codelistings, wie sie bei EfA in den eher technisch orientierten Artikeln eingesetzt werden (Beispiel–Listing). In diesem Fall sind es nummerierte Listen (`<ol>`), die sich so verhalten wie Quelltext in einem Editor.

Eine Auflistung von Navigationspunkten ist ebenfalls *eine Liste*. In Ermangelung eines passenderen HTML-Elementes kommen `OL` und insbesondere `UL` von ihrer Bedeutung dem gewünschten Ziel am nächsten.

Bedingung 3.7: Strukturelemente für Zitate (Prio. 1)

»Zitate sind mittels der hierfür vorgesehenen Elemente der verwendeten Markup-Sprache zu kennzeichnen.« <zitat>

Was heißt das?

HTML kennt für Zitate und Quellenangaben verschiedene Elemente, die hierfür (und nur hierfür) verwendet werden sollen. Im Einzelnen sind dies `<blockquote>` für längere Zitate (und nicht für Einrückungen!), `<q>` für kürzere Zitate und `<cite>` für Quellenangaben. Zusätzlich kennt HTML noch das `cite`-Attribut (nicht zu verwechseln mit dem `cite`-Element) über das Zitaten (`Q` und `BLOCKQUOTE`) eine Quellenangabe zugewiesen werden kann.

Wie können Sie das testen?

Hier hilft nur der Blick in den Quelltext, da diese Elemente je nach Gestaltung nicht von herkömmlichem Text zu unterscheiden sind. Ausnahme: mit der Web Developer Toolbar für Firefox und vergleichbaren Werkzeugen bekommen Sie die Möglichkeit, bestimmte Elemente hervorzuheben. Wählen Sie hierzu den Befehl ›Outline Custom Elements‹ und tragen im folgenden Dialog `Q`, `BLOCKQUOTE` und `CITE` ein. Nun werden diese Elemente, sofern vorhanden, auf der aktuellen Seite hervorgehoben.

Beispiele:

Im EfA-Artikel »Den Trend nicht verschlafen« finden Sie folgende Textpassage:

```
<p>Die Zugänglichkeit im Internet regelt §11: <q cite="http://www.behindertenbeauftragte.de/index.php5?nid=20&amp;Action=home">Träger öffentlicher Gewalt … gestalten ihre Internetauftritte … schrittweise technisch so, dass sie von behinderten Menschen grundsätzlich uneingeschränkt genutzt werden können.</q></p>
```

Hier wurde das Zitat aus dem Gesetzestext mit dem vorgesehenen Tag `<q>` ausgezeichnet. Zusätzlich enthält das `<q>`-Tag noch ein `cite`-Attribut, in dem der URL der Quelle angegeben ist. Das gesamte Zitat steht wiederum in einem `<blockquote>`, da hier ja eine Textpassage aus einem anderen Artikel zitiert ist.

In modernen grafischen Browsern kann man per CSS die korrekten typografischen Anführungszeichen bestimmen: `q {quotes : '\201c' '\201d' '\2018' '\2019';}`

Anforderung 4: Sprachliche Besonderheiten

»Sprachliche Besonderheiten wie Wechsel der Sprache oder Abkürzungen sind erkennbar zu machen.« <zitat>

Was heißt das?

In diesem Punkt geht es um eine der wenigen reinen Accessibility–Anforderungen der BITV: die Kennzeichnung von sprachlichen Besonderheiten. Ziel ist es, die Barrieren für Nutzer alternativer Zugangsformen oder assistiver Werkzeuge abzusenken und das allgemeine Textverständnis zu fördern.

Bedingung 4.1: Sprachwechsel ausgezeichnet (Prio. 1)

»Wechsel und Änderungen der vorherrschend verwendeten natürlichen Sprache sind kenntlich zu machen.« <zitat>

Was heißt das?

Sinn dieser Anforderung ist, dass Sprachausgaben, wie sie von Menschen mit Seh– oder Lernbehinderung genutzt werden, einen Sprachwechsel oder andere sprachliche Besonderheiten in einem Dokument erkennen. Sollten in einem hauptsächlich deutschen Text fremdsprachliche Worte vorkommen, so müssen diese auch als solche ausgezeichnet werden. Dazu gehören nicht nur mittlerweile eingedeutschte Begriffe wie Browser oder Website, sondern auch Anglizismen. Nur mit einer entsprechenden Kennzeichnung können Fremd- und Lehnwörter nach den Ausspracheregeln ihrer ursprünglichen Sprache vorgelesen werden.

Zur Entstehungszeit dieser Verordnung waren die Umsetzung und vor allem der praktische Nutzen dieser speziellen Bedingung kaum zu vermitteln. Viele assistive Programme verstanden die Anweisungen für Sprachwechsel schlicht und ergreifend nicht; selbst Programme wie der mittlerweile eingestellte IBM Home Page Reader, der als einer der ersten Sprachwechsel beherrschte, legten gerne mal eine Gedenkminute ein, um zum Beispiel zwischen der deutschen und der englischen Stimme hin- und herzuschalten.

Mittlerweile beherrschen aber alle halbwegs modernen Hilfsmittel diese Technik und können korrekt ausgezeichnete Fremdwörter wie `<span lang="en">Browser</span>` in der jeweils deklarierten Sprache vorlesen. Aus Nutzersicht besteht also kein Grund mehr, auf diese Unterstützung verzichten zu müssen. Ob ein Nutzer seine Sprachausgabe nun so eingestellt hat, dass Sprachwechsel ignoriert werden ist dabei unerheblich – es geht, also sollte es gemacht werden.

Nach wie vor problematisch ist hingegen die Auszeichnung der Sprachwechsel. Diese kann nur zum Teil automatisch von einem Content Management System oder anderen Autorenwerkzeugen geleistet werden, da hierfür ein Verständnis von Sprache notwendig ist, dass diese Programme nicht haben können. Auch mit ausgefeilten Suchmustern kann kein gängiges Programm wirklich verlässlich unterscheiden, ob mit »Tag« nun ein Tag der Woche oder ein HTML-Tag gemeint ist. Hier ist also teilweise nicht unerheblicher manueller Aufwand seitens der einpflegenden Redakteure nötig.

In der Praxis bewährt hat sich die Abgrenzung zwischen Sprachwechseln, die z. B. in einem Template nur einmal vorgenommen werden müssen und damit global für die gesamte Website gelten (Beispiel: `<span lang="en">Sitemap</span>` in einer Navigation), und Sprachwechseln im Fließtext, deren vollständige und korrekte Auszeichnung gerade in tagesaktuellen Publikationen einen durch nichts zu rechtfertigenden Aufwand verursachen würden.

Wie können Sie das testen?

Eine Bedingung, die nicht so einfach mit den Bordmitteln herkömmlicher Browser zu testen ist, es sei denn, man wagt den Blick in den Quelltext. Auch hier helfen wieder die gängigen Browser-Erweiterungen wie die AIS Web Accessibility-Toolbar für den Internet Explorer oder die Web Developer Extension für Firefox. Die AIS Toolbar hat im ›Menü‹ Seite einen Befehl ›lang-Attribute anzeigen‹, der auf der aktuellen Seite alle vorhandenen `lang`-Attribute mit deren zugehörigen Elementen anzeigt.

Bei der laufenden Überarbeitung der ›Einfach für Alle‹-Seiten wird natürlich auch auf die Einhaltung dieser Bedingung geachtet. Zur Kontrolle der Sprachauszeichnungen wurde ein separates Style Sheet erstellt, in dem auf das Layout verzichtet wird und statt dessen alle sprachlichen Besonderheiten wie Sprachwechsel und Abkürzungen farbig hervorgehoben sind. Sie können dieses Style Sheet herunterladen und in jedem modernen Browser als ihr eigenes User Style Sheet installieren, das alle Angaben auf Webseiten überschreibt.

Für die Abgrenzung, was ausgezeichnet werden sollte und was nicht findet man in der Literatur gelegentlich den Hinweis auf Wörterbücher wie den Duden. Was im Duden steht kann demnach als ›eingedeutscht‹ gelten und braucht nicht ausgezeichnet zu werden. Dies würde jedoch bedeuten, dass Screenreader zur Bestimmung der korrekten Aussprache im Duden nachschlagen müssten (was sie aus nachvollziehbaren Gründen nicht machen). Auch die Prozentzahl der ausgezeichneten fremdsprachlichen Begriffe ist ganz sicher keine verlässliche Grundlage zur Bewertung, ob diese Bedingung erfüllt ist oder

nicht. Die Regel sollte hier eher sein, dass eine unterlassene Sprachkennzeichnung nicht zu Verwechslungen oder Sinnentstellungen führen darf.

Bedingung 4.2: Sprachliche Besonderheiten ausgezeichnet (Prio. 2)

»Abkürzungen und Akronyme sind an der Stelle ihres ersten Auftretens im Inhalt zu erläutern und durch die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache kenntlich zu machen.« <zitat>

Was heißt das?

Eine schwierig umzusetzende Bedingung, und dies nicht nur weil sie, wie schon die vorhergehende, viel manuelle Nacharbeit bei der Einpflege der Inhalte bedeutet. Die eigentliche Schwierigkeit liegt eher darin, dass sich auch Sprachwissenschaftler nicht einig sind, was denn nun ein Akronym und was eine Abkürzung sei. Selbst das W3C benutzt in seiner Erklärung der Elemente falsche Beispiele.

Hintergrund dieser Bedingung ist die Annahme, dass assistive Werkzeuge dadurch Abkürzungen und Akronyme als Folge von Buchstaben wiedergeben können statt zu versuchen, diese als ein Wort auszusprechen. Nur gibt es hier in allen Sprachen mehr Ausnahmen als Regeln, z. B. ist UN ein Akronym wie GIF (ausgesprochen: giff oder dschiff) oder WAI (ausgesprochen: wäi) und müsste folgerichtig Un ausgesprochen werden – was natürlich keiner macht. Im Gegensatz dazu ist USA entgegen landläufiger Meinung kein aus den Anfangsbuchstaben gebildetes Akronym, sondern eine Abkürzung, da das o aus United States *of* America fehlt. Ganz unmöglich sind Abkürzungen, bei denen keine Einigkeit besteht, wie diese nun benutzt werden sollten: SQL kann sowohl Sequel als auch Ess Kjuh El ausgesprochen werden – für den Webentwickler ein nicht lösbares Problem.

Der Internet Explorer versteht auch in aktuellen Versionen immer noch nicht das `ABBR`-Element (es sei denn man setzt Hacks wie <abbr-cadabra/> ein). Folgerichtig können viele assistive Werkzeuge, die auf diesen Browser aufsetzen, nichts mit der eingebauten Logik dieser Tags anfangen und müssen sich auf ihre eingebauten Wörterbücher verlassen.

Nach dem Wortlaut der obigen Bedingung muss immer das **erste** Auftreten einer Abkürzung im Inhalt kenntlich gemacht und erläutert werden – was dem Nutzer jedoch nichts nützt, wenn er von einem anderem Dokument über einen lokalen Anker an eine Textpassage weiter unten gekommen ist und so vielleicht die Erläuterung bereits verpasst hat. Auch werden Menschen mit Lernbehinderungen, die unter Umständen bei jeder Instanz einer Abkürzung auf die Erklärung angewiesen sind, mit dieser Einschränkung nicht berücksichtigt.

Mehr zur Problematik von Abkürzungen finden Sie in unserem Artikel von Jan Hellbusch: »Zett Punkt Bee Punkt«.

Angesichts dieses Durcheinanders hat das W3C im aktuellen Entwurf für XHTML2 das `ACRONYM`-Element fallengelassen und sieht nur noch `ABBR` vor. »The abbr element indicates that a text fragment is an abbreviation … **this includes acronyms**.«

Kein User Agent unterstützt die dafür vorgesehenen Hilfen zur Aussprache in CSS 2 (`speak: normal` oder `speak: spell-out`), auch nicht die Screenreader und Sprachbrowser. Was zur Folge hat, dass die entsprechenden Anweisungen aus Aural CSS in der jetzigen Form in Zukunft (also CSS 3) nicht mehr vorhanden sein werden.

Wie können Sie das testen?

Grafische Browser dekorieren Abkürzungen und Akronyme üblicherweise mit einem gepunkteten Unterstrich, sobald in die entsprechenden Tags ein zusätzliches `title`-Attribut eingefügt wurde. Allerdings kann dieses Verhalten auch per CSS unterbunden werden, sodaß Sie zur genaueren Kontrolle auf die Unterstützung eines Prüfprogramms angewiesen sind. Die gängigen Toolbars verfügen über Funktionen, mit denen entsprechend ausgezeichnete Abkürzungen und Akronyme farblich hervorgehoben werden.

Alternative Beispiele:

In einer früheren Version der WCAG 2.0 wurde auf diese Vorgabe verzichtet, stattdessen sollten Verweise auf ein Glossar ermöglicht werden: »Expansion dictionaries, for instance in metadata, may be provided as an alternative to an expansion in the text of a document.« Diese Glossare können über den Aufruf `<link rel="glossary" href="http://www.einfach-fuer-alle.de/artikel/bitv/glossar/" title="Glossar">` mit jeder Seite verknüpft werden. Da es sich bei dieser Methode jedoch um weitestgehend unsichtbare Metadaten handelt sollten Sie das Glossar zusätzlich aus dem Inhalt der Seite oder Anwendung heraus verlinken.

In dieser Serie beziehen wir uns der Einfachheit halber auf das Glossar aus der BITV (Anlage 2), dort erklärte Fachbegriffe und Fremdwörter werden aus den jeweiligen Bedingungen heraus direkt verlinkt.

Bedingung 4.3: Sprache der Dokumente (Prio. 2)

»Die vorherrschend verwendete natürliche Sprache ist durch die hierfür vorgesehenen Elemente der verwendeten Markup-Sprache kenntlich zu machen.«

<zitat>

Was heißt das?

Jedes gute HTML-Dokument gibt im Kopf an, in welcher Sprache der Text des Dokumentes verfasst wurde. Dies vereinfacht die maschinelle Verarbeitung; erst so werden zum Beispiel Screenreader in die Lage versetzt, einen Text in der Sprache vorzulesen, in der ein Autor ihn verfasst hat.

Einer der Anwendungszwecke für diese Deklaration ist die Fähigkeit der meisten grafischen Browser, je nach verwendeter Sprache die korrekten typografischen Anführungszeichen bei Zitaten zu setzen. Deutsche Autoren verwenden hier üblicherweise andere Zeichen als Franzosen, Guillemets werden in der Schweiz andersrum notiert («…») als hierzulande (»…«) und Amerikaner nehmen für alles das Zeichen für Zoll ("…").

Vollkommen unverständlich ist die Tatsache, dass im Gegesatz zu den viel schwieriger umzusetzenden Vorgaben der Bedingung 4.2 dieser Bedingung nur mit Priorität 2 bewertet werden soll. Gerade bei Template-basierten Angeboten, die über ein CMS oder Weblog-System generiert werden beschränkt sich der Arbeitsaufwand auf das einmalige Eintippen eines kurzen Attributs mit den korrekten Werten.

Eine Ausnahme sind Seiten, die keine eindeutige vorherrschende Sprache habe, also wo Texte aus mehreren Sprachen in etwa gleichwertig hinterlegt sind. In diesem Fall müssen die Textabschnitte in der jeweils verwendeten Sprache ausgezeichnet werden.

Wie können Sie das testen?

Wenn Sie sich nicht jede Seite einzeln anhören wollen hilft der Blick in den Quelltext. Wonach Sie suchen müssen, zeigt das folgende Beispiel.

Beispiele:

In HTML gibt es zu diesem Zweck das `lang`–Attribut des `HTML`-Elements, daher sollten Sie im Quelltext einer HTML4-Seite nach der Zeile: `<html lang="de">` bzw. bei XHTML-Dokumenten nach `<html xml:lang="de" lang="de">` suchen. Stehen diese am Anfang des Dokumentes unmittelbar nach der DTD, haben Sie diesen Test bestanden. Wenn dort nur `<html>` steht, sollten Sie die Angabe der Sprache des Textes noch hinzufügen. Eine Liste aller von der ISO genormten Sprachkürzel finden Sie bei oasis-open.org.

Anforderung 5: Tabellen

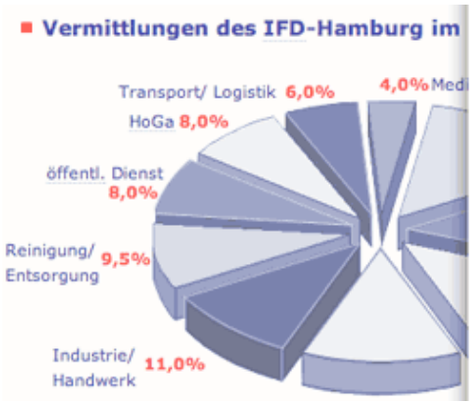
»Tabellen sind mittels der vorgesehenen Elemente der verwendeten Markup-Sprache zu beschreiben und in der Regel nur zur Darstellung tabellarischer Daten zu verwenden.«

Was heißt das?

Lange genug die einzig verlässliche Methode, um Webseiten ein einigermaßen konsistentes Aussehen mitzugeben, haben Layout-Tabellen mittlerweile ihr Verfallsdatum deutlich überschritten. Folgerichtig beschreitet die BITV den Weg weg von veralteten Praktiken des Webdesigns und damit hin zu CSS und seinen moderneren Möglichkeiten.

Die Anforderung selbst ist recht weit gefasst und lässt gerade beim Thema Layout-Tabellen viel Spielraum für nicht mehr ganz zeitgemäße Interpretationen. Die folgenden Bedingungen sind konkreter, so dass sich hieraus eine nutzbare Handlungsanweisung ableiten lässt.

Gleichzeitig wird der Einsatz der korrekten Elemente und Atribute für Datentabellen nach wie vor verlangt. Ein ansehnlich gestaltetes Beispiel zeigt der BIENE-prämierte Integrationsfachdienst Hamburg. Mit CSS sieht man ein Tortendiagramm, ohne CSS erkennt man, dass es sich bei der Torte um eine echte Datentabelle handelt, die lediglich gut verpackt wurde:



Vermittlungen des IFD-Hamburg

<th> Branche	<th> Anteil an
<td> Einzelhandel	<td> 14,5%
<td> Großhandel/ Banken/ Versicherungen	<td> 13,0%
<td> Soziales/ Gesundheit	<td> 13,0%
<td> Personaldienstleistung	<td> 13,0%
<td> Industrie/ Handwerk	<td> 11,0%
<td>	<td>

Bedingung 5.1: Tabellen mit Überschriften (Prio. 1)

»In Tabellen, die tabellarische Daten darstellen, sind die Zeilen- und Spaltenüberschriften mittels der vorgesehenen Elemente der verwendeten Markup-Sprache zu kennzeichnen.«

Was heißt das?

Es gilt zu bewerten, ob eine gegebene Datentabelle Zeilen- und Spaltenüberschriften tatsächlich benötigt. Wenn Ihre Seiten echte tabellarische Daten enthalten (also z. B. Wetterdaten, Börsenkurse oder Preislisten), ist es wahrscheinlich, dass es zu den verschiedenen Zeilen und Spalten auch die entsprechenden Zeilen- und Spaltenköpfe mit Überschriften für die jeweiligen Datenzellen gibt. Ist dies der Fall, dann sollten diese nicht als einfache Zellen (`TD`), sondern mit den hierfür in HTML vorgesehenen Überschriftenelementen (`TH` und dem dazugehörigen `THEAD`, optional auch `TFOOT`) ausgezeichnet sein.

Beispiele:

Im Rahmen unserer Artikelserie zur BITV haben wir eine Tabelle mit einer beispielhaften Übersicht der Verantwortlichkeiten zur Umsetzung der Verordnung erstellt. Darin sind die einzelnen Punkte der BITV aufgelistet bzw. kurz beschrieben und ob die jeweilige Bedingung eher in den Verantwortungsbereich von Konzeptern, Designern, Technikern oder Redakteuren fällt:

BITV	Prio.	Prüfschritt	Konzept	Design	Technik	Pflege bzw. Kunde	*›until user agents‹*
1.		Bereitstellung äquivalenter Alternativen					
1. 1.	1	Alternativtexte für Bedienelemente	-	-	+	-	-
1. 1.	1	Alternativtexte für Grafiken und Bilder	-	-	+		-

In dieser Tabelle dient die erste Zeile eindeutig als Spaltenüberschrift für die darunter stehenden Datenzellen, folgerichtig sind sie mit `<th>` ausgezeichnet.

Wie können Sie das testen?

Diese Bedingung ist relativ einfach zu testen, da die gängigen grafischen Browser bei der Betrachtung von reinem HTML die Zeilen- und Spaltenüberschriften (`TH`) anders darstellen als normale Tabellenzellen. Diese sind einfach zu identifizieren, indem Sie das Laden der Style Sheets unterbinden und sich auf das Browser-eigene Style Sheet verlassen. Am einfachsten geht dies mit einer der bekannten Toolbars; in der Web Developer Extension finden Sie den entsprechenden Befehl unter ›CSS‹ → ›Disable Styles‹ → ›All Styles‹. Die Tabelle sollte dann in etwa wie im folgenden Screenshot aussehen. Zu erkennen ist, dass hier die Spaltenüberschriften fett und zentriert dargestellt werden – diese sind also korrekt ausgezeichnet:

BITV	Prio.	Prüfschritt	Konzept	Design	Technik	Pflege bzw. Kunde	›until user agents‹
1.		Bereitstellung äquivalenter Alternativen					
1.1.	1	Alternativtexte für Bedienelemente	-	-	+	-	-
1.1.	1	Alternativtexte für Grafiken und Bilder	-	-	+		-

Bedingung 5.2: Zuordnung von Zellen zu Überschriften (Prio. 1)

<zitat>

»Soweit Tabellen, die tabellarische Daten darstellen, zwei oder mehr Ebenen von Zeilen- und Spaltenüberschriften aufweisen, sind mittels der vorgesehenen Elemente der verwendeten Markup-Sprache Datenzellen und Überschriftenzellen einander zuzuordnen.«

Was heißt das?

Die in Bedingung 5.1 besprochenen Zeilen- und Spaltenüberschriften sollten mit den Datenzellen, denen Sie als Überschriften dienen, programmatisch verküpft sein. Dies erleichtert die Orientierung in nicht-visuellen Ausgabemedien, indem es Zusammenhänge herstellt, die für sehende Nutzer offensichtlich sind und ist somit für die Nutzer vieler Hilfsmittel von essentieller Bedeutung. Dies gilt insbesondere für komplexe Tabellen mit mehreren Ebenen von Überschriften, wo eine Zuordnung ohne die hier eingeforderten Attribute nicht möglich ist.

Diese Bedingung ist anzuwenden, sobald Datentabellen zwei oder mehr `TH`-Elemente innerhalb einer Zeile `TR` aufweisen, wenn zwei oder mehr `TR`-Elemente mindestens eine `TH` enthalten oder wenn eine *TH* für mehrere Spalten oder Zeilen gilt (`rowspan="2"` oder `colspan="2"`)

HTML kennt hierfür mehrere Formen der Verknüpfung mittels Elementen und Attributen, die auch miteinander kombiniert werden können:

- Die nötigen Elemente sind die in 5.1 bereits erwähnten `THEAD`, `TFOOT` und `TBODY` zur Gruppierung von Zellen, und `COL` bzw. `COLGROUP` zur Gruppierung von Spalten.
- Die nötigen Attribute der Tabellen-Elemente sind `scope` und `headers` bzw. `id`, mit denen man Zusammenhänge zwischen Daten festlegen kann.
- In komplexen Tabellen kann der Einsatz des `axis`-Attribut durchaus sinnvoll sein, um weitergehende Verknüpfungen herzustellen. Dabei ist zu beachten, dass Zellen mit `axis`-Attribut gemäß dem HTML-Standard wie Überschriftenzellen behandelt werden (was übrigens aktuelle Screenreader wie JAWS tatsächlich auch so handhaben).

Erst wenn die Daten mit ihren jeweiligen Überschriften programmatisch verknüpft sind lassen sie sich per Skript manipulieren und zum Beispiel dynamisch umsortieren oder filtern, ohne dass hierfür die Seite oder Anwendung komplett neu geladen werden muss.

Ob Sie nun das relativ einfach zu deklarierende `scope` oder die wesentlich aufwändigere Methode mit `headers` und `id` benutzen hängt vom gewünschten Einsatzzweck und der Art der Datentabelle ab – funktional sind beide Methoden gleichwertig. Wenn alle zusammengehörigen Datenzellen in einer Reihe oder Spalte stehen, dann genügt es, in der Überschrift die Verknüpfung per `scope="col"` oder `scope="row"` herzustellen. Wenn die Datenzellen an beliebiger Stelle in der Tabelle stehen oder die Datenzellen sich auf mehrere Überschriftenzellen beziehen, dann müssen Sie auf die Verknüpfung per `headers` und `id` ausweichen. Beide Methoden schliessen einander nicht aus und können auch gemeinsam in einer Datentabelle benutzt werden.

Beispiele:

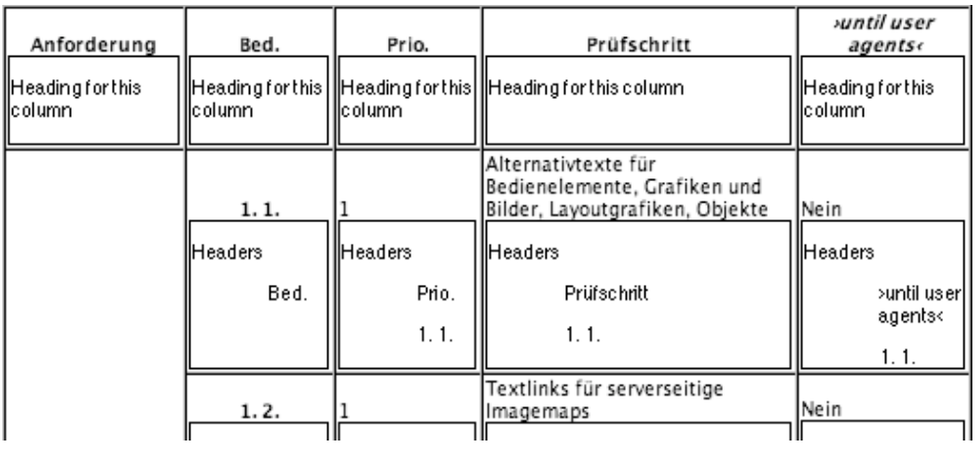
In der folgenden Beispieltabelle zeigt einen Ausschnitt aus der BITV in tabellarischer Form. Es gibt mehrere Ebenen von Überschriften (`TH`): zum einen in der ersten Zeile die Überschriften der darunter angeordneten Spalten, zum anderen in der linken Spalte die Überschriften der jeweils zu einer Anforderung gehörenden Bedingungen.

Anforderung	Bed.	Prio.	Prüfschritt	*›until user agents‹*
Alternativen	**1. 1.**	1	Alternativtexte für Bedienelemente, Grafiken und Bilder, Layoutgrafiken, Objekte	Nein
	1. 2.	1	Textlinks für serverseitige Imagemaps	Nein
	1. 3.	1	Audiodeskription für Multimedia	Ja
	1. 4.	1	Untertitel für Multimedia	Nein
	1. 5	2	Textlinks für cllientseitige Imagemaps	Ja
Farben	**2. 1.**	1	Auch ohne Farbe nutzbar	Nein
	2. 2.	1	Grafiken vor wechselndem Hintergrund erkennbar, Kontraste von Grafiken ausreichend	Nein
	2. 3	2	Wahrnehmung mit Farbfehlsichtigkeiten	Nein

Zur besseren Übersicht finden Sie hier eine separate Datei mit dem Quelltext dieser Tabelle: beispieltabelle.html.

Wie können Sie das testen?

Die Web Developer Toolbar bietet im Menü Information den Befehl ›Display Table Information‹. Mit dieser werden die im obigen Beispiel angeführten Attribute und Elemente unmittelbar im Kontext dargestellt, sodaß Sie die korrekten Verknüpfungen visuell überprüfen können:



Bedingung 5.3: Layouttabellen linearisierbar (Prio. 1)

»Tabellen sind nicht für die Text- und Bildgestaltung zu verwenden, soweit sie nicht auch in linearisierter Form dargestellt werden können.«

Was heißt das?

Der Gnadenstoß für verschachtelte Layouttabellen, da diese üblicherweise nicht sauber linearisiert werden können. Einfache Layout-Tabellen schlüpfen immer noch durch die Lücke, die durch »soweit sie nicht auch in linearisierter Form dargestellt werden können« eröffnet wird. Gemäß dieser Bedingung sind Layout-Tabellen im Geltungsbereich der BITV auch im Jahr 2007 immer noch zulässig, obwohl sie lange nicht mehr notwendig sind.

Da die Verwendung von Layout-Tabellen bei der BIENE 2006 zum **Ausschluss aus dem Wettbewerb** führte, gehen wir hier nicht weiter auf Methoden ein, wie man Tabellen eventuell doch noch zu Layoutzwecken mißbrauchen kann:

BIENE-Prüfschritt 14.1: Keine Layout-Tabellen als Seitengerüst
Prüfen Sie, ob auf den Einsatz von Layout-Tabellen verzichtet wird.

Bedingung 5.4: Kein Tabellenmarkup für Layouttabellen (Prio. 1)

»Soweit Tabellen zur Text- und Bildgestaltung genutzt werden, sind keine der Strukturierung dienenden Elemente der verwendeten Markup-Sprache zur visuellen Formatierung zu verwenden.«

Was heißt das?

Eine veraltete Vorgabe, die Sie ignorieren können. Da Layout-Tabellen generell nicht mehr dem Stand der Technik entsprechen und nicht mehr eingesetzt werden sollten, erübrigt sich die Suche nach fehlerhaft eingesetzten Elementen und Attributen für diese Tabellen.

Bedingung 5.5: (Prio. 2)

»Für Tabellen sind unter Verwendung der hierfür vorgesehenen Elemente der genutzten Markup-Sprache Zusammenfassungen bereitzustellen.«

Was heißt das?

HTML bietet mehrere Elemente, um zusätzliche Informationen zu Tabellen in diesen sichtbar oder unsichtbar zu hinterlegen. Das ist problematisch bei dieser Bedingung: unsichtbare Informationen mit dem dafür vorgesehenen `summary`-Attribut von `TABLE` sind nur für ganz bestimmte Zugangsformen zugänglich und müssen doppelt hinterlegt werden, damit alle Nutzersie wahrnehmen können. Dann kann man auf die unsichtbaren Informationen auch verzichten – ein generelles Problem unsichtbarer oder nur schwer zugänglicher Metadaten.

Das Element `CAPTION`, mit dem innerhalb einer Tabelle eine Art Bildunterschrift hinterlegt werden kann, ist eigentlich nichts anderes als ein strukturierendes Element ähnlich einer Überschrift, die aber nicht über die allgemeine Überschriftenstruktur einer Seite erreichbar ist. Die Praxis hat gezeigt, dass die identischen Informationen besser in einer echten Überschrift (`<h`$_n$`>`) vor der Tabelle stehen, womit dann auf `CAPTION` zur Vermeidung von Dopplern verzichtet werden kann.

Was meint die BIENE?

Auch im BIENE-Prüfverfahren wird dieser Punkt seit dem Jahr 2006 nicht mehr strikt gehandhabt. Es wird lediglich bewertet, ob irgendeine angemessene Form der Beschriftung im Sinne einer Zusammenfassung oder Überschrift gewählt wurde. Dies kann `CAPTION`, `SUMMARY` oder `H`$_n$ sein. In der amerikanischen Section 508, dem dortigen Äquivalent zur BITV wird der Gebrauch von Caption im übrigen nicht vorgeschrieben.

Bedingung 5.6: Verknüpfung von Datenzellen und Überschriften (Prio. 2)

»Für Überschriftenzellen sind unter Verwendung der hierfür vorgesehenen Elemente der genutzten Markup-Sprache Abkürzungen bereitzustellen.«

Was heißt das?

Für diesen Einsatzzweck sieht HTML das Attribut `abbr` vor (nicht zu verwechseln mit dem *Element* `ABBR`). Mit ihm können Sie zusätzliche Informationen hinterlegen, die zum Beispiel in einem nicht-visuellen Medium zusätzlich zu den Inhalten der zugehörigen Datenzellen ausgegeben werden. Ohne diese zusätzlichen Informationen ist es in alternativen Zugangsformen kaum möglich, in großen Tabellen die Daten in Zeile 29, Spalte 5 noch den entsprechenden Überschriften der Spalte oder Zeile zuzuordnen.

Wenn Ihre Zeilen- oder Spaltenüberschriften relativ lange Texte enthalten werden diese von einem Screenreader unter Umständen vor den Inhalten der dazugehörigen Zellen erneut vorgelesen. Im `abbr`-Attribut von `TH` können sie eine Kurzfassung hinterlegen, um Ermüdungserscheinungen beim Nutzer vorzubeugen.

Auch der umgedrehte Weg ist möglich: es könnte sein, dass die Inhalte von Überschriftenzellen nur in visuellen Ausgabeformen sinnbildend sind. Dies kann zum Beispiel der Fall sein, wenn in den Zellen Icons statt echter Texte stehen, wie

dies häufig in der tabellarischen Ansicht von Daten in Web-basierten Anwendungen der Fall ist. Hier kann es sinnvoll sein, zusätzliche Informationen zu Inhalt und Funktion der `TH` per abbr-Attribut zu hinterlegen.

Beispiele:

Ein simples Beispiel für die Bereitstellung von Abkürzungen ist die folgende Tabelle mit der Ansicht eines Kalendermonats. Ziel ist es zu verhindern, dass eine Sprachausgabe bei jedem Datum den jeweiligen Wochentag (Montag, Dienstag, …) vorliest, gleichzeitig soll Nutzern aber trotzdem ein Minimum an Orientierung gegeben werden, ob der 7. des Monats nun ein Samstag oder ein Sonntag ist.

Diese Tabelle besteht aus mehreren Teilen, nämlich dem Kopf der Tabelle mit `<thead>` und `<th>` (Anforderung 5.1) sowie den Datenzellen im `<tbody>` mit `<td>` Zusätzlich sind die Wochentage und das Wochenende in Colgroups gruppiert. Die Überschriftenzellen sind hier wie folgt angelegt:

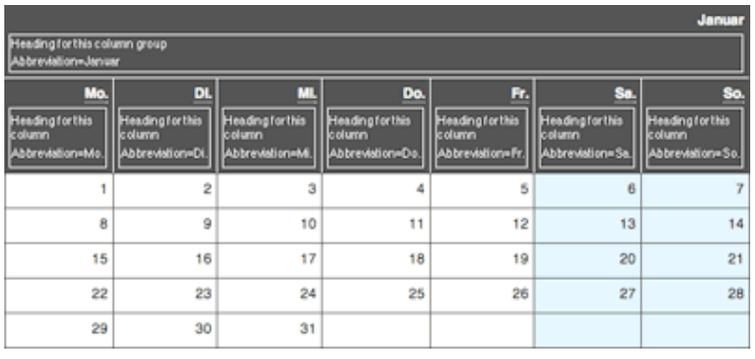
```
<th abbr="Mo." scope="col">Mo.</th>
```

Den vollständigen Quelltext finden Sie in einer Beispieldatei.

Januar						
Mo.	**Di.**	**Mi.**	**Do.**	**Fr.**	**Sa.**	**So.**
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Wie können Sie das testen?

Die Web Developer Toolbar bietet im Menü Information den Befehl ›Display Table Information‹. Mit dieser werden die im obigen Beispiel angeführten Attribute und Elemente unmittelbar im Kontext dargestellt, sodaß Sie die korrekten Verknüpfungen visuell überprüfen können:



Anforderung 6: Nutzbarkeit ohne neuere Techniken

»Internetangebote müssen auch dann nutzbar sein, wenn der verwendete Benutzeragent neuere Technologien nicht unterstützt oder diese deaktiviert sind.«

Was heißt das?

Sie können nicht davon ausgehen, dass alle Benutzer den gleichen Browser wie Sie verwenden. Ein Blick in die Statistik Ihrer Website enthüllt die Vielzahl der unterschiedlichsten Browser aus verschiedenen Generationen, die Ihre Benutzer verwenden.

Der Blick in die Statistik verrät generell nichts über das tatsächliche Können und die Konfiguration der Browser. Viele der Statistiken im Netz erinnern an die Geschichte von dem Mann, der in einem durchschnittlich 12 cm tiefem Fluß ertrunken ist. Auch das beste Statistikprogramm kann Ihnen nicht verraten, ob an einem Zielrechner lediglich ein grafischer Browser läuft, oder ob zusätzlich eine Braillezeile angeschlossen oder eine Sprachausgabe installiert ist. Die gesammelten HTML-Fähigkeiten aller bekannten Browser lassen sich heute nicht mehr auf einem Bierdeckel beschreiben.

Dies betrifft nicht nur die Standarkonfigurationen der Browser und die Frage, ob dieser überhaupt clientseitiges JavaScript, Java und ähnliches ausführen kann. Beim Einsatz von zusätzlichen Plug-In-Formaten sollte sichergestellt sein, dass Inhalte auch dann noch dargestellt und Funktionen ausgeführt werden können, wenn das Endgerät diese Techniken nicht beherrscht.

In Zeiten Web-basierter Anwendungen kann es vorkommen, dass ein Angebot prinzipiell nicht nutzbar sein *kann*, wenn die verwendete Zugangssoftware bestimmte Techniken nicht unterstützt. Der aktuelle Stand der WCAG 2.0 versucht diesem Problem zu begegnen, indem erstmalig bestimmte Grundvoraussetzungen und Basistechniken (sog. ›*baseline assumptions*‹) vorausgesetzt werden dürfen, ohne die eine sinnvolle Nutzung eines Angebots nicht möglich ist. Vor diesem Hintergrund der Weiterentwicklung der Richtlinien ist die Anforderung 6 besonders kritisch zu betrachten, zumal diese Anforderung in einem gewissen Widerspruch zur Bedingung 8.1 steht, wo die direkte Zugänglichkeit dynamischer Inhalte eingefordert wird.

Bei der Überarbeitung des Prüfverfahrens zur BIENE 2006 wurde ein Schwerpunkt darauf gelegt, die ***direkte** Zugänglichkeit* von Angeboten testen zu können. Wenn ein Angebot neuere Techniken im Sinne der Anforderung 6 benutzt und diese in mit zeitgemäßen Zugangsarten barrierefrei nutzbar sind, so verlangen die entsprechenden Prüfschritte keine statischen Alternativen mehr.

Bedingung 6.1: Auch ohne Style Sheets nutzbar (Prio. 1)

»Es muss sichergestellt sein, dass mittels Markup–Sprachen geschaffene Dokumente verwendbar sind, wenn die zugeordneten Style Sheets deaktiviert sind.«

Was heißt das?

Die Erfüllung dieser Bedingung sollte kein Problem sein, wenn Sie in Ihren Seiten den Inhalt sauber von der Verpackung getrennt halten. Allerdings bedingt der Einsatz von CSS, dass Ihre HTML-Dokumente logisch strukturiert sein müssen, um auch ohne CSS noch verwertbar zu sein, wenn Style Sheets nicht oder nur unzureichend unterstützt werden.

Dies betrifft auch Inhalte und Gestaltungselemente, die mehr als Dekoration sind und per CSS statt per HTML eingebunden sind. Techniken hierfür sind zum Beispiel sogenannter ›*Generated Content*‹, bei dem Inhalte aus einer CSS-Datei in ein HTML-Dokument übernommen werden, aber auch CSS-Hintergrundbilder, in denen sich wesentliche Inhalte befinden.

Bestimmte Methoden der Positionierung per CSS sind im Sinne der Barrierefreiheit ebenfalls kritisch zu betrachten. Wenn Elemente frei in einer Seite positioniert sind, kann es bei nachlässiger Umsetzung passieren, dass diese Inhalte in ihrer Abfolge keinen Sinn mehr ergeben, sobald CSS abgeschaltet ist. Dies betrifft insbesondere Fälle, wo das eingesetze Benutzerprogramm sehr nah am Quelltext ›arbeitet‹ und auf eine logische Struktur der Seite angewiesen ist, wie dies generell bei Screenreadern der Fall ist.

Das heisst nicht, dass absolute Positionierung grundsätzlich als problematisch oder barrierebehaftet anzusehen ist. Im Gegenteil, auch mit absoluter Positionierung und relativer Bemaßung kann man ganz hervorragende Layouts umsetzen, die sich sehr gut an Benutzereinstellungen anpassen.

Wie können Sie das testen?

In Anforderung 3 wurde ja bereits das ToggleCSS-Favelet und die Web Developer Toolbar vorgestellt, mit denen Sie sämtliche CSS–Formatierungen einer Seite deaktivieren können. Laden Sie Ihre Seiten und entfernen Sie die Style Sheets. Einzelne Dinge wie CSS-Hintergrundbilder lassen sich auch selektiv mit der Web Developer Toolbar abschalten. Sie sollten bei der Überprüfung besonderes Augenmerk darauf legen, dass die Reihenfolge aller wesentlichen Inhalte und Funktionen einer Seite oder Anwendung *ohne CSS* der Reihenfolge der visuellen Reihenfolge in der Ansicht *mit CSS* entspricht.

Ausnahme von der Regel: dies betrifft nicht Navigationsleisten oder andere Inhalte, die nicht in unmittelbarem Zusammenhang mit dem wesentlichen Inhalt der Seite stehen. Hier sind sie durch verschiedene Methoden der Positionierung vollkommen frei in der Platzierung dieser Inhalte im Quelltext, solange Sie Sprungmarken anbieten, um diese Inhalte gegebenenfalls schnell erreichen zu können.

Noch einfacher geht es, wenn sie lokalen Zugriff auf die zu testenden Dateien haben: verschieben Sie einfach den Ordner, der die Style Sheets enthält. Üblicherweise ist dies das Verzeichnis /css/ in der obersten Ebene der Website. Wenn Sie nun die HTML-Dateien im Browser öffnen, werden die Style Sheets nicht mehr gefunden und das reine HTML angezeigt.

Allerdings stimmt dies nicht so ganz – auch ohne die zugeordneten Style Sheets werden Formatierungen auf Ihre HTML-Seite angewendet: die Browser-eignen Style Sheets. In Firefox finden sie diese zum Beispiel, indem Sie resource:///res/html.css in die Adresszeile des Browsers eintippen.

Problematisch wird diese Bedingung, wenn man sie wortgetreu auf XML-Dokumente wie z. B. RSS anwendet. Auch diese Formate sind »mittels Markup-Sprachen geschaffene Dokumente«, allerdings geben Browser, Feedreader und ähnliche Programme bei XML *ohne zugeordnete* Style Sheets den rohen Quelltext aus.

Beispiele:

Was alles schief gehen kann, wenn Inhalt, Funktion und Verpackung nicht sauber von einander getrennt wurden, sehen Sie bei einer älteren Version der Seiten der Bundeswehr. Wenn man hier die Style Sheets deaktiviert, erhielt man azurblaue Links auf dunkelblauem Hintergrund. Ursache hierfür war, dass die Vordergrundfarbe der Links in einem externen Style Sheet deklariert wurde, die Hintergrundfarbe jedoch im HTML-Dokument selber über die Hintergundfarbe einer Tabellenzelle.

Bedingung 6.2: Äquivalente für dynamische Inhalte (Prio. 1)

»Es muss sichergestellt sein, dass äquivalente für dynamischen Inhalt aktualisiert werden, wenn sich der dynamische Inhalt ändert.«

Was heißt das?

Kurzfassung: das wissen wir nicht.

Langfassung: die Dokumentation zu dieser Bedingung bzw. zum Original in den Web Content Accessibility Guidelines rangiert zwischen *dürftig* und *an den Haaren herbeigezogen*. Veröffentlichungen zum Thema versuchen das Problem vermeintlich elegant zu umschiffen, indem diese Bedingung kurzerhand nicht erwähnt wird, oder lediglich kurzer Bezug auf die kaum brauchbaren Besipiele in den Dokumenten der WAI genommen wird.

All dies hat dazu geführt, dass dieser Punkt in der amerikanischen Section 508 nicht zu finden ist, da sich selbst das US Access Board nicht sicher war, was genau das W3C mit dieser Richtlinie gemeint hat: »the meaning of the provision is unclear« (zitiert nach Thatcher, Jim et al., *Accessible Web Sites*, S. 346).

Bedingung 6.3: Auch ohne Skripte nutzbar (Prio. 1)

»Es muss sichergestellt sein, dass mittels Markup–Sprachen geschaffene Dokumente verwendbar sind, wenn Scripts, Applets oder andere programmierte Objekte deaktiviert sind.«

Was heißt das?

Auch im aktuellsten Browser kann unter Umständen JavaScript abgeschaltet sein, sei es, weil eine Sicherheitsrichtlinie in einem Firmennetzwerk dies verlangt, oder auch einfach nur um lästige Werbeeinblendungen zu verhindern. Im Kontext dieser Richtlinie spielen solche Anwendungsfälle keine Rolle, da es hier ausnahmslos um den Zugang für Menschen mit Behinderung geht.

Für moderne Hilfsmittel wie halbwegs aktuelle Screenreader sind clientseitige Skripte meistens unproblematisch, sofern die Aspekte der barrierefreien Webentwicklung berücksichtigt wurden. Probleme treten hier eher aufgrund der ungewohnten Dynamik von Web-basierten Anwendungen auf. Diese Hürden lassen sich nicht durch Abschalten von JavaScript beseitigen, hierfür sind andere Bedingungen der BITV relevant, die in den Bereich der Gebrauchstauglichkeit (*Usability*) speziell für Menschen mit Behinderung gehen.

Trotzdem sollte beim Einsatz von Java, JavaScript und Flash darauf geachtet werden, dass diese nur optional sind, *sofern Sinn und Zweck der Anwendung diese Einschränkung zulassen*. Das im Fall von JavaScript für statische Alternativen vorgesehene `NOSCRIPT`-Element kann hier nur einen unzureichenden Ersatz bieten. Zum einen sind die Inhalte von `NOSCRIPT`

selbst nicht änderbar, ohne dass man eine Seite neu lädt (was wiederum den Einsatz anderweitig verbotener Mittel voraussetzen würde). Zum anderen kann man in `<noscript>` zwar einen Verweis auf eine statische Alternative ohne Skript-Funktionalitäten setzen – aber spätestens auf dieser Seite stellt sich dann das Problem, wie man adäquate statische Äquivalente für Inhalte bereitstellt, die ohne ein Mindestmaß an Dynamik prinzipiell nicht funktionieren *können*.

Einen Ausweg aus diesem Dilemma bietet die Bedingung 8.1, mit der die *direkte* Zugänglichkeit von Scripts, Applets oder anderen programmierte Objekten ermöglicht wird. Gerade aus Sicht von behinderten Menschen und damit im Sinne des barrierefreien Webdesigns ist letztere Variante sicher die attraktivere, weil sie keine Sonderwege festschreibt, sondern die integrierte Nutzung ermöglicht.

Wie geht es trotzdem?

Einen brauchbaren Ansatz, um diese beiden Bedingungen zu vereinen, bieten die Prinzipien der *»Graceful Degradation«* und des *»Progressive Enhancement«* bzw. *»Hijax«*. Die Begriffe beschreiben die schrittweise Verbesserung einer zunächst statischen Web-Anwendung durch Ajax.

Die dahinter stehenden Techniken sind nicht wirklich neu: XML, ECMAScript (vulgo: JavaScript) und das XMLHTTP-Objekt sind robuste Techniken, die von allen modernen grafischen Browsern unterstützt werden. Diese Techniken sind standardisiert oder befinden sich im Prozess der Standardisierung durch das W3C.

Stark verkürzt auf den Punkt gebracht:

- Planen Sie Ajax von Anfang an mit ein.
- Implementieren Sie Ajax erst am Ende.

Alles Wissenswerte zu diesem Thema finden sie in unserem Podcast vom 14. Juni 2006 zum Thema »Barrierefreiheit, Ajax und Web2.0«.

Positive Beispiele:

Ein hervorragend gemachtes Beispiel für diese Technik finden Sie bei der Amazon.com Diamond Search:

- **Mit aktiviertem JavaScript** erhalten Sie eine interaktive Anwendung inklusive der in HTML bisher nicht vorgesehenen und somit ohne JavaScript unmöglichen Schiebereglern (engl.: *Slider*). Bereits während der Bedienung der Schieberegler schickt der Server eine Liste der zu erwartenden Treffer auf Basis der ausgewählten Kriterien.
- **Ohne JavaScript** erhält man ein herkömmliches Formular, das zwar prinzipiell die gleiche Suche ermöglicht, jedoch mit den üblichen in HTML vorgesehenen einfacheren Formularelementen, dies jedoch ohne den zusätzlichen Komfort der AJAX-Version.

Universeller Fensteröffner

Damit Sie auch bei dieser Bedingung etwas zum Mitnehmen haben hier ein Programmierbeispiel für eine Funktion, die bei eingeschaltetem JavaScript Links in neuen Fenstern öffnet, deren Aussehen und Position Sie frei bestimmen können. Das Skript benötigt keine Inline-Event-Handler (`onclick` o.ä.) im HTML, sondern ›beobachtet‹ von aussen die Interaktionen mit den Links (`a` mit Attribut `href`) und fängt Klicks auf Elemente mit ganz bestimmten Attribut-Wert-Paaren ab. Ohne JavaScript bleiben die Links das, was sie schon immer waren: *ganz normale Links*.

Der Einfachheit halber haben wir das Skript in die folgende Datei eingebettet: fensteroeffner.html. Im Regelbetrieb sollten Sie Skripte wie diesen in externe .js-Dateien auslagern.

Applet und Object

Nicht nur für Skripte muss die direkte Zugänglichkeit sichergestellt werden oder zumindest für Alternativen gesorgt werden, sondern auch für »Applets oder andere programmierte Objekte«. Der Vorteil des (allerdings veralteten) `APPLET` und des moderneren `OBJECT`-Elementes sind die eingebauten Alternativen für den Fall, dass ihre Inhalte nicht unterstützt werden oder aus anderen Gründen nicht augeführt werden können. `<applet>` kann mit einem `alt`-Attribut versehen werden und kann zwischen Start- und End-Tag auch ganz normales HTML enthalten, das (zumindest in der Theorie) angezeigt wird, wenn z. B. Java deaktiviert ist.

`OBJECT` bietet darüber hinaus den Vorteil, dass man mehrere Elemente mit unterscheidlichen Inhalten ineinander verschachteln kann. Die Browser versuchen zunächst das äussere `<object>` auszuführen, schlägt dies fehl, wird das nächste innere `<object>` genommen usw.:

```
<object classid="java.class" width="300" height="200">
 <object data="film.mp4" type="video/mpeg">
  <object data="bild.png" type="image/png">
   <object data="bild.gif" type="image/gif">
    Text als letzter Notanker
   </object>
  </object>
 </object>
</object>
```

Negative Beispiele:

Bei aller Begeisterung für die Möglichkeiten Web-basierter dynamischer Anwendungen gilt es, einige grundlegende Dinge zu beachten:

- Viele Seiten im Netz benutzen Menüs, die beim Überfahren mit der Maus ein Untermenü aufklappen. Um dies zu erreichen, ist normalerweise ein buntes Gemisch aus HTML, JavaScript und CSS nötig. Wenn nun JavaScript im Zielbrowser deaktiviert ist, klappt nichts auf und dem Benutzer bleiben wesentliche Teile der Navigation verborgen. Das heißt aber nicht, dass Sie auf diese Ausklappmenüs verzichten müssen, oftmals sind diese die bessere Lösung, um komplexe Hierarchien abzubilden. Seit geraumer Zeit gibt es im Netz gut dokumentierte Lösungen, die auch mit abgeschaltetem JavaScript funktionieren oder gänzlich ohne auskommen und zudem mit der Taststur bedienbar sind.
- Unbedingt vermeiden sollten Sie Methoden, die statische Inhalte per JavaScript in Ihre Seiten schreiben. Ähnlich wie bei generiertem Content per CSS (Bedingung 6.1) haben Sie keine Kontrolle darüber, ob der Nutzer diese Inhalte tatsächlich wahrnehmen kann.
- Vermeiden sollten Sie Links, deren Wert keine gültige Webadresse (URL) ist, sondern ein JavaScript-Schnipsel. Diese werden oft benutzt, um neue Fenster zu öffnen und zusätzliche Parameter für diese zu definieren.

Wie können Sie das testen?

Auch diese Bedingung kann relativ einfach in den gängigen Browsern getestet werden. In den Schnelleinstellungen von Opera geht dies noch am einfachsten: hier können Sie auch ohne Browser-Erweiterungen direkt über einen Menübefehl Java, JavaScript und Plug-Ins ausschalten. In anderen Browsern müssen Sie hierfür unter Umständen die Voreinstellungen ändern, oder Sie benutzen die hinlänglich bekannten Toolbars für Webentwickler, die es mittlerweile nicht mehr nur für Firefox, sondern auch für den Internet Explorer und Opera gibt.

Deaktivieren Sie in diesen alle optionalen Elemente (also Java, JavaScript und Plug-Ins) und testen Sie, ob noch alle Inhalte (nun jedoch in anderer Form) vorhanden, erreichbar und verständlich sind und ob es für alle ausführbarem Funktionen eine *Fallback*-Lösung gibt. Dies gilt insbesondere für die Navigation und andere Links sowie Formulare.

Bedingung 6.4: Eingabe geräteunabhängig (Prio. 1)

»Es muss sichergestellt sein, dass die Eingabebehandlung von Scripts, Applets oder anderen programmierten Objekten vom Eingabegerät unabhängig ist.« <zitat>

Was heißt das?

Sie können nicht davon ausgehen, dass alle Nutzer Ihre Website lediglich mit der Maus bedienen. Streng genommen sind sowohl Tastatur wie auch Maus ›Hilfsmittel‹ zur Bedienung des Computers; im Kontext dieser Verordnung geht es jedoch ausschließlich um die äußerst vielfältigen Formen der Bedienung und Eingabe, wie sie typischerweise von Menschen mit bestimmten Behinderungen benutzt werden. Wenn diese behinderungskompensierenden Werkzeuge nicht berücksichtigt werden, z. B. weil ein Angebot nur mit ganz bestimmten Eingabegeräten zu bedienen ist, dann entstehen an dieser Stelle für viele Menschen mit Behinderung unüberwindbare Barrieren.

Beispiele:

In der praktischen Auswirkung ist diese Bedingung eng verwandt mit Bedingung 8.1 und Bedingung 9.2 bzw. 9.3, daher haben wir die Diskussion der Prinzipien und der notwendigen Techniken in diesen Bedingungen zusammengefasst.

Bedingung 6.5: Dynamische Inhalte zugänglich (Prio. 1)

»Dynamische Inhalte müssen zugänglich sein. Insoweit dies nur mit unverhältnismäßig hohem Aufwand zu realisieren ist, sind gleichwertige alternative Angebote unter Verzicht auf dynamische Inhalte bereitzustellen.«

<zitat>

Was heißt das?

Eine in Teilen problematische Bedingung, bei der zudem die Abgrenzung zu Bedingung 8.1 und Bedingung 11.3 (=WCAG 11.4) nicht klar ist. Die Schwierigkeiten liegen hier in der Festlegung der Grenzen von »unverhältnismäßig hohem Aufwand« und ab wann statische Alternativen als »gleichwertige alternative Angebote« gelten können. Auch die Veröffentlichungen zum Thema wie der Styleguide der Bundesregierung, ja sogar die Techniken der W3C WAI zeigen Beispiele zur Erfüllung dieser Bedingung, die:

a. *sachlich falsch* sind,
b. *technisch unmöglich* und
c. *inhaltlich anderen Bedingungen zuzuordnen* sind.

Ein vereinzelt anzutreffendes Hilfsargument zur Stützung dieser Bedingung ist, dass Textbrowser keine Frames verstehen. Dies war schon zum Erscheinungsdatum dieser Verordnung nicht mehr korrekt und geht zudem von der irrigen Annahme aus, dass Menschen mit Behinderung generell mit Textbrowsern im Netz unterwegs seien – das genaue Gegenteil ist jedoch der Fall.

Zudem schliesst die BITV selbst ausdrücklich alternative Textversionen als Sonderwege aus, sodaß diese Bedingung insgesamt als veraltet angesehen werden kann.

Anforderung 7: Dynamik und Bewegung

»Zeitgesteuerte Änderungen des Inhalts müssen durch die Nutzerin/ den Nutzer kontrollierbar sein.«

Was heißt das?

Das Web ist mehr als eine Sammlung strukturierter statischer Texte. Animationen erreichen manche Nutzergruppen besser und transportieren Inhalte. Web-basierte Anwendungen und dynamische Inhalte eröffnen neue Möglichkeiten, die weit über statische Dokumente hinausgehen, die zur Entstehungszeit der BITV der Stand der Technik waren. Diese dynamischen Elemente müssen sorgfältig eingesetzt werden, um eine zugängliche Website oder Webanwendung zu erstellen.

Mit »Zeitgesteuerte Änderungen des Inhalts« können sowohl automatische Weiterleitungen zu einer anderen Seite als auch flackernde oder blinkende Inhalte gemeint sein. In diese Kategorie fallen auch bewegte und periodisch aktualisierende Inhalte innerhalb einer Seite, sei es per eingebettetem Video, JavaScript oder Flash sowie Live-Ticker z. B. bei Börsenkursen und in der Sportberichterstattung. Die Anforderung bezieht sich auf die Bedürfnisse vieler unterschiedlicher Nutzergruppen.

Ist diese Anforderung noch aktuell?

Die gesamten Bedingungen der BITV-Anforderung 7 sind im Original, den Web Content Accessibility Guidelines (WCAG 1.0) sogenannte »Until user agents…«-Guidelines. Das bedeutet, daß die ihnen zu Grunde liegenden Barrieren im historischen Kontext bewertet werden müssen: aus damaliger Sicht waren viele der angesprochenen Probleme unüberwindliche Hürden für Nutzer und die damals gebräuchlichen Browser und assistiven Programme. *Aber sind sie es heute noch?*

Bestimme problematische Techniken des Webdesigns konnten die damaligen User Agents nicht unterdrücken oder so umwandeln, dass sie keine Hürde mehr darstellten – die Benutzer solcher Browser mussten einfach damit leben.

Anmerkung: Die folgenden Checkpunkte gelten, bis Benutzeragenten (einschließlich assistiver Technologien) sich dieser Probleme annehmen. Diese Checkpunkte sind als "vorläufig" eingestuft, was bedeutet, dass die Arbeitsgruppe für Web-Inhalt-Richtlinien sie als gültig und für die Web-Zugänglichkeit notwendig betrachtet zum Zeitpunkt der Veröffentlichung dieses Dokuments (Anm.: 5. Mai 1999). Die Arbeitsgruppe erwartet jedoch nicht, dass diese Checkpunkte in der Zukunft nötig sein werden, wenn vorweggenommene Features und Fähigkeiten Teil von Web-Technologien geworden sind.«

Daher wurde in den WCAG 1.0 diese Einschränkung eingeführt, dass die Richtlinien so lange zu gelten haben, bis die behandelten Barrieren nicht mehr bestehen. Die Web Accessibility Initiative meinte schon damals:

In den meisten Checkpunkten werden Entwickler von Inhalten aufgefordert, für die Zugänglichkeit ihrer Seiten und Sites zu sorgen. Es gibt jedoch Zugänglichkeitserfordernisse, die besser von Benutzeragenten erfüllt würden (einschließlich assistiver Technologien).«

Diese Einschränkungen sind *nicht*, auch nicht durch die Web Accessibility Initiative selbst, getestet und dokumentiert worden. Bei der Erarbeitung der BITV wurde auf den Halbsatz »Until user agents…« verzichtet. Zum Ersatz wurde in der Begründung zur BITV klargestellt:

7. Zu Nr. 10 der Anlage 1:
Die Sicherstellung der Verwendbarkeit assistiver Technologien und Browser ist insbesondere dann unverhältnismäßig, wenn die assistiven Technologien und Browser älter als drei Jahre sind und der Verbreitungsgrad in der einschlägigen Benutzergruppe unter 5 % liegt.

Das Problem ist für Webentwickler noch nicht gelöst, sondern nur auf eine andere Baustelle verlagert. Wer stellt fest, dass »die assistiven Technologien [sic] und Browser älter als drei Jahre sind und der Verbreitungsgrad in der einschlägigen Benutzergruppe unter 5 % liegt«? Der Gesetzgeber hat dies bisher, 5 Jahre nach Inkrafttreten der Verordnung, nicht getan, obwohl seitdem mehrere Generationen Browser und Screenreader erschienen sind und die BITV eine Möglichkeit zur Aktualisierung vorsieht, sobald »die Verfügbarkeit völlig neuer Web-Technologien und Tools, die das Problem der Barrierefreiheit fundamental berühren« vorliegt.

Auch von der Web Accessibility Initiative des W3C ist kaum Hilfe zu erwarten; man konzentriert sich auf die Version 2.0 der WCAG. Zwar wurde vor geraumer Zeit eine Testreihe unter dem Titel »User Agent Support for Accessibility« begonnen, dieses Dokument liegt aber seit 2003 brach.

Was meint die BIENE?

Im BIENE-Prüfverfahren gibt es eine ganze Reihe von Ausnahmen bei Prüfschritten, die mit dieser Anforderung korrespondieren. So werden Inhalte, in denen Bewegung, Aktualisierungen oder andere Formen von Dynamik grundlegend für das Angebot sind, in Prüfschritt 16.2 ausdrücklich ausgenommen:

<zitat>

16.2 Vermeidung oder Kontrollierbarkeit von periodischen Aktualisierungen

Prüfen Sie, ob periodische Aktualisierungen innerhalb einer Seite vermieden werden oder durch die Nutzerin/ den Nutzer kontrollierbar sind.

Ausnahme:

Prüfungssituationen, Live-Indikatoren oder Fälle, in denen überraschende Elemente grundlegend für die Anwendung sind.

Bedingung 7.1: Verzicht auf Flackern (Prio. 1)

<zitat>

»Bildschirmflackern ist zu vermeiden.«

Was heißt das?

Bei Menschen mit photosensitiver Epilepsie können in bestimmten Frequenzen flackernde oder oszillierende Inhalte einen epileptischen Anfall auslösen. Diese Anfälle können in permanenten Hirschädigungen enden, daher ist hier besondere Vorsicht geboten. Photosensitive Epilepsie betrifft ca. 3 – 5 % der Menschen mit Epilepsie, betroffen sind hauptsächlich Kinder und Jugendliche, am häufigsten tritt dies im weiblichen Teil der Bevölkerung auf.

Frequenzen zwischen 16 und 25 Hz gelten als problematisch, in der Literatur findet man aber auch Hinweise auf den Frequenzbereich von 3 bis ca. 60 Hz. Sie sollten schnelle Animationen oder zu schnelle Wechsel von Hell und Dunkel in einem Frequenzbereich um 20 Hertz vermeiden oder zumindest Warnhinweise vorschalten. Dies gilt für animierte GIFs und alle sonstigen eingebetteten Formate wie Flash oder Java. Eine Reihe von Beispielen finden Sie bei NCAM: »Examples of Flickering Images«.

Nach unserem Kenntnisstand gibt es jedoch *keinen* dokumentierten Fall, bei dem Webinhalte bei Menschen mit photosensitiver Epilepsie einen Anfall ausgelöst hätte. Im Netz findet man einige Hinweise auf eine japanische Fernsehserie, die bei Kindern eine Reihe Epilepsie-ähnlicher Symptome hervorgerufen haben soll. Auch hier sind sich die Gelehrten nicht einig, wodurch die Symptome tatsächlich hervorgerufen wurden, einige Veröffentlichungen sprechen von Faktoren, die nicht direkt mit den Inhalten selbst in Verbindung gebracht werden können.

Ist diese Anforderung noch aktuell?

Auch diese Bedingung basiert auf einer »Until user agents…«-Richtlinie der WCAG, d. h. das Original gibt bereits deutliche Hinweise darauf, dass diese Problematik eigentlich in der Zugangssoftware gelöst werden sollte – für diesen Zweck hat das W3C die User Agent Accessibility Guidelines (UAAG) geschaffen.

Mittlerweile sind auch alle gängigen Browser mit ihren Bordmitteln oder auch per Erweiterung in der Lage, die in dieser Bedingung als problematisch angesehenen Techniken zu unterbinden. Einzelne Browser können steuern, dass zum Beispiel bestimmte PlugIns nicht ausgeführt werden, Animationen unterdrückt werden etc. Selbst der Internet Explorer bietet entsprechende Möglichkeiten: im Menü ›Extras‹ finden Sie den Befehl ›Internet-Optionen‹, im folgenden Dialog lässt sich unter der Registerkarte ›Erweitert‹ im Bereich ›Multimedia‹ die Einstellung ›Animationen in Webseiten wiedergeben‹ deaktivieren.

Wie können Sie das testen?

Flackernde Inhalte in HTML-Dokumenten können mit `SCRIPT`-, `OBJECT`-, `EMBED`-, `APPLET`-Elementen sowie `IMG`-Elementen vom Typ .gif erzeugt werden. Je nach verwendetem Browser können diese generell in den Einstellungen unterdrückt werden; für unsere Tests greifen wir wieder auf die Web Developer Toolbar für Firefox zurück. Im Menü ›Disable‹ können Sie Java & JavaScript abschalten, im Menü ›Images‹ können selektiv auch lediglich animierte Bilder unterdrückt werden.

Bedingung 7.2: Verzicht auf Blinken (Prio. 1)

<zitat>

»Blinkender Inhalt ist zu vermeiden.«

Was heißt das?

Blinkende Inhalte, d. h. Inhalte, die sich in regelmäßigen Zeitabständen ändern, lenken vom Lesen eines Textes ab, da sie sehr viel (unbewusste) Aufmerksamkeit auf sich ziehen. Auch blinkende Inhalte, die nicht im primären Gesichtsfeld des Nutzers

liegen, können ausgesprochen störend wirken. Daher ist dieser Prüfschritt auch z. B. auf Werbebanner oder andere blinkende Inhalte anwendbar, die nicht den Fließtext unterbrechen, sondern in einer Marginalspalte platziert sind.

Ist diese Anforderung noch aktuell?

Auch diese Bedingung basiert auf einer »Until user agents…«-Richtlinie der WCAG, d. h. das Original gibt bereits deutliche Hinweise darauf, dass diese Problematik eigentlich in der Zugangssoftware gelöst werden sollte – für diesen Zweck hat das W3C die User Agent Accessibility Guidelines (UAAG) geschaffen.

Alle gängigen Browser sind inzwischen mit ihren Bordmitteln oder auch per Erweiterung in der Lage, die in dieser Bedingung als problematisch angesehenen Techniken zu unterbinden. Das in den Techniken der WCAG als Beispiel hierfür beschriebene Netscape-eigene Element `<blink>` wird zwar noch von Firefox unterstützt, ist aber eine kurzfristige Modeerscheinung des vergangenen Jahrhunderts und darf somit zu den Akten gelegt werden. BITV-verpflichtete Anbieter dürfen den `<blink>`-Tag sowieso nicht verwenden, da dies nie Bestandteil irgendeiner W3C-Spezifikation war. Auch die CSS-Alternative `text-decoration: blink` wird heutzutage von keinem ernsthaften Webdesigner benutzt.

Alternativ wird diese Funktion auch von Werbeblockern bereitgestellt, in vielen Browser lassen sich diese Einstellungen auch für einzelne Webangebote setzen; ähnliche Funktionen kann ein User Style Sheet oder eine Erweiterung wie FlashBlock bieten. In Opera ist diese Funktion zur Abschaltung von Animationen zusätzlich auch über die Schnelleinstellungen direkt erreichbar. Nach dem Wortlaut der WCAG müsste also der Zustand bereits erreicht sein, ab dem diese Bedingung als obsolet gelten kann.

Die Bewertung dieser Bedingung ist bei werbefinanzierte Angeboten problematisch: diese sind schon fast automatisch nicht BITV-konform, wenn sie die Platzierung animierter Werbeinhalte zulassen. Gerade *weil* blinkende Inhalte so viel Aufmerksamkeit auf sich ziehen werden diese Techniken in der Werbung eingesetzt. Die Bewertung erschwert die Tatsache, dass Werbebanner üblicherweise von zentralen Vermarktern geliefert werden und der Anbieter einer Site hat keinen Einfluss darauf, welche Inhalte mit welcher Technik zu welcher Zeit auf seinen Seiten erscheinen.

Wie können Sie das testen?

In vielen Browsern können die Animation von GIFs über die Voreinstellungen unterbunden werden. Während die entsprechende Option in Mozilla direkt in den Voreinstellungen gemacht werden kann, ist sie bei Firefox leider wie so vieles in das wenig intuitive about:config gewandert. Wenn Sie hier nach image.animation_mode filtern können Sie den Wert von normal auf none ändern und damit GIFs das Animieren untersagen.

Tipp: wenn Sie wichtige Informationen in einem animierten GIF ablegen, so sollten diese bereits im ersten Frame der Animation stehen. Diesen Frame können Sie natürlich so anlegen, dass er nur einige Millisekunden angezeigt wird, und danach dann die eigentliche Animation kommt. So verhindern Sie, dass Benutzern, die zwar Bilder anzeigen, aber Animationen unterbunden haben, wichtige Informationen vorenthalten werden.

Bedingung 7.3: Verzicht auf bewegte Inhalte (Prio. 1)

<zitat>

»Bewegung in mittels Markup–Sprachen geschaffener Dokumente ist entweder zu vermeiden oder es sind Mechanismen bereitzustellen, die der Nutzerin/dem Nutzer das Einfrieren der Bewegung oder die Änderung des Inhalts ermöglichen.«

Was heißt das?

Gerade bei Animationen, in denen sich grössere Mengen an Inhalten befinden, können Sie nicht davon ausgehen, dass diese von allen Besuchern in der gleichen Geschwindigkeit verarbeitet werden können. Menschen mit Leseschwäche, Lernbehinderung oder Nicht-Muttersprachler sollten das Tempo einer Texte enthaltenden Animation kontrollieren können. Das Erscheinenlassen oder Verstecken von Inhalten kann eine Hürde bedeuten, wenn diese Abläufe nicht durch den Nutzer kontrollierbar sind.

Bei den längst aus der Mode gekommenen Newstickern, die wie ein vertikales oder horizontales Laufband Nachrichtenschnipsel über den Bildschirm schieben, lässt sich diese Bedingung noch relativ einfach umsetzen. Für diese Ticker können Java-Applets oder JavaScript, oder der MS IE-eigene `<marquee>`-Tag verwendet werden (auf letzteren sollten Sie allein schon deswegen verzichten, weil dieser überhaupt nur vom Internet Explorer verstanden wird und auch nie Teil der HTML-Empfehlungen war).

Bei bewegten Inhalten in Flash oder Java-Applets sollten Sie für den Benutzer eine gleichwertige Funktion bereitstellen, mit der dieser die Kontrolle über Ablauf und Geschwindigkeit der Bewegungen erhält. Beim Java Accessibility Program der University of Wisconsin finden Sie ein Programmierbeispiel, das zeigt, wie Nutzer Bewegungen abschalten können.

Vorschicht, Falle!
Wie auch schon an anderer Stelle bemerkt wurde ist hier die Übersetzung der WCAG in die BITV nicht korrekt: es geht *nicht* darum, wie es in der BITV steht, dem Nutzer »die Änderung des Inhalts« zu ermöglichen, sondern das *Einfrieren* der **Änderung des Inhalts**.

Wie können Sie das testen?

Bei Animationen mit DHTML ist sowohl der Test als auch die Bedingung selbst einigermaßen schwer umzusetzen. Wenn Sie die übliche Kombination aus HTML, CSS und JavaScript einsetzen, um Inhalte über den Bildschirm zu bewegen, hilft nur das Abschalten von JavaScript, um diese Bewegung zu verhindern. Auch animierte GIFs können nicht durch Interaktion mit dem Objekt selbst, sondern nur durch einen Eingriff in die Voreinstellungen des Browsers gestoppt werden.

Bedingung 7.4: Verzicht auf Auto-Aktualisierung (Prio. 1)

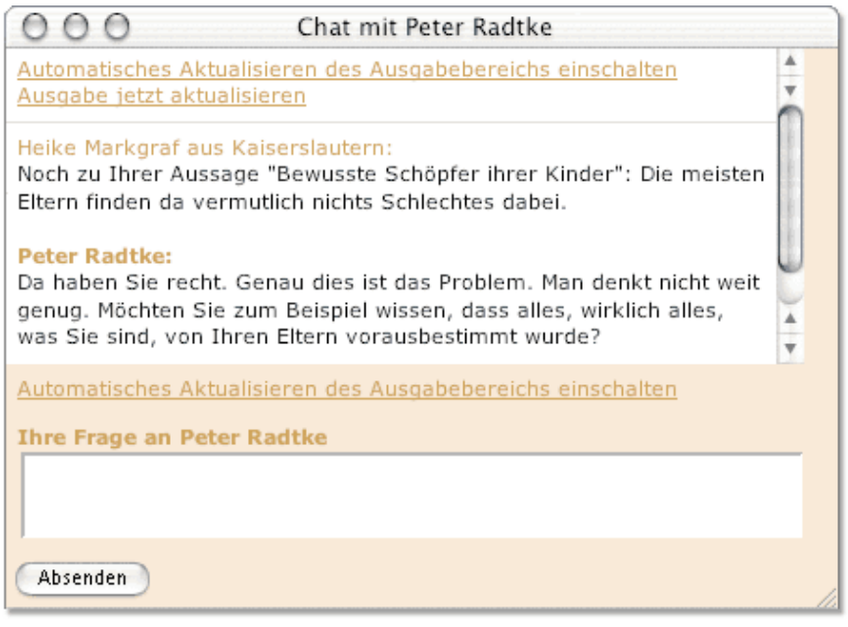
»Automatische periodische Aktualisierungen in mittels Markup–Sprachen geschaffener Dokumente sind zu vermeiden.«

Was heißt das?

Diese Bedingung erweitert die zuvor gemachten Vorgaben um sämtliche Formen der Aktualisierung in allen Arten von Dokumenten oder Anwendungen, die auf Markup-Sprachen wie HTML und XHTML, aber auch SVG und ähnlichem basieren. Die Techniken zu den WCAG 1.0 listen an dieser Stelle lediglich einige Besipiele, die mit den ebenfalls in Bedingung 7.5 behandelten sogenannten Meta-Refreshes umgesetzt werden. Mit »automatischen periodischen Aktualisierungen« könnten aber auch Live-Ticker wie bei Börsenkursen und in der Sportberichterstattung gemeint sein, wie auch Werbebanner, die per Meta-Refresh, AJAX oder in Iframes ausgetauscht werden. Diese Technik ist problematisch bei Seiten, die nach mehreren Minuten erneut laden, um andere Werbebanner zu zeigen. In diesem Fall springt der Fokus im Screenreader wieder zum Beginn der Seite, oder, je nach Einstellung, zum Ort der Änderung, auch wenn der Nutzer gerade mit etwas ganz anderem beschäftigt war.

Unter diese Bedingung fallen auch Echtzeit-Chats, in denen sich das Ausgabefenster in bestimmten Intervallen selbsttätig aktualisiert. Diese Chats können auf verschiedenen Client-seitigen Techniken wie Java basieren oder auch komplett in reinem HTML-Markup ablaufen. Andererseits stellt sich auch hier, wie bei allen Bedingungen der Anforderung 7 die Frage nach der Verantwortung der Hilfsmittelhersteller und der Eigenverantwortung ihrer Kunden: ein Chat, der sich nicht dann aktualisiert wenn jemand etwas zur Diskussion beiträgt, ist kein wirklich sinnvoller Chat. Ähnlich abwegig ist es wohl auch, die automatische Aktualisierung von Börsenkursen oder Sportergebnissen zu untersagen.

Die Aktion Mensch nutzt ein solches Chat-System im Rahmen Ihrer Ethik-Kampagne unter 1000fragen.de, wo in regelmäßigen Abständen Besucher– und Experten-Chats angeboten werden. Die nach intensiven Tests hierfür gefundene beste Lösung ist ein rein HTML-basiertes System, in dem der Besucher selbst zwischen automatischer und manueller Aktualisierung wählen kann:



Periodische Aktualisierungen können aber auch durch Meta-Refreshes in HTML-Dokumenten, Frames oder eingebetteten Frames (`iframes`) erreicht werden. Um diese Technik kümmert sich die nächste Bedingung.

Bedingung 7.5: Verzicht auf Weiterleitung (Prio. 1)

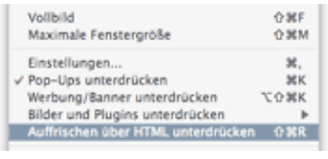
<zitat>

»Die Verwendung von Elementen der Markup–Sprache zur automatischen Weiterleitung ist zu vermeiden. Insofern auf eine automatische Weiterleitung nicht verzichtet werden kann, ist der Server entsprechend zu konfigurieren.«

Was heißt das?

Die weit verbreitete Technik Meta-Refresh zur automatischen Weiterleitung einzusetzen ist für viele Benutzer verwirrend und kann gerade bei der Verwendung assistiver Programme desorientierend wirken. Besonders wenn die Zeitspanne für die Auslösung der Weiterleitung grosser als 0 Sekunden ist, lesen Screenreader den Beginn des Inhalts und brechen ab, sobald die Weiterleitung ausgelöst wird.

Nach dem Wortlaut dieser Bedingungen sind Weiterleitungen, die mit *anderen* Mitteln als durch die »Verwendung von Elementen der Markup-Sprache« durchaus legitim. Dies liegt unter anderem daran, dass diese Weiterleitungen bzw. deren Unterbindung leichter durch den Nutzer kontrollierbar sind. Auch diese Bedingung basiert auf einer »Until user agents…«-Richtlinie der WCAG, d. h. das Original gibt bereits deutliche Hinweise darauf, dass diese Problematik eigentlich in den User Agents gelöst werden sollte.



Tatsächlich bieten die meisten verbreiteten User Agents mittlerweile in der Werkseinstellung oder per Erweiterung nachrüstbare Möglichkeiten, diese per HTML ausgelösten automatischen Weiterleitungen zu unterbinden. Somit wäre diese Bedingung strenggenommen hinfällig.

Zudem ist das strikte Verbot von Weiterleitungen (wie auch zuvor schon bei Aktualisierungen) in vielen Anwendungsszenarien nicht haltbar, insbesondere wenn diese Weiterleitung oder Aktualisierung ein funktionaler Bestandteil der Anwendung ist. Ein Beispiel ist häufig in Angeboten zu finden, die komplexere Datenbankabfragen oder Abfragen bei Dritten beinhalten. Als Beispiel sei hier nur die Validierung von Kreditkartennummern am Ende eines Einkaufs genannt. Hier gibt es keine andere Möglichkeit als die periodische Aktualisierung, um den Nutzer über den aktuellen Zustand seiner Abfrage zu informieren und nach erfolgter Überprüfung auf eine andere Seite weiterzuleiten.

Wie können Sie das testen?

Wenn Sie bei der Überprüfung Ihrer Seiten auf Texte wie diesen stoßen:

»Wenn Ihr Browser automatische Weiterleitung unterstützt, werden Sie in 5 Sekunden weitergeleitet. Ansonsten klicken Sie hier: www.foo.de/bar.«

dann setzt Ihr Webangebot genau diese Technik ein. Das dafür verantwortliche Tag steht im Quelltext des Dokumentes oder der Anwendung und sieht in etwa wie folgt aus:

```
<meta http-equiv="refresh" content="5; http://www.foo.de/bar">
```

Beispiele:

Wenn Inhalte eines Webauftrittes verschoben werden, kann es passieren, dass auf sie zeigende Hyperlinks ins Leere laufen. Der Server kann aber entsprechend konfiguriert werden, dass automatisch serverseitig auf den neuen URL der Datei weitergeleitet wird. In der Fachsprache der Serveradministratoren ist dies ein 301 moved permanentely. Dies hat zudem den Vorteil, dass hierdurch der Datenverkehr zwischen Browser und Server reduziert wird und Ihre Seiten insgesamt performanter werden.

Bei den vergangenen Relaunches der ›Einfach für Alle‹–Seiten wurden viele Inhalte überarbeitet und auch immer wieder die Struktur der Website verändert. Wie immer bei einem solchen Umbau stellte sich das Problem, dass Nutzer mit alten Bookmarks oder Links, die von außerhalb auf das Webangebot zeigen, ins Leere laufen und einen der berüchtigten Fehler 404 zurückbekommen. Daher wurde die .htaccess-Datei des Servers so eingerichtet, dass sie diese Anfragen automatisch an den neuen Speicherplatz (URL) der Dokumente weitergibt, ohne dass der Nutzer etwas davon mitbekommt. Gleichzeitig weden damit Suchmaschinen über die neue Adresse der Ressource informiert.

Anforderung 8: Zugänglichkeit von Benutzerschnittstellen

»Die direkte Zugänglichkeit der in Internetangeboten eingebetteten Benutzerschnittstellen ist sicherzustellen.« <zitat>

Was heißt das?

In dieser Anforderung dreht sich alles um die Zugänglichkeit von Inhalten, die Informationen zur Darstellung und Bedienung nicht über die Standard-Schnittstellen des Browsers, sondern über eigene Mechanismen an Ausgabegeräte wie assistive Programme liefern. Dies sind üblicherweise programmierte Objekte, die über das reine HTML oder Bilder hinausgehen und mit den Elementen `OBJECT`, `APPLET` (veraltet), `EMBED` (noch nicht standardisiert), aber auch `SCRIPT` realisiert sind. Dabei ist zwischen verschiedenen Arten von Objekten zu unterscheiden;

- Objekte, die zwar in eine Seite eingebettet sind, aber wie bei einer einfachen, mit Flash, SVG oder ähnlichem realisierten Infografik, keine Interaktion seitens des Benutzers benötigen. Hierfür gelten ähnliche Bedingungen wie bei eingebetteten statischen Formaten wie z.B. Bildern.
- Objekte, die mit Programmlogik versehen sind und eine Interaktion seitens des Benutzers benötigen. Hierfür ist die Bedingung 8.1 und ergänzend die Bedingungen der Anforderung 9 anzuwenden.

Alternativen für den Fall, dass diese vom Anwender nicht in der allgemein üblichen Weise genutzt werden können, wurden bereits in Anforderung 1 und Anforderung 6 besprochen. Während diese Bedingungen jedoch zum Teil noch von veralteten Hilfsmitteln ausgehen, die mit alternativen Inhalten und Funktionen zu versorgen sind, geht es hier um den moderneren Ansatz, die direkte Zugänglichkeit zu ein und demselben Angebot sicherzustellen.

Was meint die BIENE?

Eine der wesentlichen Neuerungen im Prüfverfahren zur BIENE 2006 war, dass nun mehr Wert auf die direkte Zugänglichkeit solcher Objekte gelegt wird. Im Sinne des integrativen Ansatzes der Prinzipien des Barrierefreien Webdesigns soll damit die gleiche Nutzbarkeit von Webinhalten durch alle Nutzer geprüft werden, was insbesondere Menschen mit Behinderungen einschliesst.

Andere Testverfahren machen sich bisher nicht die Mühe, die direkte Zugänglichkeit von programmierten Objekten zu testen. Diese bewerten lediglich, ob es alternative Angebote gibt, mit denen solche abzulehnenden Sonderwege nur fortgeschrieben werden. Dies führt dazu, dass Anbieter von Web-basierten Anwendungen keinerlei Hilfestellungen erhalten, um ihre Applikationen zugänglich zu gestalten, da diese von manchen als grundsätzlich fehlerhaft im Sinne der Barrierefreiheit angesehen werden. Hier ist es aus unserer Sicht in naher Zukunft unerlässlich, dass sich diese beiden bisher strikt getrennten Welten aufeinander zubewegen, und zwar aus beiden Richtungen.

Bedingung 8.1: Zugängliche Alternativen für programmierte Objekte (Prio. 1)

»Programmierte Elemente (insbesondere Scripts und Applets) sind so zu gestalten, dass sie entweder direkt zugänglich oder kompatibel mit assistiven Technologien sind.« <zitat>

Was heißt das?

In Anforderung 1 wurden ja bereits die alternativen Inhalte zu eingebetteten statischen Objekten diskutiert. Diese Anforderung kümmert sich nun um die Zugänglichkeit von nicht-statischen Inhalten, die üblicherweise per `OBJECT` oder `EMBED` in eine Seite eingebunden werden und ein zusätzliches Plug-In benötigen, um im Browser ausgeführt zu werden.

Nach dieser Anforderung müssen **sämtliche** Inhalte eines Web-Angebotes genauso zugänglich sein wie der Rest der Seite, gleichgültig welche Technologie eingesetzt wird. Folgerichtig gelten also für Flash- und Java-Anwendungen, aber auch AJAX- und JavaScript-basierte Web-Anwendungen die gleichen Kriterien für die Ausgabe und Bedienung mit assistiven Hilfsmitteln. Eingebettete oder verlinkte PDF-Dokumente können ebenfalls »Programmierte Elemente« sein, da auch in diesen zum Beispiel in Formularen Berechnungen angestellt werden oder andere Skripte ausgeführt werden können.

Nicht ganz so einfach ist die Abgrenzung bei multimedialen Inhalten, angefangen von reinen Flash-Videos über komplexere Präsentationen bis hin zu Angeboten mit echtem Anwendungscharakter. Auch ein einfaches Video bietet bereits eigene »eingebettete Benutzerschnittstellen« im Sinne dieser Anforderung und kann im Ernstfall auch »programmierte Elemente« enthalten. Somit fällt dieses gegebenenfalls auch unter die Bedingung 8.1 und ist entsprechend zu testen.

Wie können Sie das testen?

Im Prinzip handelt es sich um eine sehr allgemein gehaltene Bedingung, die lediglich den Rahmen für die Zugänglichkeit bestimmter Web-Inhalte absteckt. Was im Detail zu beachten und zu testen ist, wird in einer Vielzahl anderer Anforderungen und Bedingungen der BITV festgehalten. Für eingebettete Objekte und programmierte Elemente gelten die bekannten

Meßkriterien zu (Text-)Alternativen, Skalierbarkeit, Farben und geräteunabhängiger Bedienung, zu Bewegungen und automatischen Aktualisierungen sowie zu Flackern bzw. Blinken und vieles mehr.

Die hier ausdrücklich erwähnten assistiven Werkzeuge, die z. B. von blinden oder sehbehinderten Menschen, motorisch behinderten Menschen oder Menschen mit Lernbehinderung benutzt werden, sind sehr vielfältig. Daher können Sie *ohne* Tests mit echten Benutzern *nicht* davon ausgehen, dass Ihr Web-Angebot mit diesen Hilfsmitteln auch tatsächlich bedienbar ist.

Aber auch bereits in der Entwicklung von Web-basierten Anwendungen können Sie eine ganze Reihe Tests durchführen um die Erfüllung dieser Bedingung zu überprüfen, ohne sich gleich in die zugegebenermaßen sehr komplexe Bedienung eines Screenreaders einarbeiten zu müssen.

In begrenztem Maße geht dies bereits mit einem sogenannten DOM-Inspector, der für Browser wie Firefox, Internet Explorer, Safari oder Opera nachrüstbar ist. Besonders wenn Sie Client-seitige Skripte zur Manipulation der Inhalte oder Funktionen einsetzen geben diese Werkzeuge einen guten Überblick, in welchem Zustand sich ihr Angebot nach einer Interaktion durch den Benutzer befindet, welche geänderten Elemente verfügbar sind, welche Attribute diese besitzen usw.

Da die weitaus meisten assistiven Werkzeuge nach wie vor unter Windows laufen und dort auf den Internet Explorer aufsetzen beschränken wir uns hier auf Testwerkzeuge, mit denen Sie die Zugänglichkeit Ihrer programmierten Objekte auf dieser Plattform testen können. Diese Hilfsmittel werden vom Betriebssystem über die sogenannte *Microsoft Active Accessibility*-Schnittstelle (MSAA) mit Informationen über die vorgefundenen Inhalte, ihre Rollen und mögliche Zustände versorgt. (Andere Betriebssysteme bieten ähnliche Schnittstellen, auf die wir hier nicht eingehen. Entsprechende Dokumentationen finden Sie bei Interesse auf den Seiten von Apple, Sun und GNOME)

Inspect32 in der Anwendung

Ein Bestandteil der frei erhältlichen Active Accessibility Software Development Kit Tools von Microsoft ist das Werkzeug inspect32.exe. Mit dieser Anwendug lässt sich beobachten, welche zusätzlichen Accessibility-Informationen einer Anwendung über die MSAA-Schnittstelle nach ›aussen‹ durchgereicht werden.

Ein weiteres Werkzeug ist das mittlerweile als Open Source veröffentlichte MSAAVerify, mit dem Sie testen können, ob die Properties und Methoden ihrer Schnittstellen wie zum Beispiel Taststurkürzel den Richtlinien der MSAA entsprechen.

Anforderung 9: Unabhängigkeit der Funktionen von Ein- und Ausgabegeräten

»Internetangebote sind so zu gestalten, dass Funktionen unabhängig vom Eingabegerät oder Ausgabegerät nutzbar sind.«

Was heißt das?

Wenn Ihr Angebot aus sauberem HTML 4 oder XHTML besteht, ist es schon relativ unabhängig vom Ausgabegerät. HTML 4 verzichtet, insbesondere in der Variante *strict*, auf viele Elemente, die in den Vorgängerversionen ausschließlich zur grafischen Präsentation gedacht waren. In dieser Anforderung kommt eine weitere Dimension hinzu: ein Angebot soll nicht nur in einer Vielzahl von Ausgabemedien *wahrnehmbar* sein, sondern auch *bedienbar*.

In der BITV ist dies eine sehr allgemeine Formulierung, die als Fallback für alle nicht explizit erwähnten Eventualitäten dient. Deutlicher wird der Sinn dieser Anforderung durch die Lektüre der ursprünglichen Richtlinie 9 der WCAG: »Verwenden Sie Features, die die Aktivierung von Seitenobjekten über eine Reihe von Eingabegeräten ermöglichen.«

Die BITV geht hier über das Original hinaus und erwähnt zusätzlich zu den Eingabegeräten auch noch Ausgabegeräte, da beide in bestimmten Formen der Nutzung zwangsläufig miteinander verbunden sind.

Alternative Arten der Ein- und Ausgabe, insbesondere assistive Werkzeuge wie Sprachein- und -ausgaben, Braillezeilen oder die verschiedenen Hilfsmittel von Menschen mit motorischer Behinderung stellen andere Anforderungen an ein Web-Angebot als die klassische Kombination aus Maus und Monitor. Da nicht alle Benutzeragenten sämtliche Formen der Ausgabe und Bedienung beherrschen ist diese Anforderung nur dann testbar, wenn man sich intensiver mit den möglichen Hilfsmitteln beschäftigt und sein Angebot tatsächlich in diesen testet bzw. testen lässt.

Ein häufiger Fehler ist die Abfrage von Browserversionen, ohne die tatsächlichen Fähigkeiten der Browser zu testen. Die unzulässige Verallgemeinerung, dass in einem bekannten Browser alles genutzt werden kann führt im Umkehrschluß oft dazu, dass unbekannte Browser ausgesperrt werden, da sie von den entsprechenden Skripten nicht erkannt werden. Mit Ausnahme einiger Screenreader lässt sich kaum prüfen, ob Ihre Seiten auf einem Monitor dargestellt werden oder über Lautsprecher und Braillezeile ausgegeben werden, oder ob der Nutzer die Seiten mit der Maus oder der Tastatur bedient.

Bedingung 9.1: Verzicht auf serverseitige Imagemaps (Prio. 1)

»Es sind clientseitige Imagemaps bereitzustellen, es sei denn, die Regionen können mit den verfügbaren geometrischen Formen nicht definiert werden.«

Was heißt das?

Wie bei den bereits zuvor in Bedingung 1.2 und Bedingung 1.5 genannten clientseitigen Imagemaps geht es um die Verarbeitung einer Interaktion mit pixelgenauen Koordinaten bei `IMG`-Elementen, aber auch bei anderen, z. B. per `OBJECT` eingebetteten Formaten wie Java oder Flash.

Der entscheidende Unterschied zwischen diesen unterschiedlichen Techniken liegt aus Nutzersicht in der Bedienbarkeit per Tastatur und der Vorhersagbarkeit des Ergebnisses einer Interaktion. Die Informationen über die Links und deren Ziel liegen bei einer *client*seitigen Imagemap bereits lokal im Browser des Nutzers vor (`AREA`); bei einer *server*seitigen Imagemap muss hierzu erst die Interaktion (d.h. ein Klick auf bestimmte Koordinaten) durch den Server ausgewertet werden, dieser schickt dann das Ergebnis (z.B. eine neue Seite oder eine geänderte Imagemap) an den Browser zurück.

Im Gegensatz zu den serverseitigen Varianten bieten clientseitige Imagemaps aus Sicht der Barrierefreiheit den Vorteil, per Tastatur bedienbar zu sein, da die aktivierbaren Regionen einer Grafik oder eines Bildes echte, im HTML-Dokument enthaltene Hyperlinks sind.

Bei einer serverseitigen Imagemap sind hingegen prinzipiell alle Pixel der jeweiligen Grafikdatei klickbar, ob diese tatsächlich mit einer Funktion belegt sind oder ob sich hinter einem Teil der Imagemap gar keine weiterführenden Links oder Funktionen befinden kann nur der Server entscheiden. Clientseitige Imagemaps sind also zu bevorzugen.

Aaaaber…
Nicht alle möglichen klickbaren Regionen einer Imagemap lassen sich mit clientseitigen Techniken abbilden, daher die Einschränkung »es sei denn die Regionen können mit den verfügbaren geometrischen Formen nicht definiert werden«. Dabei geht es nicht um die technische Machbarkeit der Definition geometrischer Formen. Bei der Erstellung einer Imagemap lassen

sich strenggenommen alle denkbaren Regionen mit den zur Verfügung stehenden Attributen Rechteck, Kreis und Polygon definieren.

Die Bedingung zielt vielmehr auf den Einsatzzweck der Imagemap ab: in Stadtplänen oder Landkarten (wie Yahoo! Maps oder Google Maps) werden generell Imagemaps mit serverseitiger Logik verwendet, da sich die Informations- und Funktionsdichte einer solchen Karte unmöglich in Form einer clientseitigen Imagemap abbilden lässt. In diesem Fall ist die Einschränkung »es sei denn …« durchaus zutreffend und entsprechend anwendbar.

Wie können Sie das testen?

Statische serverseitige Imagemaps lassen sich, sofern es sich um ein HTML-basiertes Objekt handelt, recht gut von clientseitigen Imagemaps unterscheiden. Bei letzteren sind die Koordinaten in der jeweiligen HTML-Seite hinterlegt, intern wird über das `usemap`-Attribut auf diese Koordinaten verweisen. Bei serverseitigen Imagemaps fehlen die Koordinaten, stattdessen wird das `ismap`-Attribut gesetzt und auf eine Map-Datei auf dem Server verwiesen.

Werkzeuge wie die Web Developer Toolbar und ähnliche helfen hier nur begrenzt, da die Auswertung und Übergabe der Koordinaten ebenso per Skript geschehen kann. Ohne die genaue Analyse des Quelltextes lässt sich nicht abgrenzen, ob es sich um eine clientseitige oder um eine serverseitige Imagemap handelt, oder ob sogar eine Mischform vorliegt.

Bedingung 9.2: Auch ohne Maus nutzbar (Prio. 1)

<zitat>

»Jedes über eine eigene Schnittstelle verfügende Element muss in geräteunabhängiger Weise bedient werden können.«

Was heißt das?

Inhaltlich ist diese Bedingung mit einigen anderen Bedingungen wie 6.4, 8.1 und 9.3 verwandt. In diesen Bedingungen geht es jedoch weitestgehend um die klassische Combo aus HTML, CSS und JavaScript, deren Ausgabe und Bedienung über die Standard-Schnittstellen läuft, die von den jeweiligen Browsern zur Verfügung gestellt werden.

Hier geht es nun um die spezifischen Eigenheiten von Inhalten, die über *eigene* Schnittstellen mit dem eingesetzten Benutzeragenten und eventuell vorhandenen Hilfsmitteln kommunizieren. Üblicherweise handelt es sich um eingebettete Inhalte und programmierte Objekte. Diese werden nicht nativ durch den Browser dargestellt oder ausgeführt, sondern laufen innerhalb eines Zusatzmoduls (sog. *Plug-In* wie zum Beispiel Acrobat oder Flash), das dem Browser die Verarbeitung dieses Formats überhaupt erst ermöglicht. Dabei ist unerheblich, ob diese Inhalte innerhalb des ursprünglichen Browserfensters stehen, oder in einer anderen Anwendung geöffnet werden, da dieses Verhalten von den Einstellungen des Nutzers und seiner Software-Ausstattung abhängt.

Diese Formate werden nicht im HTML-Editor, sondern mit zusätzlichen Programmen oder Entwicklungsumgebungen erstellt. Daher verweisen die der BITV zu Grunde liegenden Web Content Accessibility Guidelines (WCAG) an dieser Stelle zu Recht auf eine weitere Empfehlung des W3C, die Zugänglichkeitsrichtlinien für Autorenwerkzeuge (Authoring Tool Accessibility Guidelines, ATAG. In den ATAG-Richtlinien ist festgelegt, dass diese Programme den Anwender bei der Erstellung barrierefreier Inhalte unterstützen müssen. Hierzu gehört natürlich auch die Sicherstellung der Bedienbarkeit mit alternativen Eingabemethoden.

Hier liegt ein Teil der Verantwortung beim Nutzer und bei den Herstellern der Browser und Hilfsmittel. Die Bereitstellung von zugänglichen Schnittstellen durch den Anbieter nützt nichts, wenn diese durch die verwendeten Hilfsmittel nur unzureichend unterstützt werden, oder wenn ein Nutzer Hilfsmittel oder Browser verwendet, die für seine spezifische Behinderung ungeeignet sind.

Beispiele:

Die Bedingung 9.2 bedeutet auch, dass Features von Webinhalten, die sich aus anderen Bedingungen ergeben, ebenso in in geräteunabhängiger Weise bedienbar sein müssen. Dazu gehört zum Beispiel, dass man ein eingebettes Video ebenso per Tastatur pausieren, stoppen, und vor- bzw. zurückspulen kann, wie dies per Maus möglich ist.

Zur geräteunabhängigen Nutzbarkeit gehört auch, dass der Anwender stets über den aktuellen Fokus informiert wird. Im CSS sollte hierfür zusätzlich zum `:hover`-Selektor die `:focus`-Pseudoklasse gesetzt werden, um den gleichen visuellen Effekt auch für Tastaturnutzer zu erreichen. Wie Sie dieses Verhalten für die in dieser Bedingung besprochenen Formate erreichen, die über HTML & CSS hinausgehen ist vom jeweils verwendeten Autorenwerkzeug abhängig; eine detaillierte Besprechung sprengt den Rahmen dieser Artikelserie.

Wie können Sie das testen?

Um diese Bedingung testen zu können muss Ihr Browser natürlich mit den entsprechenden Plug-Ins ausgestattet sein und diese müssen aktiviert sein. Abgesehen von dieser Einschränkung läuft der Test an dieser Stelle wie bei allen anderen

Bedingungen zur Geräteunabhängigkeit: ziehen Sie den Mausstecker und versuchen Sie, sämtliche Funktionen Ihres Angebots mit der Tastatur zu bedienen. Zu beachten ist dabei lediglich, dass die Tasten zum Auslösen von Aktionen in den verschiedenen Browser unterschiedlich belegt sind, im Zweifelsfall sollten Sie das jeweilige Handbuch konsultieren.

Bedingung 9.3: Geräteunabhängige Eventhandler (Prio. 1)

»In Scripts sind logische anstelle von geräteabhängigen Event Handlern zu spezifizieren.« <zitat>

Was heißt das?

Event-Handler sind sozusagen das Bindeglied zwischen HTML und JavaScript: hiermit können Sie Ereignisse angeben, die von einem Skript zur weiteren Verarbeitung ›abgefangen‹ werden können. Über den Event-Handler wird die Art des Ereignisses mitgeteilt, das der Nutzer durch seine Interaktion mit der Seite ausgelöst hat. Dies kann ein einfacher Klick oder ein Doppelklick sein, eine Tastatureingabe oder bestimmte Bewegungen mit der Maus. Auf Basis dieser verschiedenen Interaktionen können dann per Skript zum Beispiel die Inhalte einer Seite modifiziert oder weitergehende Aktionen wie die Überprüfung von Formulareingaben ausgelöst werden.

HTML und JavaScript kennen eine ganze Reihe solcher Event-Handler, im einzelnen sind dies: `onblur`, `onchange`, `onclick`, `ondblclick`, `onfocus`, `onkeydown`, `onkeypress`, `onkeyup`, `onload`, `onmousedown`, `onmousemove`, `onmouseout`, `onmouseover`, `onmouseup`, `onreset`, `onselect`, `onsubmit`, `onunload`. Des weiteren gibt es noch Browser-spezifische Events wie `onabort` und `onerror`, die aber nicht Teil der HTML-Spezifikation sind und nicht von allen Browsern verstanden werden.

In HTML gibt es nicht für alle geräteabhängigen Event-Handler einen entsprechenden logischen Event-Handler. So nützt es bei `ondoubleclick` leider nichts, statt dessen zweimal die Aktion `onkeypress` ausführen zu lassen.

Tipp: ignorieren Sie bei diesem Punkt die Dokumentation der entsprechenden Richtlinie in den WCAG 1.0 bzw. in den Techniken zur Umsetzung – diese ist ausgesprochen lücken- und fehlerhaft. Die Techniken stammen noch aus Zeiten, als Scripting von Teilen des W3C als generell schädlich angesehen wurde. Zudem lag noch keine ausreichende Erfahrung im Umgang von assistiven Werkzeugen, insbesondere Screenreadern mit modernen Formen des Scripting vor. Leider sind diese Dokumente seit der Verabschiedung im Jahr 1999 nicht mehr aktualisiert worden, auch der aktuelle Stand des Nachfolgers WCAG 2.0 hat in diesem Bereich noch Luft nach oben.

Probleme bei der Bewertung

Die meisten in der Literatur zu findenden Ansätze, vermeintlich geräteabhängige Event-Handler wie Maus-Ereignisse mit einem (ebenso vermeintlich) geräteunabhängigen Event-Handler zu paaren sind, gelinde gesagt, mit Vorsicht zu genießen. Dies liegt zum Teil daran, dass die Event-Handler zum Teil mißverständlich benannt sind. Entgegen landläufiger Meinung lassen sich *keine* Rückschlüsse auf ihre Geräteunabhängigkeit alleine auf Basis des Vorkommens von Wörtern wie *Mouse* oder *Click* machen. ›`Onclick`‹ müsste strenggenommen ›`onactivation`‹ heißen, da dieser Event auch durch Tastatureingaben ausgelöst wird.

Auch der umgekehrte Fall tritt auf: Events wie `onfocus` und `onblur` sind nach der HTML-Spezifikation geräte**un**abhängige Ereignisse. In den üblichen Browsern sind diese aber als Events implementiert, die ausschließlich über die Tastatur (und in diesem Fall *nicht* durch die Maus) ausgelöst werden.

Negatives Beispiel:

Ein häufiger Fehler ist, Formularelemente per `onchange` oder `onblur` zu prüfen. Diese Art der Überprüfung ist zwar direkt und schnell und damit scheinbar hilfreich, kann aber das Ausfüllen eines Formulars per Tastatur oder Spracherkennung zu einer Tortur machen. Die wohl am meisten verbreitete Anwendung dieser Technik sind Dropdown-Menüs:

HTML:

```
<form>
 <label for="go2">Gehe zu:</label>
  <select name="go2" id="go2" onchange="send(this)">
   <option value="anforderung-7.php">Anforderung 7</option>
   <option value="anforderung-8.php">Anforderung 8</option>
   <option value="anforderung-9.php">Anforderung 9</option>
  </select>
</form>
```

JavaScript:

```
function send(f)
{
var chosen;
chosen=f.options[f.selectedIndex].value;
self.location=chosen;
}
```

Dieses nicht-barrierefreie Beispiel zum ausprobieren

Das Hauptproblem hierbei ist, dass Tastaturnutzer versuchen werden, mit Tabulator- und Pfeiltasten eine der Optionen auszuwählen. Sie werden jedoch niemals weiter als die erste Möglichkeit kommen, da die Funktion `send()` durch das Event `onchange` ausgeführt wird, sobald sich etwas ändert.

Erfahrene Tastaturbenutzer wissen, dass man zuerst per Alt + ↓ die Liste anzeigen muss, um dann die Option mittels der Pfeiltasten und Eingabe auszuwählen. Allerdings ist diese Bedien-Technik nicht jedem Besucher bekannt.

Zudem passiert im obigen Beispiel nichts, wenn JavaScript abgeschaltet ist, da das Formular für diese Fälle keinen Absenden-Button als Fallback vorsieht.

Positives Beispiel:

Das folgende Beispiel funktioniert immer und erspart einen Besuch beim Server, wenn JavaScript vorhanden ist. Sollte JavaScript abgeschaltet sein, übernimmt das serverseitige change.php-Skript die weitere Verarbeitung.

HTML:

```
<form action="change.php" method="post" ↵
onsubmit="return send(this)">
 <label for="go2">Gehe zu:</label>
  <select name="go2" id="go2">
   <option value="anforderung-7.php">Anforderung 7</option>
   <option value="anforderung-8.php">Anforderung 8</option>
   <option value="anforderung-9.php">Anforderung 9</option>
  </select>
 <input type="submit" value="OK" />
</form>
```

JavaScript:

```
function send(f)
{
var chosen;
chosen=f.go2.options[f.go2.selectedIndex].value;
self.location=chosen;
return false;
}
```

PHP:

```
<?PHP if(isset($_POST['go2'])){
header('Location:'.$_POST['go2']);
}?>
```

Dieses barrierefreie Beispiel zum ausprobieren

Und was ist mit `onkeypress`?

In den Techniken zu WCAG 1.0 findet sich die Vorgabe, Aktionen ausschließlich über solche Befehle aufzurufen, die unabhängig vom Eingabegerät sind:

<zitat>

»[…] if you must use device-dependent attributes, provide redundant input mechanisms (i.e., specify two handlers for the same element):

- Use ›onmousedown‹ with ›onkeydown‹.
- Use ›onmouseup‹ with ›onkeyup‹
- Use ›onclick‹ with ›onkeypress‹

Das klingt zunächst plausibel, wirft allerdings in der realen Welt Probleme auf. Der Befehl `onkeypress` ist in einigen Browsern sehr schlecht implementiert. Mozilla und Firefox zum Beispiel starten die Aktion, sobald man die Tabulator-Taste betätigt – was eigentlich den Mausklick nicht ersetzen, sondern zum nächsten interaktiven Element gehen sollte.

Websurfer, die auf Tastaturen angewiesen sind, haben häufig eine bestimmte Taste als Ihre ›Klicktaste‹ definiert, meistens die Leertaste oder Eingabe, um einen `onclick`-Befehl auszuführen. Wenn wir zusätzlich zu diesem noch einen `onkeypress`-Befehl verwenden, kann es passieren, dass das notwendige Tastaturkürzel blockiert oder die Funktion mehrmals aufgerufen wird. Beispiele für Tastaturfunktionen sind Mozillas Such- und Tippfunktion, Operas Tastaturkürzel (wie zum Beispiel ›A‹ für den nächsten Verweis) oder die Tastaturkürzel von Screenreadern wie JAWS.

Es kann also passieren, dass eine Befolgung der Richtlinien bis ins letzte Detail Besucher aussperrt oder es Ihnen unnötig schwierig macht, eine Seite oder Anwendung zu benutzen. Falls Sie befürchten müssen, dass Ihre Seiten einem automatisierten Accessibilitytest unterzogen werden, dann können Sie für die Dauer des Tests ein zusätzliches `onkeypress` einbauen und hinterher wieder auskommentieren.

Unser Dank geht an Christian Heilmann, der uns die Beispiele freundlicherweise überlassen hat. Das vollständige Tutorial finden sie auf der Website »Barrierefreies JavaScript«.

Sonderfall Screenreader

Einen speziellen Anwendungsfall für barrierefreies JavaScript stellen die verbreiteten Screenreader wie JAWS, Virgo oder Window Eyes dar. Einfache Events wie `onmouseover` oder Klick-Events bei Links oder Formularelementen werden von diesen mittlerweile unterstützt. Das Problem hierbei ist, dass gängige Screenreader gut mit waghalsigen Code-Konstrukten aus dem vergangenen Jahrhundert zurechtkommen (s. diese Beispielseite), bei moderneren (und damit WCAG- bzw. BITV-konformen) Methoden der Umsetzung solcher Funktionen regelmäßig scheitern, wie Untersuchungen von Accessibility-Experten gezeigt haben.

Ein möglicher Lösungsweg aus diesem Dilemma könnte sein, den eigentlich unerwünschten alternativen Versionen für Screenreader-Nutzer zu neuem Leben verhelfen. In Web-basierten Anwendungen kann es durchaus sinnvoll sein, eine statische Alternative anzubieten. Dem Nutzer wird dadurch die Wahl gelassen zwischen einer statischen, jedoch weniger komfortablem Version und einer dynamischen, mit JavaScript angereicherten Version. Diese statischen Alternativen sollten jedoch nur als Interims-Lösung betrachtet werden, um den Herstellern von Hilfsmitteln die Chance zu geben, ebenfalls standardkonforme Anwendungen zu entwickeln.

Bedingung 9.4: Per Tabulator schlüssig navigierbar (Prio. 2)

»Es ist eine mit der Tabulatortaste navigierbare, nachvollziehbare und schlüssige Reihenfolge von Hyperlinks, Formularkontrollelementen und Objekten festzulegen.« <zitat>

Was heißt das?

Eine wichtige Bedingung, der immer noch zu wenig Bedeutung zugemessen wird. Bereits im Vortest des BIENE-Prüfverfahrens scheitern eine große Anzahl an Einreichungen, weil sich die Angebote nicht mit der Tastatur bedienen lassen, sondern zur Nutzung zwingend eine Maus voraussetzen. Dabei ist gerade die Bedienung per Tastatur oder vergleichbaren Hilfsmitteln für viele Menschen mit unterschiedlichsten Behinderungen von enormer Bedeutung. Menschen mit motorischen Behinderungen, blinde und stark sehbehinderte Nutzer sind gleichermassen auf die Erfüllung dieser Bedingung angewiesen und werden ohne eine sinnvolle Abfolge der Inhalte an einer Nutzung gehindert.

Eine »navigierbare, nachvollziehbare und schlüssige Reihenfolge« ergibt sich beinahe von selbst, wenn der Quelltext einer Seite oder Anwendung entsprechend strukturiert ist. Generell arbeiten assistive Werkzeuge die Inhalte in der Reihenfolge ab, in der sie im Quelltext stehen.

Wenn diese Reihenfolge z. B. in einem komplexen Formular nicht mehr sinnvoll für die Bewältigung der Aufgabe erscheint (*und **nur** dann*), sollten Sie die Reihenfolge durch das `tabindex`-Attribut korrigieren. Hiermit wird die Reihenfolge festgelegt, in der sich ein Nutzer mit der Tabulatortaste durch eine Seite bewegt. Die Werte dieses Attributs müssen nicht fortlaufend durchnummeriert sein. Sie können in einem Dokument zum Beispiel mit `1`, `2`, `3` …anfangen und dann mit `10`, `11`, `12`, … weitermachen. So bleibt Ihnen Raum für eventuelle Erweiterungen, bei denen Sie nicht immer wieder alles neu nummerieren müssen.

Das heißt nicht, dass Sie alle erdenklichen Elemente einer Seite mit einem solchen `tabindex` ausstatten sollten; wesentlich sinnvoller ist eine brauchbare Strukturierung. Erlaubt sind übrigens Werte zwischen 0 und 32767, die Sie aber hoffentlich nie benötigen werden. Auch negative Werte (`tabindex="-1"`) sind möglich. Hierdurch wird ein Element per Skript oder Mausklick fokussierbar, dieses Element ist dann aber *nicht* Bestandteil der Tabreihenfolge.

Ein übermäßiger oder unbedachter Einsatz von `tabindex` kann neue Barrieren aufbauen. Mehr dazu in unserem Artikel: »Widersprüche in

Im HTML-Standard ist das `tabindex`-Attribut für die Elemente `A`, `AREA`, `BUTTON`, `INPUT`, `OBJECT`, `SELECT` und `TEXTAREA` vorgesehen. In Internet Explorer und Firefox ist es darüberhinaus möglich, *sämtlichen* Elementen einer Seite ein `tabindex`-Attribut zuzuweisen. Mit einer entsprechenden Erweiterung des HTML-Standards ist zu

der Barrierefreiheit: Tastaturbedienung und Tabindex«.

rechnen, weitere Informationen zu diesem Thema finden Sie in der »Dynamic Accessible Web Content Roadmap« des W3C bzw. unter »Accessible DHTML« im Mozilla Developer Center.

In rein inhaltsorientierten Webauftritten ist diese Erweiterung nicht nötig – hier sollten Sie von vorne herein für einen logischen Aufbau der Dokumente sorgen, sodaß der Einsatz von tabindex unnötig ist. In Web-basierten Anwendungen lässt sich dieser logische Ablauf im Quelltext jedoch nicht so ohne weiteres herstellen, zumal die Reihenfolge der Inhalte oft aufgrund von Interaktionen des Nutzers verändert wird. Dazu kann es nötig sein, mehr Elemente als Links und Formularelemente fokussierbar zu machen, sodaß man hier auf den Einsatz des Attributs an Stellen angewiesen ist, die eine Validierung des HTML unmöglich machen.

Attribut / Wert	**Tab-Reihenfolge**	**Mit der Maus oder per `element.focus()` fokussierbar**
Kein `tabindex` definiert	Normales Verhalten des Elements	Nur Formularelemente und Links sind fokussierbar
`tabindex="0"`	In der Tab-Reihenfolge entsprechend seiner Position im Dokument	Ja
`tabindex="n"` (Wert `n` = ganze Zahl zwischen 1 und 32768)	Der Wert bestimmt die Position in der Tabreihenfolge	Ja
negativer Wert `tabindex="-1"`	Nicht in der Tab-Reihenfolge, aber per JavaScript mit der `element.focus()`-Methode auf Basis eines Tastendrucks (z.B. Pfeiltasten) fokussierbar.	Ja

Wie können Sie das testen?

Endlich mal ein Test, bei dem Sie nicht mehr, sondern weniger Testwerkzeuge brauchen: ziehen Sie den Stecker Ihrer Maus ab und benutzen Sie nur die Tabulator- und Eingabetasten zur Navigation Ihrer Website. Wenn Sie alle Inhalte in einer Reihenfolge erreichen, die sinnvoll und nachvollziehbar erscheint, haben Sie diesen Test bestanden.

Wie viele andere Bedingungen gilt diese nicht nur für reine HTML-Dokumente, sondern sie ist auch auf Formate wie PDF und insbesondere Flash anwendbar. Wenn Sie Flash verwenden sollten Sie hier ebenfalls per Tastatur testen, ob alle relevanten Objekte wie Schaltflächen, Movieclips oder Textfelder in einer sinnbildenden Reihenfolge erreichbar sind.

Was meint die BIENE?

Im BIENE-Prüfverfahren fallen mit schöner Regelmässigkeit eine hohe Anzahl von Einreichungen bereits im Vortest durch, die auf bestimmten Redaktionssystemen basieren. So hinterlegt ein bekanntes Open Source Content Management System Navigationslinks mit dem JavaScript-Schnipsel `<a href="index.php?id=1" onfocus="blurLink(this);">Link</a>`, um eine vermeintlich hässliche Hervorhebung von Links zu verhindern. Damit sind die Seiten praktisch nicht mit der Tastatur bedienbar, da beim Antabben des Links der Fokus sofort wieder per Skript entfernt wird.

Bedingung 9.5: Accesskeys (Prio. 2)

<zitat>

»Es sind Tastaturkurzbefehle für Hyperlinks, die für das Verständnis des Angebots von entscheidender Bedeutung sind (einschließlich solcher in clientseitigen Imagemaps), Formularkontrollelemente und Gruppen von Formularkontrollelementen bereitzustellen.«

Was heißt das?

Die hier geforderten Tatstaturkurzbefehle werden in HTML mit dem `accesskey`-Attribut realisiert. Sie funktionieren wie die Taststur-Shortcuts zu den Menübefehlen in den unterschiedlichen Betriebssystemen: Alt + D öffnet unter Windows üblicherweise das Menü ›Datei‹, Alt + B steht für Bearbeiten und so weiter.

Soweit die Theorie: leider sind so gut wie alle Zeichen in den gängigen Browsern schon mit irgendwelchen Menübefehlen belegt, so dass fast nur noch die Ziffern des 10er–Blocks für die Inhalte und Funktionen einer Webseite übrig bleiben. Selbst die Zahlen werden von manchen assistiven Tools für Sonderfunktionen verwendet, so wie Alt + 1 vom IBM Home Page Reader für dessen Überschriftenfunktion.

Bei komplexen Formularen mit vielen Kontrollelementen ist es unmöglich, für alle Elemente einen passenden Accesskey zu finden, der nicht in irgendeiner Anwendung mit anderen Funktionen belegt ist. Daher bleibt nur die Empfehlung, sich maximal

auf die zehn zur Verfügung stehenden Zahlen zu beschränken und diese möglichst sinnvoll zu verteilen. Wie dies geschieht, ist dem Seitenbauer selbst überlassen, da es keinen einheitlichen Standard gibt. Sie sollten die Verwendung der Accesskeys auf jeden Fall z. B. auf einer Hilfe-Seite dokumentieren.

Das Fehlen von Regeln wirft jedoch weitere Probleme auf, die die Sinnhaftigkeit von `accesskey` weiter in Frage stellen. In Ermangelung eines de-fakto-Standards müssen Anwender die Tastaturbelegungen auf jeder Website neu erlernen – ein nicht zu unterschätzender Aufwand, der vom Anbieter gegen den zu erwartenden Nutzen abzuwägen ist, bevor man diese Attribute einsetzt. In Angeboten, die eher Anwendungscharakter haben (wie einem Online-Banking-Angebot oder einer Intranet-Anwendung) mag die Entscheidung für Accesskeys noch zu rechtfertigen sein, in reinen Content-Angeboten, wo der Nutzer nur mal gelegentlich ›vorbeischaut‹, ist dies eher nicht der Fall.

Screenreader geben die Informationen über vorhandene Accesskeys und deren Funktion erst dann aus, wenn das Element bereits fokussiert ist – dann braucht man den Accesskey aber nicht mehr, um das Element zu bedienen. Für den Anwender setzt der Nutzen erst bei wiederholten Besuchen ein, vorausgesetzt dieser ist willens und in der Lage, sich die Belegung zu merken.

Was meint die BIENE?

Im BIENE-Prüfverfahren wird dieser Punkt seit dem Jahr 2006 nur noch abgeschwächt geprüft:

<zitat>

»Accesskeys/ Shortcuts sollen nur dort eingesetzt werden, wo es für die Anwendung sinnvoll ist. Ihr Einsatz ist konsistent und transparent zu realisieren.«

Wie können Sie das testen?

Generell geben die heutigen grafischen Browser, aber auch viele Hilfsmittel einen hinterlegten Accesskey nicht sicht- oder hörbar aus, daher hilft hier nur ein kleiner Kniff, um den Blick in den Quelltext zu vermeiden. In den meisten grafischen Browsern lässt sich ein Benutzer-eigenes Style Sheet definieren, das eventuell vorhandene Accesskeys in die Dokumente schreibt. Fügen Sie die Zeile:

```
*[accesskey]:after {content: "[" attr(accesskey) "]";}
```

in Ihr Benutzer-eigenes Style Sheet ein und schon werden Ihnen die Accesskeys angezeigt. Dieser verwendete CSS–Trick nennt sich »*Generated Content*« und kann zu vielen verschiedenen Dingen benutzt werden, die das Surfen oder Testen erleichtern.

Einfacher geht es wie üblich mit den Webentwickler-Werkzeugen für Firefox und andere Browser. In der Web Developer Toolbar finden Sie im Menü ›Information‹ den Befehl ›Display Access Keys‹, der eventuell vorhandene Taststurkürzel neben den Elementen darstellt, denen sie zugewiesen sind.

Anforderung 10

»Die Verwendbarkeit von nicht mehr dem jeweils aktuellen Stand der Technik entsprechenden assistiven Technologien und Browsern ist sicherzustellen, so weit der hiermit verbundene Aufwand nicht unverhältnismäßig ist.«

Was heißt das?

Nicht alle Hilfen zur Barrierefreiheit, die man in seine Seiten einbauen kann, werden von allen Benutzerprogrammen unterstützt. Diese Anforderung regelt das Verhalten einer Website für genau den Fall: ist das gesamte Angebot immer noch mit älteren oder nicht Standard-konformen assistiven Werkzeugen verwendbar, wenn diese die eingesetzten neueren Techniken nicht beherrschen?

Die hier ausdrücklich erwähnten »assistiven Technologien« [sic!], die z. B. von Sehbehinderten oder blinden Menschen, Menschen mit motorischer Behinderung oder Lernbehinderung benutzt werden, sind sehr vielfältig, aber auf Grund der teilweise immensen Kosten nicht immer auf dem neuesten Stand der Technik. Daher können Sie *ohne* Tests mit tatsächlichen Benutzern solcher Werkzeuge nicht davon ausgehen, dass Ihre Seiten mit einer Vielzahl von Hilfsmitteln bedienbar sind.

Allerdings: sollte sich in der Entwicklung einer Website herausstellen, dass veraltete oder insbesondere nicht Standard-konforme Benutzerprogramme nur mit unverhältnismäßig hohen Mitteln oder gar nicht zu unterstützen sind, so können diese außen vor bleiben. Für diesen Zweck hat die Web Accessibility Initiative (WAI) des W3C die User Agent Aggessibility Guidelines (UAAG) veröffentlicht, ohne deren Berücksichtigung und Implementierung in die Benutzerprogramme ein wirklich barrierearmes Netz nicht möglich ist.

Ist diese Anforderung noch aktuell?

Die gesamten Bedingungen der BITV-Anforderung 10 sind im Original, den Web Content Accessibility Guidelines (WCAG 1.0) sogenannte »Until user agents…«-Guidelines. Das bedeutet, daß die ihnen zu Grunde liegenden Barrieren im historischen Kontext bewertet werden müssen: aus damaliger Sicht waren in der Tat viele der angesprochenen Probleme unüberwindliche Hürden für Nutzer und die damals gebräuchlichen Browser und assistiven Programme. *Aber sind sie es heute noch?*

Die Formulierung »nicht mehr dem jeweils aktuellen Stand der Technik entsprechenden assistiven Technologien« betrifft vorzugsweise Benutzerprogramme, die schon 1999 zur Entstehungszeit der Richtlinie als veraltet oder fehlerhaft gelten konnten. Auch die BITV spielt damit indirekt den Ball wieder an die Hersteller solcher Hilfsmittel zurück, die für eine Verbesserung Ihrer Produkte sorgen müssen.

Manchmal sind es auch die kleinen Dinge, über die Benutzer assistiver Programme oder auch ›ganz normale‹ Browser stolpern. So gibt es auch heute noch verbreitete Screenreader, die nicht das komplette Vokabular von HTML 4 verstehen, obwohl dieser Standard mittlerweile schon in die Jahre gekommen ist. Mehr dazu in den einzelnen Bedingungen:

Bedingung 10.1: (Prio. 1)

»Das Erscheinenlassen von Pop-Ups oder anderen Fenstern ist zu vermeiden. Die Nutzerin, der Nutzer ist über Wechsel der aktuellen Ansicht zu informieren.«

Was heißt das?

Das unerwünschte Öffnen von Pop-Up-Fenstern ist generell unter Internetnutzern nicht beliebt und wirft für alle Probleme in der Nutzbarkeit eines Webangebots auf, sofern diese nicht mit der unmittelbaren Nutzung des Angebot zusammenhängen.

Der zweite Teil der Bedingung bezieht sich nicht nur auf selbstöffnende Pop-Ups, sondern auch auf Links, die selbsttätig, d.h. ohne explizite Anforderung durch den Nutzer in einem neuen Fenster geöffnet werden oder Fenster, die den Fokus an sich ziehen.

Für viele Nutzer entsteht bei sich automatisch öffnenden Fenstern das Problem, dass die Ansicht unaufgefordert geändert und der sog. *Clickstream* unterbrochen wird und sich nicht mehr über die History des Browsers (den ›Zurück‹-Knopf) wiederherstellen lässt.

Wenn Ihre Seiten sauberes HTML oder XHTML in der Variante **strict** verwenden, können Sie keine neuen Fenster mehr mit den Bordmitteln von HTML öffnen, da in diesen das hierfür notwendige Attribut `target` nicht mehr spezifiziert ist. Hier müssen Sie im Bedarfsfall auf gleichwertige Funktionen in JavaScript zurückgreifen, die zudem den Vorteil haben, vom Nutzer besser über die Einstellungen seines Browsers kontrollierbar zu sein.

Was heißt das *nicht*?

Fehlermeldungen, die in Form von modalen Dialogen per JavaScript geöffnet werden, sind keine Pop-Ups im Sinne dieser Bedingung. Sie sind zum einen unproblematisch und zum anderen können oder sollten sie sogar funktionaler Bestandteil einer Web-basierten Anwendung sein; ein generelles Verbot wäre also unrealistisch.

Ist diese Anforderung noch aktuell?

Nein: auch diese Bedingung ist in den WCAG eine der »Until user agents…«-Guidelines. In der deutschen Übersetzung steht der Zusatz »bis Benutzeragenten es gestatten, die Erzeugung neuer Fenster zu unterbinden«, der in der BITV bekanntlich fehlt.

Sämtliche aktuell auf dem Markt befindlichen Browser (inkl. Internet Explorer) verfügen über Funktionen, mit denen sich Pop-Ups blockieren lassen. In älteren Browsern lassen sich entsprechende Funktionen über Zusatzprogramme oder Erweiterungen nachrüsten. Diese sind in der Regel kostenlos erhältlich, sodaß diese vermeintliche Barriere längst keine mehr ist - die Bedingung ist somit obsolet.

Diese Bedingung stammt aus Zeiten, als das Konzept des *»Tabbed Browsing«* noch nicht erfunden war, das mittlerweile *sämtliche* modernen Browser beherrschen. In diesen hat der Nutzer volle Kontrolle darüber, ob die Angabe eines target="_new" überschrieben wird und sich sämtliche vom Autor vorgegebenen Aktionen in einem neuen Tab, im gleichen Tab oder tatsächlich in einem neuen Fenster öffnen.

Bedingung 10.2: (Prio. 1)

»Bei allen Formular-Kontrollelementen mit implizit zugeordneten Beschriftungen ist dafür Sorge zu tragen, dass die Beschriftungen korrekt positioniert sind.«

Was heißt das?

»Implizit zugeordnete Beschriftungen« bezieht sich auf die Verwendung von `LABEL`-Elementen in Formularen, die nicht ausdrücklich (*explizit*) per `ID` dem Kontrollelement zugeordnet sind. Diese Label umfassen ihr dazugehöriges Kontrollelement und stellen somit eine durch diese Verschachtelung als gegeben angenommene (*implizite*) Verknüpfung her:

```
<label>… <input …></label>
```

Wie üblich führt die BITV hier nicht weiter aus, was zu tun ist und warum. Zur genaueren Einkreisung der Definition von »korrekt positioniert« ist man auf einen Blick in die WCAG 1.0 angewiesen:

»Die Beschriftung muss unmittelbar vor dem Kontrollelement in derselben Zeile stehen (was mehr als ein Kontrollelement mit Beschriftung pro Zeile gestattet) oder in der Zeile vor dem Kontrollelement (mit nur jeweils einer Beschriftung und einem Kontrollelement pro Zeile).«

Allerdings ist auch diese Bedingung eine Interimslösung für Browser und Hilfsmittel, die die seit über zehn Jahren in der HTML4-Spezifikation definierte Methode der expliziten Zuordnung von Beschriftungen zu ihren Kontrollelementen nicht beherrschen:

```
<label for="foo">…</label> <input id="foo" … >
```

Die Bedingung geht von der veralteten Annahme aus, dass Formulare mit Layout-Tabellen umgesetzt sind. Bei modernen, mit CSS gestalteten Formularen können Sie natürlich mehrere Kontrollelement und ihre dazugehörigen Beschriftungen in eine Zeile setzen, solange diese in irgendeiner Form miteinander verknüpft sind. Dies schliesst auch die ausdrückliche Verknüpfung ein, wie sie in Bedingung 12.4 eingefordert wird. Die Bedingung 10.2 kann als obsolet angesehen werden.

Bedingung 10.3: (Prio. 2)

»Für alle Tabellen, die Text in parallelen Spalten mit Zeilenumbruch enthalten, ist alternativ linearer Text bereitzustellen.«

Was heißt das?

In den Anfangstagen des Web, als Tabellen-Layouts der letzte Schrei waren, hatten damalige Screenreader ein Problem, wenn Textspalten zu nahe beieinander standen. Durch zu geringe Spaltenabstände wurden diese nicht als solche erkannt. Dies konnte dazu führen, dass zwei Überschriften, die in verschiedenen Tabellenzellen in einer Reihe nebeneinander standen, nacheinander vorgelesen wurden und danach dann die erste Zeile der ersten Spalte, die erste Zeile der zweite Spalte, usw.

Streng genommen ist dies kein Problem von Layout-Tabellen per se. Verantwortlich sind hier die inneren Modelle der Screenreader, mit denen sie versuchen, eine Seite zu verstehen und für den Nutzer aufzubereiten. Die gleichen Probleme traten in diesen Screenreadern nämlich auch auf, wenn die Spalten nicht mit Tabellen, sondern mit CSS zu eng beieinander standen. Ursache ist also nicht die Technik der Umsetzung, sondern die zu kompreße Gestaltung des fertig gerenderten Ergebnisses.

Moderne Hilfsmittel diesen Fehler zeigen nicht mehr – Spaltensatz ist als unproblematisch anzusehen, womit auch diese Bedingung hinfällig wäre. Auch im BIENE-Prüfverfahren wird dieser Punkt seit dem Jahr 2006 nicht mehr geprüft.

Bedingung 10.4: (Prio. 2)

»Leere Kontrollelemente in Eingabefeldern und Textbereichen sind mit Platzhalterzeichen zu versehen.« <zitat>

Was heißt das?

Diese Vorgabe war dazu gedacht, für Benutzer assistiver Programme den Umgang mit Textfeldern in Formularen zu vereinfachen. Wenn man der spärlichen Dokumentation des W3C glauben darf, gab es in der Frühzeit des Web angeblich einige Screenreader, die leere Formularfelder nicht erkannten, wenn der Benutzer diese mit Netscape 2 **(!)** ansteuerte.

Spätestens an diesem Punkt sollte für den geneigten Leser offensichtlich sein, warum auch diese Bedingung als veraltet gekennzeichnet ist.

Bedingung 10.5: (Prio. 2)

»Nebeneinanderliegende Hyperlinks sind durch von Leerzeichen umgebene, druckbare Zeichen zu trennen.« <zitat>

Was heißt das?

Mit dieser Bedingung sollte verhindert werden, dass in einer Sprachausgabe Aufreihungen von Links ohne Trennung nacheinander wiedergegeben werden und so nicht mehr differenzierbar sind. Druckbar heißt, dass dies Zeichen aus dem verwendeten Zeichensatz sein müssen, die keine Leerzeichen (Leerschritt oder   ;) sind. Auch Bilder als alleinige Trennung von aufeinanderfolgenden Links sind nicht zulässig.

Dieses Problem spielt heutzutage keine Rolle mehr: sämtliche uns bekannten Hilfsmittel haben keinerlei Probleme damit, Links sauber voneinander zu trennen, selbst wenn diese nicht »durch von Leerzeichen umgebene, druckbare Zeichen« voneinander getrennt sind. Eine Testreihe des W3C aus dem Jahr 1999 weist in die selbe Richtung.

Die Bedienung nebeneinander stehender Links ist für Menschen mit motorischen Behinderungen unter Umständen einfacher, wenn diese etwas weiter voneinander entfernt sind **und** der Nutzer nicht die Tastatur, sondern eine Maus oder ein vergleichbares Zeigegerät nutzt. Allerdings hilft hier der in dieser Bedingung beschriebene Lösungsansatz auch nicht weiter. Auch ein Link, der lediglich drei Pixel von einem benachbarten Links entfernt ist, würde diese Bedingung erfüllen (ein Punkt als druckbares Zeichen, umgeben von Leerschritten, die per CSS auf 1px Breite gebracht wurden).

Negative Beispiele:

Wie wenig sinnvoll diese Bedingung ist zeigt der folgende Screenshot: auf den Seiten des Bundesministeriums für Verkehr, Bau und Stadtentwicklung (BMVBS) wird ein Servicemenü angeboten (was auch immer das sein mag), in dem horizontal aufgereihte Links alle mit einem senkrechten Strich (| = *Pipe*) voneinander getrennt sind. Ein Prüfwerkzeug wie Cynthia Says gibt hier das OK, bemängelt aber bei der darauf folgenden vertikalen Liste von Links, dass diese nicht von druckbaren Trennzeichen umgeben seinen:

Der grosse Unterschied? Die erste Variante ist formell in Ordnung, aber nicht unbedingt benutzbar; die zweite Variante ist unter allen denkbaren Umständen uneingeschränkt nutzbar, aber formell fehlerhaft.

Was meint die BIENE?

Auch im BIENE-Prüfverfahren wird dieser Punkt seit dem Jahr 2006 nicht mehr geprüft. Die Bedingung kann somit als veraltet angesehen werden.

Anforderung 11: Webstandards, Offentlichkeit und Dokumentation verwendeter Techniken

»Die zur Erstellung des Internetangebots verwendeten Technologien sollen öffentlich zugänglich und vollständig dokumentiert sein, wie z. B. die vom World Wide Web Consortium entwickelten Technologien.« <zitat>

Was heißt das?

Die Verwendung von offenen Standards garantiert die Interoperabilität Ihres Internetangebots und spart bei Wartung und Pflege bares Geld.

Man beachte die Einschränkung in dieser Vorgabe (»**wie z.B.** die vom World Wide Web Consortium entwickelten Technologien [sic!]«), in der die W3C-Empfehlungen nur als *eine mögliche* Technik genannt werden. Einer der Gründe für diese Einschränkung ist die Tatsache, daß das W3C keine normative Kraft wie die ISO besitzt, sondern eine privatwirtschaftlich organisierte Vereinigung von Vertretern der Software-Industrie und anderen Interessierten ist. Daher ist es falsch, bei den Empfehlungen des W3C von *Standards* im Sinne einer DIN- oder ISO-Norm zu sprechen, da diese Empfehlungen (Recommendations) nur eines sind: freundliche Empfehlungen.

Dieser Anforderung bedeutet, daß Sie bei Ihren Seiten auf *proprietäres*, d.h. herstellerspezifisches Markup grundsätzlich verzichten sollten. Was dies für die Webentwicklung bedeutet, wird in Anforderung 3 erläutert.

Mit dieser ›Öffnungsklausel‹ schliesst die BITV auch andere de-Fakto-Standards ein, die nicht vom W3C kontrolliert werden: JavaScript ist als ECMA-262 vollständig dokumentiert und normiert worden ebenso wie die PDF- und Flash-Formate vom Hersteller Adobe offengelegt und dokumentiert worden sind und teilweise der ISO zur Normierung vorliegen.

Für eine Verordnung ist es aus formaljuristischen Gründen nicht möglich, sich unmittelbar auf Standards zu beziehen, die nicht der regierungsamtlichen Kontrolle unterliegen. Daher diese vagen Formulierungen, die uns auch an anderen Stellen der Verordnung begegnen.

Bedingung 11.1: Verwendung von W3C-Empfehlungen (Prio. 1)

»Es sind öffentlich zugängliche und vollständig dokumentierte Technologien in ihrer jeweils aktuellen Version zu verwenden, soweit dies für die Erfüllung der angestrebten Aufgabe angemessen ist.« <zitat>

Was heißt das?

Die BITV weicht in diesem Punkt erheblich von der originalen Formulierung in den WCAG 1.0 ab. Dort steht:

»Verwenden Sie W3C-Technologien, wenn sie verfügbar und der Aufgabe angemessen sind und benutzen Sie die neueste Version, wenn sie unterstützt wird.« <zitat>

Aus Sicht des W3C ist diese Betonung der eigenen Empfehlungen sicher verständlich und zum Zeitpunkt der Entstehung gab es kaum Alternativen. Die BITV hingegen spricht lediglich davon, dass »zugängliche und vollständig dokumentierte Technologien [sic!]« zu verwenden sind. Dies bedeutet entgegen der gelegentlich zu lesenden Interpretation **nicht**, dass man keine Hersteller-gebundenen und mit (Patent-)Rechten geschützte Formate benutzen darf.

Auch die Empfehlungen des W3C unterliegen einer sog. »Patent Policy« und sind somit nur innerhalb der festgelegten Rahmenbedingungen wirklich »frei«. Die Grenzen verlaufen somit zwischen »öffentlich zugänglich und vollständig dokumentiert« vs. »geheim und undokumentiert« und **nicht** zwischen »proprietär« vs. »Open Source«. Niemand würde ja auch ernsthaft verlangen, dass man keine GIF- oder JPEG-Dateien benutzen solle, weil diese nicht durch die Gremien des W3C ratifiziert wurden. Der Nachweis, dass GIF für Menschen mit Behinderung eine unüberwindbare Hürde darstellt während PNG hervorragend zugänglich ist, dürfte kaum erbracht werden können. Die meisten Barrieren liegen also nicht in der eingesetzten Technik, sondern daran, was man mit ihr macht.

Bei der Forderung, die jeweils aktuelle Version einer Empfehlung zu benutzen unterscheiden sich BITV und WCAG 1.0 nur durch die etwas andere Formulierung – wenn man als Anbieter davon ausgehen muss, dass eine Technik von der überwiegenden Anzahl der Nutzer verwendet werden kann, dann muss man diese benutzen, statt vergleichbare, jedoch veraltete Standards beizubehalten.

Dies bedeutet nicht, dass man seit dem Erscheinen von XHTML 1.1 kein XHTML 1.0 oder HTML 4.01 verwenden darf. XHTML 1.1 ist *nicht* die aktuelle Version von HTML (ohne X), dies ist nach wie vor HTML 4.01. Falls es irgendwann tatsächlich

mal ein HTML 5 geben sollte, so wäre dies die aktuelle Version von HTML; sollte XHTML 2 jemals fertig gestellt werden, so wäre dies dann die aktuelle Version von **X**HTML.

WCAG und BITV machen hier die Einschränkung, dass eine modernere Technik »für die Erfüllung der angestrebten Aufgabe angemessen« sein sollte; in der BITV fehlt jedoch der Zusatz »wenn sie unterstützt wird« – hierdurch könnten Webentwickler in die Falle laufen, indem sie Techniken einsetzen, die zwar der *›dernier cri‹* sind, aber von der Mehrzahl der Zugangssoftware nicht oder nur unzureichend unterstützt werden.

Ab wann ist eine Technik angemessen?

Die Empfehlungen des W3C sind grösstenteils mit einem besonderen Augenmerk auf die *Accessibility* entwickelt worden. Eine Aufstellung finden Sie im Abschnitt »Technologies Reviewed for Accessibility« der WCAG 1.0.

Zur Entstehungszeit dieser Richtlinie konnte man dies von Formaten wie PDF oder Flash ganz sicher nicht behaupten: diese Formate konnten aus damaliger Sicht als weitestgehend **un**zugänglich gelten. Die Hersteller hatten das Thema *Accessibility* nicht auf dem Radar und die Formate waren zur damaligen Zeit noch relativ neu, die Hersteller der Hilfsmittel mussten ihre Anwendungen erst an diese Formate anpassen.

Beides hat sich inzwischen umgekehrt: viele nicht-W3C-Formate lassen sich barrierefrei nutzen, wenn bei der Erstellung die Grundregeln beachtet wurden. Ebenso lassen sich diese Formate schon seit einigen Generationen mit den gängigen Hilfsmitteln behinderter Menschen nutzen. Ob diese Formate nun für die »Erfüllung der angestrebten Aufgabe angemessen«, oder ob es nicht doch bessere Alternativen gibt sind steht auf einem anderen Blatt:

Sonderfall Flash

Zur Darstellung interaktiver multimedialer Inhalte ist Flash das Werkzeug der Wahl, da die Möglichkeiten bei weitem über das hinausgehen, was man mit statischen Texten erreichen kann. Durch die Einbettung von Videos per Flash kann sich ein Webdesigner eine Menge Arbeit ersparen und der Nutzer braucht sich nicht mit unterschiedlichen Playern und fehlenden Codecs herumzuschlagen. Zudem lassen sich Videos und Multimedia-Präsentationen recht einfach mit Untertiteln versehen, sehr schöne Beispiele hierfür gibt es beim amerikanischen Qualitätsfernsehen PBS sowie bei click.tv.

Ungeeignet ist Flash hingegen zur reinen Darstellung grösserer Mengen von reinen strukturierten Texten ohne zusätzliche interaktive Elemente – hier ist herkömmliches HTML definitiv die bessere Wahl. Eine hervorragend umgesetzte Kombination von Texten und Videos zeigt die BIENE-prämierte Seite der Firma Pfizer: »Gebrauchsinformation, Packungsbeilage oder Beipackzettel - Tipps, wie man ihn liest!«

Sonderfall PDF

Eine Abgrenzung, für welche Anwendungszwecke das PDF-Format angemessen ist und für welche nicht finden Sie in unserem Artikel »Fakten und Meinungen zur Barrierefreiheit von PDF«.

Bedingung 11.2: Verzicht auf veraltete Elemente und Attribute (Prio. 1)

»Die Verwendung von Funktionen, die durch die Herausgabe neuer Versionen überholt sind, ist zu vermeiden.« <zitat>

Was heißt das?

Älteren Auszeichnungssprachen wie HTML 3.2 ist damit die Geschäftgrundlage entzogen, da es mit HTML 4.01 und XHTML 1.x bereits seit längerem neuere und modernere Version gibt. Diese Bedingung gilt insbesondere für Dinge aus älteren HTML-Versionen wie `<font>` und ähnliche Elemente, die in HTML 4 als *deprecated* (sinngemäß: veraltet) gekennzeichnet sind oder gar nicht mehr spezifiziert sind. Ebenso gilt diese Bedingung für veraltete Attribute, mit denen Struktur und Aussehen vermischt wird, wie zum Beispiel Attribute mit Angaben zur Gestaltung von Inhalten (z. B. `bgcolor`, `hspace`, `vspace` u.v.m.), die besser im CSS Ihrer Seiten aufgehoben sind. Diese Elemente und Attribute sind zwar in HTML 4 *Transitional* und *Frameset* noch spezifiziert, nicht jedoch in der Variante *Strict* und sie werden in zukünftigen Versionen von HTML sicherlich ganz entfallen.

Übrigens: entgegen landläufiger Meinung sind die Elemente `B` und `I` **nicht veraltet** (*deprecated*), sondern sie sind aus gutem Grund nach wie vor Bestandteil der HTML-Spezifikation, und dies sogar in der Variante *Strict*.

Die folgenden Elemente sind veraltet, an ihrer Stelle sollte CSS verwendet werden, mit dem man identische oder in vielen Fällen sogar bessere Effekte erzielen kann: `BASEFONT`, `CENTER`, `FONT`, `S`, `STRIKE` und `U`. Ebenfalls veraltet sind die folgenden Elemente, an Ihrer Stelle sollten neuere Elemente benutzt werden, die wir in Klammern dahinter aufgelistet haben: `APPLET` (→ `OBJECT`), `DIR` (→ `UL`), `ISINDEX` (→ `INPUT`) und `MENU` (→ `UL`).

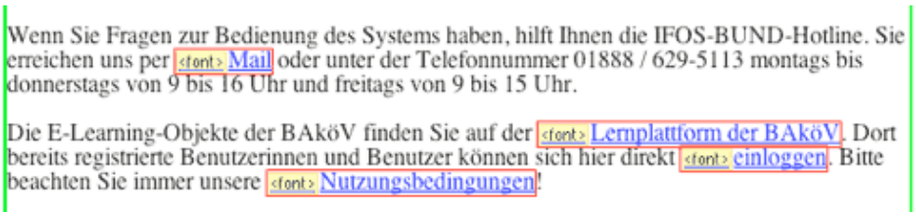
Wie können Sie das testen?

Bei den Tests zu dieser Vorgabe helfen Online-Werkzeuge wie der W3C-Validator: viele Hilfen zur Barrierefreiheit wurden erst mit der Version 4 in die Seitenbeschreibungssprache HTML eingeführt. Wenn der

Validator bei der Überprüfung Ihrer Seiten zu dem Ergebnis kommt, daß es sich um HTML 3.2 oder noch älteres handelt, ist leider einiges an Nacharbeit nötig. Obwohl die Seiten formal nach diesen veralteten Empfehlungen validieren können, sind sie damit noch nicht automatisch auf der technischen Ebene barrierefrei – eher das Gegenteil ist wahrscheinlich der Fall.

Selbst valide Seiten können Elemente und Attribute enthalten, die als nicht mehr zeitgemäß gelten und insofern nicht mehr verwendet werden sollten, da aus ihrer Verwendung heraus Barrieren entstehen können. Wie so oft hilft hier bei der Suche die Web Developer Toolbar für den Firefox: im Menü ›Outline‹ finden Sie den Befehl ›Outline Deprecated Elements‹, der Ihnen die Verwendung veralteter Elemente anzeigt.

Ein weiterer einfacher Test auf die Verwendung veralteter Funktionen ist das Abschalten von Style Sheets. Wenn sich dabei, wie zum Beispiel bei den Seiten der ›Bundesakademie für öffentliche Verwaltung im Bundesministerium des Inneren‹ (BAköV), lediglich die Schriftgröße oder die Linkfarbe ändert, dann stehen zu viele veraltete Formatierungsanweisungen im HTML der Seite und zu wenige im CSS:



Beispiele:

Die Verwendung der W3C–Vorgaben bringt noch einen nicht zu unterschätzenden Vorteil: nicht nur dass Ihre Seiten insgesamt schlanker und damit performanter werden, auch die nachträgliche Pflege wird stark vereinfacht. Ein mit der Pflege einer Website beauftragter Entwickler braucht sich nicht mehr in den individuellen Code des ursprünglichen Programmierers einzuarbeiten, sondern kann sein Wissen der Standard-konformen Entwicklung sofort produktiv einsetzen.

Wenn man mit Entwicklern spricht, die den Schritt zu sauberem Code und den gängigen Maßnahmen zur Barrierefreiheit vollzogen haben, hört man immer wieder die selben Antworten: der Lernaufwand ist zwar zunächst vielleicht hoch, vor allem, weil man einige Gewohnheiten ablegen und mühsam erworbenes Wissen um verschachtelte Tabellen-Layouts ›entlernen‹ muß. Aber trotzdem bereut kaum jemand diesen Schritt, da er ein wesentlich einfacheres, strukturierteres Arbeiten ermöglicht.

Bedingung 11.3: Seiten für alle (keine Nur-Text-Version) (Prio. 1)

<zitat>

»Soweit auch nach bestem Bemühen die Erstellung eines barrierefreien Internetangebots nicht möglich ist, ist ein alternatives, barrierefreies Angebot zur Verfügung zu stellen, dass äquivalente Funktionalitäten und Informationen gleicher Aktualität enthält, soweit es die technischen Möglichkeiten zulassen. Bei Verwendung nicht barrierefreier Technologien sind diese zu ersetzen, sobald aufgrund der technologischen Entwicklung äquivalente, zugängliche Lösungen verfügbar und einsetzbar sind.«

(**Anmerkung:** die Bedingungen 11.3 und 11.4 wurden in der BITV vertauscht, d. h. BITV 11.3 entspricht WCAG 1.0 11.4, BITV 11.4 entspricht WCAG 1.0 11.3)

Was heißt das?

Eine der problematischsten Bedingungen der gesamten BITV. Lädt sie doch dazu ein, kurzerhand das Scheitern der Bemühungen um einen integrierten barrierefreien Webauftritt zu erklären und Menschen mit Behinderung mit einer reduzierten und vermeintlich barriereärmeren Sonderlösung abzuspeisen. Die Begründung zur BITV schliesst zwar die berühmt-berüchtigten »barrierefreien Nur-Text-Versionen« ausdrücklich aus:

<zitat>

»Grundsätzlich zielt die Verordnung darauf, Sonderlösungen für behinderte Menschen oder für einzelne Gruppen behinderter Menschen zu vermeiden. Die Erstellung eines Internetangebots, das für alle Benutzergruppen gleichermaßen uneingeschränkt nutzbar ist, hat Vorrang insbesondere vor einer ›Nur-Text-Lösung‹ als Alternative zum eigentlichen Internetangebot, da eine solche Darstellung in erster Linie nur für bestimmte Benutzergruppen von behinderten Menschen, etwa für Benutzer von Braille-Zeilen oder Screen-Readern, Barrierefreiheit erreicht.

Gerade durch die Bedingung 11.3 wird jedoch die Möglichkeit geschaffen, solche Sonderwege zuzulassen. Und es wird genutzt, auch und gerade im Geltungsbereich der BITV. Das Ergebnis ist dann häufig nicht ein vollkommen unzugängliches Haupt-Angebot mit einer leidlich zugänglichen Alternative, sondern zwei suboptimale parallele Angebote, die beide nicht zu benutzen sind.

Das solche Szenarien nicht ganz unrealistisch sind zeigen die Websites von einigen Behörden, die voll und ganz zur Einhaltung der BITV verpflichtet sein müssten. Auch im Jahr 2007 finden sich dort Inhalte in abenteuerlichen Code-Konstruktionen wie dieser:

```
<center>
<table>
<tr>
<td valign=top>
<table border="0" cellspacing="0" cellpadding="0" width="700" height="424"
background="bilder/willkommen.gif">
 <tr>
   <td align="center">
<br><br><br><br><br>
<br><br><br><br><br>
<a href="Barrierefrei/index.htm"><img
 border=0 src="bilder/Barrierefrei.gif"
 width="54" height="30" alt="Barrierefrei" hspace=10></a>
<a href="de/de.htm"><img
 border=0 src="bilder/splash_germ.gif"
 width="54" height="30" alt="Zur deutschen Homepage" hspace=10></a>
<a href="en/en.htm"><img
 border=0 src="bilder/splash_brit.gif"
 width="54" height="30" alt="Zur englischen Homepage" hspace=10></a>
<br><br>
<font size="-2" color="white">
<br><a class="sub" href="Barrierefrei/wir/Barrierefrei.htm">Definition Barrierefrei</a><br>
<br>Für Internetexplorer und Netscape ab Version 4 optimiert.
<br>(Bei der Benutzung anderer Browser kann es zu Darstellungsproblemen kommen.)
</font>
</td>
</tr>
</table>
</td>
</tr>
</table>
</center>
```

Nimmt man den Code weg ist der einzig verwertbare Text dieser Seite:
Barrierefrei Zur deutschen Homepage Zur englischen Homepage Definition Barrierefrei Für Internetexplorer und Netscape ab Version 4 optimiert. (Bei der Benutzung anderer Browser kann es zu Darstellungsproblemen kommen.)

Die Überschrift zur Seite »Barrierefrei (Info)« versteckt sich bei besagter Behörde in einem wahren Code-Schmankerl:

```
<td class=headline width="80%" height="40" nowrap bgcolor="#2d4b9b"> <font color="#ffffff" size="+2"><div class=headline>   Barrierefrei</div></font> </td> <td class=sub width="10%" height="40" bgcolor="#2d4b9b">  </td>
```

Hier stellt sich die Frage, warum man überhaupt ein Parallel-Angebot mit »äquivalenten Funktionalitäten und Informationen gleicher Aktualität« bereitstellen sollte. Wenn letzteres die identischen Inhalte nur in anderer Form zur Verfügung stellt, wäre es dann nicht sinnvoller, gleich auf die Barrieren zu verzichten?

Wie können Sie das testen?

Die Abgrenzung zwischen einer unzulässigen Sonderlösung und anderen Formen der alternativen Aufbereitung von Inhalten ist nicht so ohne weiteres zu machen. So kann je nach eingesetzter Technik ein durchaus zulässiger Style Switcher den URL der Seite so verändern, dass diese auf einem eigenständigen Pfad liegt, wie dies auch bei einer Sonderlösung der Fall wäre – dies ist also kein hinreichendes Unterscheidungsmerkmal. Die Bewertung sollte eher auf der konzeptionellen Ebene erfolgen: wie wird generell mit dem Thema »Intergration von Menschen mit Behinderungen« umgegangen? Werden diese zur Nutzung eines gesonderten Angebotes angehalten (z. B. durch entsprechend beschriftete Links in der Navigation) oder ist gleichberechtigte Nutzung des Angebots durch alle Besucher vorgesehen.

Diese Kritik an der Bedingung 11.3 bezieht sich ausdrücklich nicht auf Ansätze, die für spezielle Bedürfnisse behinderter Menschen gedacht sind und zum Beispiel alternative Formen wie Videos in Gebärdensprache oder Texte in Leichter Sprache anbieten. Eine Nur-Text-Version für blinde Nutzer hingegen ist nicht notwendig, denn sauberes HTML ist von Hause aus nichts anderes: strukturierter Text.

Bedingung 11.4: Content Negotiation (Prio. 2)

> Der Nutzerin/dem Nutzer sind Informationen bereitzustellen, die es ihnen erlauben, Dokumente entsprechend ihren Vorgaben (z. B. Sprache) zu erhalten.« <zitat>

(**Anmerkung:** die Bedingungen 11.3 und 11.4 wurden in der BITV vertauscht, d. h. BITV 11.3 entspricht WCAG 1.0 11.4, BITV 11.4 entspricht WCAG 1.0 11.3)

Was heißt das?

Kurzfassung: das wissen wir selbst nicht.

Langfassung: die Dokumentation zu dieser Bedingung bzw. zum Original in den Web Content Accessibility Guidelines rangiert zwischen *ausgesprochen mager* und *technisch fragwürdig*. Problematisch ist diese Bedingung gleich in mehrfacher Hinsicht: sie besagt *nicht*, dass man bereits vorhandene Versionen in anderen Sprachen verlinken muss – dies versteht sich eigentlich von selbst und ist auch eher anderen Bedingungen zum Thema Navigation zuzuordnen.

Die Bedingung dreht vielmehr den Spieß herum und verlangt, dass Anbieter auf eventuell voreingestellte Sprachpräferenzen des Nutzers eingehen muss und (ohne Einschränkung) übersetzte Versionen von Inhalten ausliefern muss, sobald ein Nutzer zum Beispiel Swahili als seine bevorzugte Sprache eingestellt hat. Die Interpretation, dass mit dieser Bedingung auch Versionen in Leichter Sprache oder Gebärdensprache gemeint sein könnte ist ebenfalls unzutreffend, da der Nutzer hierüber keine »Vorgaben« machen kann, wie dies von der BITV als Auslöser für diese Bedingung beschrieben wird.

Anforderung 12: Kontext- und Orientierungsinformationen

»Der Nutzerin/dem Nutzer sind Informationen zum Kontext und zur Orientierung bereitzustellen.«

Was heißt das?

Für viele Menschen mit Behinderungen bestehen die Barrieren nicht in der technischen Erreichbarkeit von Webinhalten, sondern im Verständnis, der Orientierung und der Bedienung wenn man »schon drin ist«. Die Nutzung der Inhalte wird oft durch die mangelnde Übersichtlichkeit und Bedienbarkeit einer Website verwehrt. Eine Absicht des Anbieters steckt in den seltensten Fällen dahinter, sondern eher das mangelnde Wissen um die speziellen Bedürfnisse und die Funktionsweise der assistiven Werkzeuge behinderter Menschen.

Ohne das Wissen um die Funktionen z. B. eines Screenreaders liest sich diese Anforderung zunächst erst einmal wie eine Selbstverständlichkeit. Jede Website stellt doch über die Navigation Informationen zur Orientierung bereit, oder? Dass die Reduzierung von Barrieren jedoch nicht alleine durch das bloße Vorhandensein von Navigationsmechanismen und Bedienelementen sichergestellt ist, wird durch die folgenden Bedingungen verdeutlicht:

Bedingung 12.1: Frames mit Titel und Name (Prio. 1)

Jeder Frame ist mit einem Titel zu versehen, um Navigation und Identifikation zu ermöglichen.«

Was heißt das?

Für die Verwendung der Rahmentechnik (sog. *Frames*) kann es trotz vieler Vorbehalte in einzelnen Fällen durchaus begründete Ausnahmen geben. Allerdings gilt es dabei einige Dinge zu beachten, die sich aus der Funktionsweise nicht-grafischer Zugangsarten ableiten.

Für sehende Nutzer ergeben sich Sinn und Position der einzelnen Frames (*Rahmen*) in einem Frameset schon alleine aus der optischen Anordnung; für blinde Nutzer von Screenreadern, aber auch in einer Vergrößerungssoftware mit Sprachausgabe sind diese Funktionen nicht ohne weiteres ersichtlich. Damit diese Nutzer nicht erst durch Erkundung der Inhalte raten müssen, welche Funktion ein Frame hat, sollten diese mit Attributen ausgestattet sein, die eine Orientierung erleichtern.

In der HTML-Spezifikation sind hierfür die `frame`-Attribute `name` und `title` vorgesehen. `Name` ist eigentlich nur für die Ansteuerung der Frames gedacht, der Wert des Attributs wird jedoch von älteren Hilfsmitteln in Kombination mit älteren Browsern ausgegeben und sollte daher sinnbildend sein. Vermeiden sollten Sie, die Frames im `name`-Attribut nach ihrer Position zu benennen (`name="linksaussen"`), stattdessen sollten Sie die Funktion hinterlegen (`name="navigation"`). Das Universal-Attribut `title` erlaubt auch längere sinnbildende Beschreibungen, auch hier liegt die Würze in der Kürze.

Beispiele:

Das folgende Codelisting zeigt den beispielhaften Aufbau eines Framesets mit `name`- und `title`-Attribut. Zusätzlich ist für einen der Frames das in der folgenden Bedingung 12.2 geforderte `longdesc`-Attribut hinterlegt:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
  "http://www.w3.org/TR/html4/frameset.dtd">
<html>
 <head>
  <title>Frameset</title>
 </head>
 <frameset cols="25%,75%">
  <frame src="navi.html"
   name="navigation" title="Navigationsleiste"
   longdesc="framebeschreibung.html">
  <frame src="start.html"
   name="inhalt" title="Inhaltsbereich">
  <noframes>
    […]
  </noframes>
 </frameset>
</html>
```

Wie können Sie das testen?

Die Mozilla Accessibility Extension für Firefox und Verwandte bietet zu diesem Zweck den Befehl ›Frames‹ im Menü ›Navigation‹. Falls in einem Frameset die notwendigen Attribute `name` oder `title` fehlen wird Ihnen dies in einem Dialog angezeigt.

Wenn Sie Ihre Frame–Seiten auch mit assistiven Werkzeugen testen möchten: der kostenlose WebFormator der Firma Frank Audiodata zeigt die Struktur eines Framesets inklusive der (hoffentlich korrekten) Benennungen der Frames und der in den einzelnen Frames vorgefundenen Textinhalte an.

Bedingung 12.2: Aufbau der Frames beschrieben (Prio. 1)

> Der Zweck von Frames und ihre Beziehung zueinander ist zu beschreiben, soweit dies nicht aus den verwendeten Titeln ersichtlich ist.« <zitat>

Was heißt das?

Zusätzlich zu den in Bedingung 12.1 genannten Attributen `name` und `title` *kann* ein `FRAME` oder `IFRAME` mit einer ausführlichen Beschreibung über das `longdesc`-Attribut versehen werden, dass auf einen URL mit einem erklärenden Text verweist.

Sollte Ihr Aufbau jedoch so kompliziert sein, dass eine weitergehende Erklärung zum Verständnis unbedingt vonnöten ist, dann sollten Sie ernsthaft über eine Vereinfachung nachdenken. Diese Bedingung regelt also streng genommen Probleme, die in der gesamten Architektur des Angebots entstanden sind und somit auch besser dort repariert werden sollten.

Diese Bescheibung gilt nur für den ursprünglichen Zustand des Framesets; sobald die Nutzerin den Zustand des Framesets z.B. durch Anklicken eines Links verändert, stimmt die Beschreibung des ursprünglichen Framesets nicht mehr. Da diese Beschreibung (per `longdesc`) jedoch im aufrufenden Frameset hinterlegt ist, kann sie nicht geändert werden, ohne das komplette Frameset neu zu laden. Dann wiederum ist auch der letzte verbliebene Vorteil von Frames dahin.

Wie können Sie das testen?

Am besten gar nicht. Bauen Sie einfach keine Websites mehr mit Frames.

Bedingung 12.3: Inhalt gegliedert (Prio. 1)

> Große Informationsblöcke sind mittels Elementen der verwendeten Markup–Sprache in leichter handhabbare Gruppen zu unterteilen.« <zitat>

Was heißt das?

Bei der Gestaltung mit CSS geschieht diese Unterteilung in handhabbare Gruppen schon fast zwangsläufig. Diese Sprache entfaltet ihre ganze Eleganz erst, wenn der Code nach einem festen Regelwerk strukturiert ist. Diese Regeln werden von externen Style Sheets benötigt, um eine Seite überhaupt effizient formatieren zu können. Allerdings sind die hierfür verwendbaren Container-Tags wie `<div>` auf der semantischen Ebene eher geschmacksneutral und werden von assistiven Werkzeugen auch nicht in ihrer vermeintlich strukturierenden Form erkannt.

Wir haben hier also eine Vorgabe, die streng genommen erst erfüllbar ist, wenn in ferner Zukunft einmal HTML 5 und/oder XHTML 2 verabschiedet werden. Dort wird es nach dem jetzigen Stand der Entwicklung erstmalig Elemente geben, die eigenständige Bereiche voneinander abtrennen und die eine ihrer Funktion entsprechende Bedeutung in sich tragen. Bis dahin bietet HTML ausgesprochen wenige Elemente, um »Große Informationsblöcke« in handlichere Portionen zu unterteilen.

Ausnahmen, die im Kontext dieser Bedingung sinnvoll erscheinen:

- das `FIELDSET`-Element zur Strukturierung von mehrteiligen Formularen,
- das `OPTGROUP`-Element zur Abgrenzung einer Gruppe von Einträgen in einer Menüstruktur,
- das `HR`-Element zur Unterteilung von größeren, inhaltlich nicht zusammenhängenden Blöcken einer Seite.

Weitere in den Techniken der WCAG an dieser Stelle aufgeführte Elemente wie (Daten-) Tabellen, Listen und Überschriften sind bereits an anderer Stelle hinreichend gewürdigt und somit im Kontext dieser Bedingung irrelevant. Auch die in anderen Veröffentlichungen bei dieser Bedingung aufgeführten Elemente wie `STRONG` oder `EM` sind für die Unterteilung ***großer Informationsblöcke*** alleine schon deswegen ungeeignet, weil es sich um Inline-Elemente handelt, die nur innerhalb solcher Blöcke vorkommen dürfen.

Beispiele:

Ein Beispiel für HTML-Elemente, die in ihrer strukturierenden Funktion eingesetzt werden sollten, findet man häufig in Formularen mit langen Auswahllisten (`SELECT`). Im Extremfall stehen in einer Länderauswahl sämtliche Staaten dieser Erde, was die Navigation innerhalb der Liste nicht vereinfacht, sofern man nicht im oberen Drittel des Alphabets wohnhaft ist. Durch die Gruppierung mittels `OPTGROUP` können die Länder nach Kontinenten zusammengefasst werden, Produkte nach Hersteller oder Produktgruppen u.v.m.:

```
<form action="foo">
<fieldset>
 <legend>Länderauswahl</legend>

 <select id="bar" name="bar">

  <optgroup label="Asien">
   <option>Brunei</option>
   <option>…</option>
  </optgroup>

  <optgroup label="Afrika">
   <option>Burkina Faso</option>
   <option>…</option>
   </optgroup>

  <optgroup label="Europa">
   <option>Belgien</option>
   <option>…</option>
  </optgroup>

 </select>

</fieldset>
</form>
```

Bedingung 12.4: Label mit Formularelementen verknüpft (Prio. 1)

Beschriftungen sind genau ihren Kontrollelementen zuzuordnen.« <zitat>

Was heißt das?

Sie kennen diese Funktion aus Ihrem Betriebssystem: bei Radiobuttons oder Checkboxen müssen Sie nicht unbedingt das Element selbst anklicken; meistens reicht es, wenn Sie auf den danebenstehenden Text (das sog. Label) klicken und schon ist das Häkchen gesetzt. Genau das geht seit HTML 4 auch mit allen Elementen in Web–Formularen, nur wird es viel zu selten eingesetzt.

Menschen mit motorischer Behinderung wird durch diese Labels die Möglichkeit gegeben, ein wesentlich größeres Ziel zu treffen (Fitts' Law) oder das Formular per Tastatur zu bedienen. Die Orientierung in nicht-visuellen Ausgabeformen wird durch die eindeutige Zuordnung erst ermöglicht.

Beispiele:

In der Gestaltung Ihrer Formulare sollten Sie sich von den Interface-Konventionen der großen Betriebssysteme leiten lassen, da diese den Benutzern vertraut sind. So sollten die Labels von Checkboxen und Radiobuttons im Idealfall *immer unmittelbar **rechts*** neben dem jeweiligen Kontrollelement stehen, bei Texteingabefeldern kann dies sowohl links daneben als auch unmittelbar darüber sein. Im Magazin von ›Einfach für Alle‹ finden Sie mehrere Tutorials, wie Sie Formulare barrierefrei und benutzerfreundlich gestalten und programmieren können.

Das folgende Codelisting zeigt die Verknüpfung eines Formularelementes, in diesem Fall eines Textfelds zur Eingabe einer E-Mail-Adresse mit der dazugehörigen Beschriftung:

```
1. <labelfor="email">Ihre <spanlang="en">E-Mail</span>:</label>
2. <inputtype="text"id="email">
```

Dabei unterscheidet HTML zwischen zwei Formen der Verknüpfung:

1. der expliziten, d.h. der ausdrücklichen Verknüpfung über die Attribute `for` für das Label und `id` für das Formularlement sowie
2. der impliziten, d.h. der sich zwangsläufig ergebenen Verknüpfung durch eine entsprechende Verschachtelung der Elemente, wie das folgende Beispiel zeigt:

```
1. <label>Vorname:
2. <inputtype="text">
3. </label>
```

Wie können Sie das testen?

Diesmal dürfen Sie mit einer Maus testen und brauchen nichts in Ihrem Browser abzuschalten! Nehmen Sie sich einfach alle Formulare Ihrer Website nacheinander vor und testen Sie, ob die Kontrollelemente über den Klick auf den danebenstehenden Text aktivieren lassen. Wenn Radiobuttons (Entweder–Oder–Auswahlen) in einem Formular vorhanden sind, sollten es logischerweise immer mindestens zwei sein, Checkboxen können auch alleine auftreten.

Bei Textfeldern sollte die Einfügemarke nach dem Klick auf das Label im entsprechenden Textfeld stehen, so dass Sie sofort mit der Eingabe beginnen können. Auswahlboxen werden bei der Aktivierung meistens durch eine Umrandung hervorgehoben, die Auswahl selber lässt sich dann mit den Pfeiltasten (rauf/runter) vornehmen. Achten Sie bei Ihren Tests auch darauf, dass durch den Klick das richtige, d.h. das zu dem angeklickten Label gehörende Kontrollelement aktiviert wird.

Anforderung 13: Gestaltung von Navigationsmechanismen

»Navigationsmechanismen sind übersichtlich und schlüssig zu gestalten.« <zitat>

Viele Usability–Faktoren haben einen ebenso starken Einfluß auf die Barrierefreiheit eines Angebots wie die bislang besprochenen technischen Maßnahmen. Während die rein technischen Merkmale weitgehend maschinell, d.h. mit Validatoren oder zumindest mit der Unterstützung durch Prüfwerkzeuge verifizierbar sind, lassen die Anforderungen 13 und 14 einigen Spielraum für Interpretationen.

Dabei ist die Usability im Kontext dieser Verordnung ausschließlich als die spezifische Gebrauchstauglichkeit für Menschen mit Behinderungen zu verstehen. Darüber hinaus gehende Usability-Kriterien für *alle* Nutzer und Nutzungsformen kann und sollte ein Webangebot selbstverständlich ebenso erfüllen, dies ist jedoch **nicht Gegenstand der BITV**.

Diese Einschränkung bedeutet jedoch nicht, dass die folgenden »weichen« Bedingungen weniger Gewicht haben als die vorhergehenden technischen Anforderungen an eine barrierefreie Webpräsenz. Die nicht-Erfüllung der folgenden Bedingungen stellt für viele Menschen mit motorischen oder kognitiven Behinderungen eine unter Umständen wesentlich gravierendere Barriere dar als der ein oder andere vergessene Alternativtext.

Zu beachten ist auch, dass einige dieser Bedingungen (insbesondere 13.6 und 13.10) von der technischen Entwicklung überholt worden sind und als veraltet einzustufen sind. Bei anderen Bedingungen (z.B. 13.2) sind lediglich die in den WCAG genannten Techniken veraltet, die Bedingung selbst kann auf neuere Entwicklungen wie Microformats genauso angewendet werden.

Bedingung 13.1: Eindeutige Linktexte (Prio. 1)

»Das Ziel jedes Hyperlinks muss auf eindeutige Weise identifizierbar sein.« <zitat>

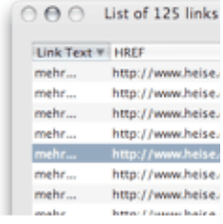
Was heißt das?

Gemäß dieser Bedingung muss der Text eines Hyperlinks auch ohne seinen Satzzusammenhang noch zuverlässige Aussagen über das Ziel des Links machen. Der Hintergrund für diese Bedingung ist, dass einige Hilfsmittel die Links einer Seite zusammenfassen und getrennt darstellen können. Wenn Ihre Links nicht eindeutig benannt sind, sind sie mit diesen Funktionen nur eingeschränkt brauchbar.

Besonders wichtig ist dies auf Seiten, die eine verteilende Funktion haben und die der allgemeinen Orientierung innerhalb eines Angebots dienen, also z.B. eine Startseite, ein Inhaltsverzeichnis oder eine Sitemap. Links in Navigationsleisten erfordern ebenfalls eine strikte Anwendung dieser Bedingung, da die Vermittlung des Linkziels der eigentliche Zweck einer Navigationsleiste ist.

Was heißt das *nicht*?

Weniger sinnvoll ist diese Bedingung für Links im eigentlichen Inhalt eines Angebots, also zum Beispiel in Fließtexten. Das World Wide Web ist ein Hypertext-basiertes Medium und die Verlinkung aus dem Kontext heraus ist eine der wichtigsten Grundlagen des Web. Hier liegt es in der Natur der Dinge, dass Links oftmals nur im Kontext sinnbildend sein können. Für den Nutzer ist es keine besondere Erschwernis, sich im Kontext eines Links zu orientieren und aus dem umgebenden Text Schlüsse zu ziehen, wohin der Link zeigt. Dazu gehört auch, dass mehrere Links mit den gleichen Linktexten auf unterschiedliche Ressourcen zeigen dürfen, wenn sich der Unterschied aus dem Kontext ergibt.



Die hin und wieder im Netz zu findende Argumentation, dass sich blinde Nutzer vornehmlich und in Ermangelung anderer Mittel anhand von Linklisten innerhalb einer Seite orientieren, können wir nicht folgen. Abgeleitet wird dies aus der in den Techniken zur WCAG 1.0 beschriebenen Annahme, dass Links zur Orientierung *in* einer Seite geeignet seien – eine ausgesprochen fragwürdige Annahme: Links beschreiben von Ihrer Natur her *nicht* die Seite, in der sie eingebettet sind (und schon gar nicht deren Struktur), sondern verweisen auf *andere* Ressourcen jenseits der aktuellen Position im Inhalt. Zudem stehen dem nicht-sehenden Nutzer durch Überschriften (vgl. BITV 3.5) und andere Strukturelemente (vgl. BITV 3.6 und 3.7) genügend und vor allem geeignetere Mittel zur Verfügung, um sich einen Überblick über die Inhalte zu verschaffen, der dem visuellen ›Überfliegen‹ einer Seite durch den sehenden Benutzer adäquat ist.

Weitergehende Eigenschaften wie der Dateityp eines verlinkten Dokumentes sollten entweder im Zusammenhang mit dem Link erwähnt bzw. per Icon angezeigt werden oder mit zusätzlichen Attributen des Links kenntlich gemacht werden. Die Anzeige des Universal-Attributs `title` ist jedoch in vielen Benutzeragenten optional und in grafischen Browsern bei reiner Taststurbedienung generell nicht zugänglich. Das Attribut zur Definition eines Mime-Types einer verlinkten Ressource (`<a`

`href="foo.pdf" type="application/pdf">` …) wird wiederum nach unserem Wissenstand von assistiven Programmen nicht ausreichend unterstützt.

Wie können Sie das testen?

Die Mozilla Accessibility Extension für Firefox und Verwandte bietet zu diesem Zweck den Befehl ›Links‹ im Menü ›Navigation‹. In einem Dialog werden sämtliche Links einer Seite in Listenform angezeigt; zusätzlich werden weitere Informationen wie eventuelle `title`-Attribute zur Differenzierung, Linktyp (Protokolle wie `mailto:`), die Tabreihenfolge und der URL aufgelistet.

Bedingung 13.2: Metadaten (Prio. 1)

»Es sind Metadaten bereitzustellen, um semantische Informationen zu Internetangeboten hinzuzufügen.« <zitat>

Was heißt das?

»Semantik« bezeichnet in der Sprachwissenschaft die Lehre von der Bedeutung der Dinge; dieser Begriff wird im Web etwas liberaler genutzt, um beschreibende Informationen eines Webangebots zu definieren.

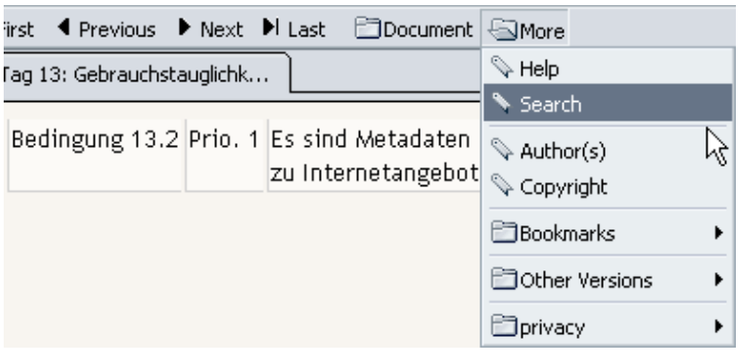
»Metadaten« wiederum sind Daten, mit denen die eigentlichen Inhalte und Funktionen eines Angebots auf einer höheren (*Meta-*) Ebene beschrieben werden. HTML kennt zu diesem Zweck eine ganze Reihe Elemente und Attribute, allerdings reicht deren Unterstützung in Benutzeragenten und anderen Werkzeugen wie Suchmaschinen von ›gebräuchlich‹ über ›lückenhaft‹ bis ›gar nicht‹ oder ›experimentell‹.

Der Bereich der Metadaten beginnt streng genommen bereits beim Titel eines Dokuments: in vielen assistiven Anwendungen behinderter Menschen ist der Inhalt von `<title>` das erste, was beispielsweise nach dem Laden einer Seite vorgelesen wird. Zudem ist dies der Text, der in den meisten Browsern für ein Lesezeichen (Favorit oder Bookmark) benutzt wird, wenn ein Besucher sich ein solches Bookmark für ein Dokument setzt. Dieser Text muss bereits eindeutige Schlüsse zulassen, was den Nutzer auf der betreffenden Seite erwartet.

Beispiele: Metadaten für Maschinen

Zu den beschreibenden Metadaten gehören Informationen zu Zweck und Position der jeweiligen Seite innerhalb eines Angebots. Wenn ein Dokument ein Bestandteil einer Serie ist (wie dieses Dokument hier), so sollten Sie die Hierarchie dieser Serie mit sogenannten *Link Relations* abbilden. Mögliche Link-Typen hierfür sind `<link rel="start">`, `<link rel="previous">`, `<link rel="next">` und `<link rel="last">` im `HEAD` der Dokumente. Diese Metadaten können auch Verweise auf ein Glossar (`rel="glossary"`), Hinweise zum Datenschutz (`rel="privacy"`), ein Inhaltsverzeichnis (`rel="contents"`), eine Suchfunktion (`rel="search"`) oder eine Hilfe-Seite (`rel="help"`) sein.

Aktuelle Browser wie Opera, Mozilla und Exoten wie iCab stellen diese Link Relations in einer (allerdings nur optional angezeigten) Menüleiste dar. Auch der Textbrowser Lynx wertet seit jeher diese Informationen aus und stellt diese am Beginn der Dokumente als eine Reihe von Links dar.



Die Definition dieser inneren Zusammenhänge ist nicht auf den (unsichtbaren) Kopf eines HTML-Dokumentes beschränkt: die Link Relations können ebenso für ganz herkömmliche Links im sichtbaren Inhalt der Seite gesetzt werden (`<a href="http://www.einfach-fuer-alle.de/glossar/" rel="glossary">…`). Der Einsatz im `BODY` eines Dokumentes hat den Vorteil, dass man auf den Einsatz der Link Relations im `HEAD` verzichten kann, da diese für menschliche Nutzer weitestgehend unzugänglich bleiben und eher für die maschinelle Verarbeitung gedacht sind.

Beispiele: Metadaten für Menschen

Womit wir bei einer moderneren Form der Metadaten sind, die es zur Entstehungszeit der Verordnung in dieser Form noch nicht gab: den Microformats. Auch Microformats beschreiben Daten und reichern Dokumente semantisch an, wenn auch mit teilweise fragwürdigen Mitteln. Sie setzen jedoch nicht auf schwer zugängliche Elemente oder sogar eigene Sprachen wie das in den WCAG-Techniken empfohlene RDF, sondern sie wollen sichtbare Metadaten sein, die primär von Menschen und erst in zweiter

Linie von Maschinen genutzt werden können. Weitere Informationen zum Thema Microformats finden Sie in unserem Podcast zum Thema Microformats.

Bedingung 13.3: Orientierungshilfen (Prio. 1)

»Es sind Informationen zur allgemeinen Anordnung und Konzeption eines Internetangebots, z. B. mittels eines Inhaltsverzeichnisses oder einer Sitemap, bereitzustellen.«

Was heißt das?

Sie sollten sich nicht darauf verlassen, dass alle Besucher das Konzept hinter Hypertexten kennen und sich eine komplette Website alleine über Links im Text erschliessen, und sei diese interne Verlinkung auch noch so ausgefeilt. Viele Besucher erwarten auch im Web ein aus den Printmedien vertrautes Inhaltsverzeichnis zur schnellen Orientierung. Die mentalen Modelle von der Struktur ihrer Daten, die Besucher während der Nutzung Ihrer Website aufbauen, sind jedoch in hohem Maße individuell und von vielen Faktoren wie Erfahrung, kognitiven Fähigkeiten etc. abhängig. Um einer möglichst großen Zahl von Menschen die Nutzung eines Angebots zu ermöglichen kann es angebracht sein, verschiedene parallele Mechanismen zur Erkundung der Struktur einer Website anzubieten.

Ein Tipp: benutzen Sie lieber den Begriff »Übersicht« oder ähnliches statt wie die BITV »Sitemap«. Tests haben gezeigt, dass die Klickraten auf einen Link mit der Beschriftung »Sitemap« wesentlich geringer sind als auf den adäquaten deutschsprachigen Link »Übersicht«.

Die BITV macht hier keine Unterscheidung, ob diese Informationen tatsächlich auch nötig sind, weil ein Angebot eine gewisse Komplexität hat. Auch einfach strukturierte Websites müssen demnach »Informationen zur allgemeinen Anordnung und Konzeption« des Angebots bereitstellen, selbst wenn die hierfür notwendige Komplexitätsstufe nicht erreicht wird (oder im Ernstfall erst durch die Bereitstellung dieser Informationen erreicht wird).

Diese Bedingung sagt *nicht*, dass ein Angebot unbedingt über eine Sitemap oder ein Inhaltsverzeichnis verfügen muss – die notwendigen Informationen können auch in Textform oder bei einfach strukturierten Angeboten über eine unmittelbare Navigation zu allen Inhalten ersichtlich sein. Ein Zwang zum Anbieten eines Inhaltsverzeichnisses würde zudem alle Angebote automatisch als nicht barrierefrei deklarieren, bei denen dies aus inhaltlichen oder konzeptionellen Gründen gar nicht möglich ist. Das Prinzip des Inhaltsverzeichnis geht von einem Dokumenten-zentrischen Ansatz aus, der auf Angebote mit Anwendungscharakter nicht anwendbar ist. Aber auch in hochdynamischen Content-Seiten, in denen sich der Nutzer die Inhalte selbst zusammenstellen kann, ist ein Inhaltsverzeichnis oder eine Sitemap technisch und konzeptionell unmöglich.

Die WCAG 1.0 führen diese Richtlinie weiter aus und definieren, was mit »Informationen zur allgemeinen Anordnung und Konzeption« ebenfalls gemeint ist:

»Heben Sie Zugänglichkeits-Features in der Beschreibung einer Site hervor und erläutern Sie diese.«

Dies kann zum Beispiel eine Hilfe-Seite sein, in der der Aufbau des Angebots auf einer höheren, abstrakteren Ebene beschrieben wird und bestimmte Features erklärt werden, die für Nutzer von Hilfsmitteln von Interesse sind.

Wie können Sie das testen?

Sie müssen echte Nutzer bei der Interaktion mit ihrem Webangebot zu beobachten. Ob Sie damit ein Usability-Labor beauftragen, oder ob Sie informell ein paar Freunde und Bekannte um einen Test bitten hängt vom zur Verfügung stehenden Budget und der Größenordnung des Projektes ab. Wichtig ist nur, daß diese Tests in einer möglichst frühen Phase der Entwicklung stattfinden. Nur so können die Erkenntnisse noch bis zum Start der Seiten effektiv und kostensparend umgesetzt werden.

Auf keinen Fall sollte diese Bedingung von jemandem getestet werden, der in irgendeiner Form im Projekt involviert ist. Diesen ›Nutzern‹ sollte der Sinn und Zweck sowie der Aufbau des Angebotes bekannt sein und man darf kein brauchbares Meßergebnis aus einem solchen Test erwarten.

Bedingung 13.4: Navigation schlüssig und nachvollziehbar (Prio. 1)

»Navigationsmechanismen müssen schlüssig und nachvollziehbar eingesetzt werden.«

Was heißt das?

Auf vielen Seiten staatlicher, aber auch privater Anbieter wird der Besucher von einer Navigation begrüßt, die nur Eingeweihte verstehen können. Häufig wird die eigene Organisationsstruktur des Unternehmens oder der Behörde als Grundlage für die Struktur der Website und der Navigation genutzt. Diese Anbieter erwarten von ihren Besuchern, dass diese sich zum erfolgreichen

Navigieren mit dem Organigramm der Behörde oder des Unternehmens auseinanderzusetzen. Oder können Sie aus der abgebildeten Navigation des Bundesamt für Bauwesen und Raumordnung erkennen, wohin die Links führen?

Die wenigsten Benutzer wissen, was sich hinter Konzernbereichen, Abteilungen und angeschlossenen Referaten verbirgt. Für den Benutzer bedeutet dies, dass man sich auf jeder angesurften Website zunächst mit der inneren Struktur des Anbieters beschäftigen muss, bevor man an die Inhalte gelangen darf – ein Aufwand, der sicher allen Menschen zuviel ist.

Was meint die BIENE?

Im BIENE-Wettbewerb werden die Vorgaben dieser Bedingung gleich in einer ganzen Reihe von Prüfschritten getesetet. Dazu gehört die Prüfung:

- ob für die Navigation verständliche und treffende Begriffe verwendet werden,
- ob Navigationselemente, die gleich bezeichnet sind, mit der gleichen Funktion belegt sind,
- ob Symbole verwendet werden und ob diese die Verständlichkeit erhöhen,
- ob die Navigation auf jeder Seite konsistent und schlüssig gestaltet ist und sich auf jeder Seite einer inhaltlichen Einheit an der gleichen Stelle befindet,
- ob die Struktur innerhalb der Navigation, bei Vorhandensein von Untermenüs visualisiert wird (z. B. durch zusätzlichen Einsatz von Farbe, Schriftgröße etc.),
- ob in einer Navigation im Fußbereich der Seite keine inhaltserschließenden Navigationskategorien auftreten, die nicht auch in anderen zentralen Navigationsbereichen angeboten werden,
- ob auf einer Ebene eine dem Inhalt angemessene Zahl an Navigationskategorien angeboten wird,
- ob auf rekursive Links verzichtet wird, d. h. auf Links, die auf sich selbst zeigen. Rekursive Links sollten Sie nur dann einsetzen, wenn die Anwendung diese erfordert, z. B. zur Aktualisierung der Inhalte.

Negative Beispiele:

Auch wenn sich Pfadangaben (engl.: *Breadcrumbs*) streng genommen nicht aus den bisherigen Vorgaben zur Barrierefreiheit ableiten lassen, so werden sie doch oftmals als *›Best Practice‹* zur Anzeige der aktuellen Position innerhalb der Website empfohlen. Dabei gibt es inzwischen eine ganze Reihe von Gründen, auf Breadcrumbs zu verzichten:

a. Zum einen sind Breadcrumbs in Form von Pfadangaben kaum brauchbar, wenn es sich um dynamische Strukturen handelt und eine Zuordung in einen einzigen Pfad unmöglich ist. Sie müssten also eine Entscheidung für den Nutzer treffen, die nicht notwendigerweise dessen Sichtweise wiederspiegelt.
b. Zum anderen geben Breadcrumbs insbesondere in flachen Strukturen nur das wieder, was bereits an anderer Stelle im Dokument in der Navigation angeboten werden sollte. Man erhält also im besten Fall eine Verdopplung, im schlimmsten Fall aber sogar eine Verwirrung des Nutzers.
c. In Web-basierten Anwendungen kann die Anzeige eines klickbaren *Breadcrumb-Trails* sogar fatale Folgen haben. Diese suggerieren dem Benutzer, er könne zum Beispiel in einer Abfolge von Formularen die Anwendung in einen früheren Zustand zurückversetzen, was im Ernstfall zu ungewollten Datenverlusten führen kann.

Bedingung 13.5: Verschiedene Arten der Navigation (Prio. 2)

»Es sind Navigationsleisten bereitzustellen, um den verwendeten Navigationsmechanismus hervorzuheben und einen Zugriff darauf zu ermöglichen.« <zitat>

Was heißt das?

Aufmerksamen Lesern der vorhergehenden Bedingung 13.4 wird nicht entgangen sein, dass es kein einheitliches Regelwerk geben kann, *welche Art der Navigation* eine Website anbieten sollte oder ob nicht sogar mehrere parallele Erkundungswege zielführend sind – dazu ist das Web einfach zu komplex und die Anforderungen und Möglichkeiten zu unterschiedlich. Viele Empfehlungen beruhen auf Konventionen, die Nutzer tagtäglich neu erlernen; aber selbst diese Konventionen sind in einem hochdynamischen Medium laufend im Umbruch – sonst sähen Webseiten heute noch so aus wie 1995.

Dass jedoch *irgendeine* Form der Navigation angeboten werden *muss* ist unstrittig und genau darum geht es in dieser Bedingung. Sie können sich schon lange nicht mehr darauf verlassen, dass sämtliche Besucher Ihre Website durch den Haupteingang betreten, im Gegenteil: bei vielen Websites kommt heutzutage ein grosser Teil des Verkehrs aus Richtung Google & Co., und diese Besucher landen dann häufig direkt auf irgendwelchen Unterseiten Ihres Angebots.

Da versteht es sich von selbst, dass Ihr Design eine Erschließung der Site-Struktur von beliebigen Einstiegspunkten aus ermöglichen sollte. Oder, um den einflußreichen Architekten Rem Koolhaas zu zitieren: »design is about entrances and exits«. Zugleich sollte der Navigationsmechanismus gerade bei informationslastigen Websites offensichtlich sein und sich klar von anderen Inhalten des Angebots abheben.

Der zweite Teil der Bedingung (»einen Zugriff darauf zu ermöglichen«) lässt sich auch so interpretieren, dass die unmittelbare Erreichbarkeit der Navigation auch in den typischen Nutzungsszenarien behinderter Nutzer gewährleistet sein muss. Die kann zum Beispiel durch sog. *›Skip Links‹* zum direkten Anspringen der Navigation geschehen, die in der nächsten Bedingung ausführlicher behandelt werden. Die Navigation sollte aber nicht nur erreichbar sein, Zugriff bedeutet auch, dass sie bedienbar sein muss. Dazu gehört insbesondere die Bedienung per Tastatur, die in vielen Punkten ein guter Gradmesser für die barrierefreie Nutzung eines Angebots ist.

Bedingung 13.6: Gruppierung und Umgehung von Linkgruppen (Prio. 2)

»Inhaltlich verwandte oder zusammenhängende Hyperlinks sind zu gruppieren. Die Gruppen sind eindeutig zu benennen und müssen einen Mechanismus enthalten, der das Umgehen der Gruppe ermöglicht.«

<zitat>

Was heißt das?

Eine Bedingung, die nur noch in Teilen anwendbar ist. Im englischen Original ist dies eine der *»until user agents«*-Checkpunkte, in der deutschen Übersetzung der WCAG 1.0 heißt es an dieser Stelle: »… und ermöglichen Sie das Überspringen der Gruppe, *bis Benutzeragenten dies gestatten*«.

Wenn sie Ihre Navigation in Listenform (`OL` oder `UL`) anlegen, so werden diese Listen von modernen Screenreadern erkannt und der Nutzer kann sie mit Tastaturbefehlen überspringen. Einige Screenreader besitzen mittlerweile sogar die Fähigkeit, sich wiederholende Elemente wie eine auf allen Seiten gleiche Navigation zu ignorieren. Sie müssen also nicht jeden einzelnen Block einer Navigation mit eigenen Sprungmarken versehen, da die Einschränkung aus den WCAG als erfüllt betrachtet werden kann.

Die Möglichkeit, ganze Bereiche einer Seite zu überspringen kann jedoch für eine ganzte Reihe anderer Nutzungsszenarien hilfreich sein. Damit zum Kern der Bedingung: »Inhaltlich verwandte oder zusammenhängende Hyperlinks« lässt eigentlich nur den Schluß zu, dass es sich hier um Navigationsleisten oder ähnliches handeln muss. Dies kann sowohl die Hauptnavigation des Angebots sein als auch verschiedene Ebenen der Unter- oder Zusatznavigation.

Durch diese Bedingung sollen alternative Ausgabemedien in die Lage versetzt werden, den Benutzer bei der Ansteuerung und der Bedienung der Navigation zu unterstützen. In der Sprachausgabe und anderen nicht-grafischen Umgebungen fehlt der visuelle Kontext, mit denen sich Benutzer grafischer Browser auf einer Seite orientieren und ohne großen Aufwand zwischen den verschiedenen Inhaltsblöcken einer Seite springen können. Hierzu gehört auch die deutliche Kennzeichnung der Navigation über mehr als nur gestalterische Mittel, so zum Beispiel durch entsprechend getextete Überschriften, die den Navigationsebenen vorangestellt werden.

Für Nutzer mit motorischer Behinderung kann die Bedienung per Tastatur oder mit anderen Hilfsmitteln ausgesprochen mühselig sein, wenn Navigationsblöcke nicht zusammengefasst, sondern lose über die Seite verteilt präsentiert werden. Zur Unterstützung dieser Nutzer ist es sinnvoll, Navigationsblöcke über einen Verweis innerhalb der Seite direkt ereichbar zu machen. Auch der umgekehrte Fall, das Überspringen einer vorgeschalteten Navigation zum direkten Erreichen des eigentlichen Inhalts ist eine einfache Hilfe, die Sie Ihren Nutzern anbieten sollten. Falls Ihre Navigation im Quelltext vor dem eigentlichen Inhalt der Seiten steht ist diese Technik auch eine Hilfe für Screenreader-Nutzer, die so verhindern können, daß ihnen die Navigation immer wieder vorgelesen wird, bevor sie zum Inhalt der Seite kommen.

Tipp: wenn sie Sprungmarken einsetzen, dann sollte der Text das Ziel des Links beschreiben und nicht den Bereich, den man überspringt. Eine Formulierung wie »Navigation überspringen« sagt dem Nutzer nicht, wo es hingeht; angebrachter wären hier zwei Skip-Links »Zum Inhalt« und »Zur Navigation«.

Dies geschieht in HTML über lokale Anker (`<a href="#inhalt>`), die auf einen Punkt einer Seite mit einer bestimmten ID (`<div id="navigation">`) verweisen. Dies sind genau die IDs, die man zur Identifizierung von Blöcken einsetzt, um per CSS Stile zuweisen zu können. Der leichte Einbau einer solchen Navigation innerhalb einer Seite ist also fast schon ein Abfallprodukt der CSS-Formatierung und entsprechend einfach umzusetzen. Daher sollten Sie diese Vorgabe abweichend von der BITV durchaus mit Priorität 1 behandeln.

Negative Beispiele:

Bei einem unkritischen und übermäßigen Einsatz von Skip-Links kann es jedoch passieren, dass Sie mehr Barrieren auf- als abbauen. So können dauerhaft sichtbare Skip-Links bei der rein visuellen Orientierung den Nutzer verwirren, wenn sie zwar immer angezeigt werden, aber keine offensichtliche Aktion auslösen, da das Linkziel sich wie üblich schon im sichtbaren Bereich befindet.

Auch die Methode, Skip-Links zwar einzubauen, sie dann aber dauerhaft per CSS zu verstecken ist keine optimale Lösung. Gerade für Tastaturnutzer ist es ausgesprochen verwirrend, wenn man erst fünfmal auf Tab drücken muss, um festzustellen,

dass es eine ganze Batterie von versteckten Skip-Links gibt, deren Funktion man nur durch Entziffern der URL in der Statuszeile des Browsers erraten kann.

Positive Beispiele:

Eine elegante Lösung dieser Probleme ist auf den Seiten der Stadt Wien zu finden: die Skip-Links sind zunächst per CSS aus dem sichtbaren Bereich der Seite geschoben und werden in grafischen Desktop-Browsern erst bei Tastaturbedienung sichtbar. Sobald der Nutzer nach dem Laden der Seite die Tab-Taste drückt (und sich somit als Tastaturnutzer zu erkennen gibt) werden die Skip-Links (Zur Navigation springen und Zum Inhalt springen) per CSS in den sichtbaren Bereich der Seite geschoben und sind damit sowohl optisch wahrnehmbar als auch per Tastatur (Enter) auszulösen.

Wie können Sie das testen?

Um die angesprochenen unsichtbaren Skip-Links zu finden müssen sie in Ihrem Browser CSS abschalten. Am einfachsten geht dies wie üblich mit den Toolbars für Webentwickler, die über entsprechende Funktionen verfügen. Alternativ können Sie auch einen Sprachbrowser oder einen Screenreader verwenden und Ihren Monitor ausschalten. Versuchen Sie, sich nur per Sprachausgabe und Tastaturbedienung auf Ihren Seiten zu orientieren. Gelingt Ihnen dies in einer angemessenen Zeit (und ohne den Monitor wieder anzuschalten!), so haben Sie den Test bestanden.

Warnhinweis:

Das Leben des Webentwicklers wäre geradezu langweilig, wenn es nicht zu allem den passenden Internet Explorer-Bug gäbe. So auch hier: Skip Links funktionieren im IE nur, wenn das Linkziel in einem Block-Element mit definierter Breite liegt. Mehr zu dieser Problematik im Artikel »On having layout — the concept of hasLayout in IE/Win« von Ingo Chao.

Bedingung 13.7: Verschiedene Suchmethoden (Prio. 2)

»Soweit Suchfunktionen angeboten werden, sind der Nutzerin/ dem Nutzer verschiedene Arten der Suche bereitzustellen.« <zitat>

Was heißt das?

Sie können nicht davon ausgehen, dass Ihre Besucher Boolesche Operatoren beherrschen, geschweige denn von deren Existenz wissen. Das leider immer noch häufig anzutreffende Schild »Fahrradfahren und Ballspielen verboten« ist ein Beleg hierfür: für Mathematiker oder Informatiker bedeutet dies, daß man während des Fahrradfahrens nicht auch noch gleichzeitig Ballspielen darf, das Eine *ohne* das Andere ist aber nach wie vor erlaubt.

Daher sollten Sie für Ihre Besucher verschiedene Schwierigkeitsgrade der Suche anbieten, zum Beispiel einmal eine simple Volltextsuche und zusätzlich eine Komfortsuche mit erweiterten Optionen und der Möglichkeit, logische Verknüpfungen zu benutzen. Wichtig ist auch eine Fehlertoleranz bei der Suche: gerade für Menschen mit kognitiven Behinderungen ist es oftmals nicht erkennbar, dass sie einen formellen Fehler bei der Eingabe von Suchbegriffen gemacht haben. Viele Such–Skripte sind mittlerweile in der Lage, Rechtschreibfehler in der Eingabe zu erkennen und ähnlich geschriebene oder phonetisch verwandte Suchtreffer mit anzuzeigen. Kombiniert mit den Möglichkeiten von AJAX lassen sich bereits während der Eingabe des Suchbegriffs Korrekturvorschläge oder Vorhersagen über zu erwartende Suchtreffer anzeigen.

Was meint die BIENE?

Zur Umsetzung einer barrierefreien Suche gehört nicht nur die Suchfunktion als solche, auch die Suchergebnisse müssen für behinderte Nutzer zugänglich und nutzbar sein. Daher werden im BIENE-Prüfverfahren eine ganze Reihe weitergehender Kriterien gestestet. Neben der Überprüfung, ob verschiedene Suchoptionen angeboten werden oder ob eine fehlerfreundliche Suche angeboten wird sind dies zum Beispiel:

- ob die Präsentation der Suchergebnisse einstellbar ist (Art und Umfang der Suchergebnisse, z. B. Anzahl in der Trefferliste),
- ob in der Trefferliste Beschreibungen zu den Treffern vorhanden sind, wenn die Treffer selbst nicht selbsterklärend sind,
- ob die Nutzerin / der Nutzer über ein erfolgloses Suchergebnis informiert wird und angemessene Hilfestellung erhält.

Bedingung 13.8: Differenzierung unterschiedlicher Inhaltsblöcke (Prio. 2)

»Es sind aussagekräftige Informationen am Anfang von inhaltlich zusammenhängenden Informationsblöcken (z. B. Absätzen, Listen) bereitzustellen, die eine Differenzierung ermöglichen.« <zitat>

Was heißt das?

In anderen Bedingungen wie 3.5 ist geregelt, dass bereits vorhandene Überschriften und andere strukturierende Elemente als solche auszuzeichnen sind. Hier geht es nun darum, solche Inhalte überhaupt erst einmal zu konzipieren und entsprechend zu texten. Angesprochen sind hiervon also Konzepter und Redakteure, die eine Strukturierung vorsehen müssen, und nicht, wie in

den anderen Bedingungen, die Designer und Programmierer, die diese Struktur gestalten und umsetzen sollen. Vereinfacht gesagt: es geht hier nicht um technische Umsetzung, sondern um die Inhalte *ausserhalb* der spitzen und geschweiften Klammern.

Screenreader, Lupenprogramme und andere Hilfsmittel können mit einer sauberen Struktur den Aufbau einer Seite besser wiedergeben und es dem Benutzer ermöglichen, sich in dieser Struktur zu bewegen. Für sehende Nutzer ist die Hinterlegung und deutliche Abgrenzugen von logischen Einheiten auch ein wichtiges Mittel zum Verständnis und zur effizienten Nutzung eines Angebots. Immer mehr grafische Browser können Seiten auf diese Art zusammenfassen oder zusätzliche Navigationshilfen bereitstellen, die weit über das bloße Antabben von Links hinausgehen – ein gerade für Menschen mit bestimmten motorischen Behinderungen wichtiges Feature.

Weitere in den WCAG 1.0 beschriebene Techniken sind durchaus kritisch zu betrachten: so ist das in wissenschaftlichen Texten gängige sog. *›Front-Loading‹* durchaus angebracht, wenn es sich um entsprechende Dokumente handelt, bei denen diese inhaltliche Strukturierung gebräuchlich ist (wiss.: Synopse). Das Web ist jedoch mittlerweile mehr als nur eine Ansammlung wissenschaftlicher Texte, und diese Art, Inhalte zu strukturieren lässt sich beileibe nicht allen Inhalten und Funktionen im Web ›überstülpen‹.

Wie können Sie das testen?

Da es hier nicht um die technische Umsetzung geht, sondern um die Bewertung der Inhalte und ihrer Struktur, lässt sich diese Bedingung nur durch intensive Beschäftigung mit den Inhalten testen. Wenn der Charakter ihrer Dokumente dies zulassen, dann sollten Sie im Browserfenster prüfen, ob die wichtigsten Aussagen (*wer, was, wann, warum, wie*) am Anfang des Dokuments oder der logischen Einheit bzw. in einer Zusammenfassung stehen z. B. als Vorspann, Anreisser.

Bedingung 13.9: Zusammenfassung von mehrteiligen Dokumenten (Prio. 2)

»Soweit inhaltlich zusammenhängende Dokumente getrennt angeboten werden, sind Zusammenstellungen dieser Dokumente bereitzustellen.« <zitat>

Was heißt das?

Kurzfassung: das wissen wir selbst nicht.

Langfassung: die Dokumentation zu dieser Bedingung bzw. zum Original in den Web Content Accessibility Guidelines ist ausgesprochen dürftig. Die Techniques for WCAG 1.0 zeigen Beispiele, die eher in anderen Bedingungen bzw. Checkpunkten anzusiedeln sind; so zum Beispiel die bereits in 13.2 behandelten *Link Relations* und ähnliche Techniken, oder die Bereitstellung von Navigationsmechanismen und deren schlüssige und nachvollziehbare Verwendung, die bereits in 13.4 bzw. 13.5 ausreichend geregelt sind.

Bei anderen Beispielen der WCAG-Techniken (Bereitstellung eines alternativen Angebots zum Download in Form von zip-, tar-, gzip- und stuffit-Archiven) will uns beim besten Willen nicht einleuchten, wo hier der Zusammenhang zum Thema ›Barrierefreies Webdesign‹ besteht. Diese Bedingung ist also bestenfalls ein Doppler und somit als überflüssig zu betrachten.

Bedingung 13.10: Umgehung von ASCII-Zeichnungen (Prio. 2)

»Es sind Mechanismen zum Umgehen von ASCII–Zeichnungen bereitzustellen.« <zitat>

Was heißt das?

Was mit dieser Bedingung tatsächlich gemeint ist wird erst deutlich, wenn man sie mit dem Original in den WCAG 1.0 vergleicht. Dort steht:

Ermöglichen Sie das Überspringen von ***mehrzeiligen*** ASCII-Zeichnungen. <zitat>

(Anm.: Hervorhebung hinzugefügt)

Hintergrund: in den Anfangstagen des Web konnten noch nicht alle Browser Tabellen darstellen. Daher wurden gelegentlich tabellarische Daten mit Mitteln aufbereitet, die mit viel gutem Willen eine Tabelle erahnen liessen, wie dieses Beispiel in den WCAG zeigt. Da es sich hierbei jedoch nicht um eine echte Tabelle mit nachvollziehbaren Inhalten und Zusammenhängen handelt, sondern nur um die willkürliche Anordnung von Zeichen wie __, I, * und @, stellten solche ›Tabellen‹ damals eine echte Barriere in der Nutzung dar. Daher die Einschränkung auf solche »**mehrzeiligen** ASCII-Zeichnungen«, die in der BITV leider fehlt.

Der Nachweis, dass eine simple ASCII-Zeichnung wie ein Smilie :-) tatsächlich eine Barriere darstellt, die in der in den WCAG-Techniken vorgeschlagenen Art und Weise zu beseitigen ist, wird wohl kaum erbracht werden können:

```
<P>
<a href="#post-art">skip over ASCII art</a>
<!-- ASCII art goes here -->
<a name="post-art">caption for ASCII art</a>
```

Die Bedingung fällt damit eher in die Kategorie ›historische Kuriositäten‹ und ist als nicht mehr anwendbar zu betrachten.

Anforderung 14: Förderung des allgemeinen Verständnisses

»Das allgemeine Verständnis der angebotenen Inhalte ist durch angemessene Maßnahmen zu fördern.« <zitat>

Was heißt das?

In der letzten Anforderung der BITV geht es nochmal um Barrierefreiheit auf inhaltlicher Ebene. Selbst wenn eine Website alle technischen Tests besteht, der Code validiert und die gängigen Prüfprogramme auf keine groben Schnitzer hinweisen: die Website ist damit noch nicht für alle Besucher tatsächlich auch benutzbar.

Während die übrigen Anforderungen der BITV größtenteils technische Vorgaben oder Mischformen von technischen, gestalterischen und konzeptionellen Bedingungen enthalten, handelt es sich hier um die einzige Anforderung, die ausschließlich Inhalte und deren Gestaltung regelt. Dabei ist die Bedingung 14.1 im Original, den WCAG 1.0, sogar unter Priorität 1 abgelegt; im Gegensatz zur Verpflichtung zu validem Code (s. 3.2), die lediglich bei Priorität 2 angesiedelt ist. Dies verdeutlicht, dass den Fähigkeiten der Nutzer entsprechende Inhalte wesentlich wichtiger für die barrierefreie Nutzung eines Angebots sind als das eine oder andere uncodierte Ampersand.

Grenzbereiche des Machbaren

Hier beginnt eine Grauzone: die inhärente nicht-lineare Organisationsform des Web und die Menge und Komplexität der Inhalte stellt viele Menschen mit kognitiven Behinderungen vor unüberwindbare Probleme. Diese können nicht mit technischen Mitteln und nur begrenzt mit gestalterischen und inhaltlichen Mitteln gelöst werden. Trotz einer ganzen Reihe von Erkenntnissen zur Lesetypografie oder zur Organisation von Inhalten für Menschen mit kognitiven Behinderungen besteht weiterhin die Schwierigkeit, dass aus diesen Erkenntnissen keine allgemeinen Handlungsanweisungen für Autoren oder Regeln für Verordnungen wie die BITV und Richtlinien wie die WCAG ableitbar sind.

Bereits in den WCAG 1.0 führte dies zur Abhandlung dieser Checkpunkte auf den hinteren Rängen. In den kommenden WCAG 2.0 vergrössert sich diese Lücke noch weiter: eine der Grundvoraussetzungen für die Überarbeitung der Richtlinien ist, dass alle Vorgaben anhand von definierten Erfolgskriterien testbar sein müssen. Der für die übrigen Anforderungen der BITV durchaus zu vertretende Ansatz der Evaluation durch entsprechend geschulte Tester (ggf. mit Unterstützung durch geeignete Prüfprogramme) ist jedoch hier nicht zu gebrauchen.

Computer-gestützte Verfahren wie z.B. Lesbarkeitsstatistiken können immer nur erste Hinweise auf mögliche Probleme geben. Ein verlässliches, nachvollziehbares und vor allem wiederholbares Testergebnis kann man hier *nicht erwarten*. Im Gegensatz zu den technischen Bedingungen ist immer auch der menschliche Faktor zu berücksichtigen.

Widersprüche der Barrierefreiheit

Die praktische Erfahrung im Rahmen des BIENE-Prüfverfahrens hat Grenzen des Ansatzes *»ein Webangebot für alle Nutzer«* aufgezeigt. Ab dieser Grenze erscheint eine weitere »Vereinfachung« der Inhalte wenig sinnvoll, weil das Angebot damit für andere Nutzergruppen nicht mehr zu gebrauchen wäre. Über die Übersetzungsleistung in eine einfachere Sprache hinaus ist eine völlig andere Struktur und Anpassung der Abläufe erforderlich, die teilweise den Grundprinzipien des Mediums Internet widerspricht.

Hinzu kommt, dass Vorgaben, die für manche Behinderungsformen Unterstützung bieten, für Menschen mit anderen Behinderungen eine echte Barriere darstellen können. Ein Beispiel: die Vorgaben zu möglichst hohen Kontrasten. Die WCAG 2.0 setzen hierbei eine *Unter*grenze fest, ab der ein Farb-oder Helligkeitskontrast als problematisch gelten kann. Nach den aktuellen Erfolgskriterien erhält man die Höchstpunktzahl für den maximalen Kontrast, im Ernstfall also schwarzer Text auf weissem Hintergrund. Eine *Ober*grenze für Kontraste ist jedoch nicht definiert, obwohl zu starke Kontraste für viele Menschen mit spezifischen Lernbehinderungen eine echte Hürde darstellen. Manche Hilfsmittel wie zum Beispiel Lupenprogramme für Sehbehinderte haben Probleme mit zu starken Kontrasten.

Der Anbieter steht nun vor dem Problem der Abwägung zwischen der möglichen Eigenverantwortung des Nutzers und dem Willen oder der Verpflichtung, möglichst viele Nutzergruppen zu bedienen.

Bedingung 14.1: Einfache Sprache (Prio. 1)

Für jegliche Inhalte ist die klarste und einfachste Sprache zu verwenden, die angemessen ist.« <zitat>

Was heißt das?

Fachsprache ist in vielen Bereichen durchaus gerechtfertigt. Nur sollten Inhalte, die für die Erledigung von Bedürfnissen des täglichen Lebens gedacht sind, auch so getextet sein, dass sie für die Benutzer keine unnötigen Hürden aufbauen. Kein Anbieter kann davon ausgehen, dass die Besucher seiner Website die in der Organisation des Anbieters gängigen Fachtermini beherrschen. Viele Texte, die sich an ein großes Publikum richten, werden trotzdem in einer Sprache verfasst, die fast schon ein

abgeschlossenes Studium im jeweiligen Fachbereich voraussetzt. Dazu gehören nicht nur Behördengänge, sondern auch der Informationsbedarf der Internet-Nutzer oder der immer mehr an Bedeutung zunehmende Bereich des elektronischen Handels (engl.: *e-Commerce*).

Sofern es sich nicht um Fachtexte handelt, sollten Sie weitestgehend auf Anglizismen und andere fremdsprachliche Formulierungen verzichten. Viele Vorgaben und Ideen, die sich in den Richtlinien zur Erstellung von Texten in leichter Sprache (PDF, z.Zt. 404) für Menschen mit kognitiven Behinderungen finden, lassen sich auf auch Texte anwenden, die sich an ein großes Publikum mit unterschiedlichsten Sprachkompetenzen richten:

<zitat>

- Verwendung von einfacher und unkomplizierter Sprache
- Keine abstrakten Begriffe
- Kurze Worte aus der Alltagssprache
- Persönliche Ansprache
- Praktische Beispiele / Bilder
- Kurze Sätze
- Keine Abkürzungen, Fremdwörter, Initialen
- Gleiche Begriffe für eine Sache
- Positive Sprache

Diese Regeln sind der Mitschrift eines Beitrags von Cordula Edler zum Kongress ›Mehr Wert für Alle‹ in Trier (2004) entnommen.

Wie können Sie das testen?

Die Bedingung 14.1 ist eine der Vorgaben, die nicht maschinell zu überprüfen sind. Hier hilft tatsächlich nur, die Texte einer Website von potentiellen Nutzern des Angebots auf Verständlichkeit überprüfen zu lassen. Sie können schon an Hand einiger simpler Tests wie »Melden Sie Ihren Hund um«, »Bestellen Sie ein Produkt X« oder »Versuchen Sie Informationen über XYZ zu finden« mit wenigen Testpersonen die Verständlichkeit Ihres Angebots überprüfen lassen.

Diese Tests sollten jedoch *nie* von Personen durchgeführt werden, die in irgendeiner Weise mit der Entwicklung des Angebots oder der anbietenden Institution in Verbindung stehen. Nur so können Formulierungen gefunden werden, die dem ›Insider‹ vielleicht geläufig sind, für alle anderen aber ein Buch mit sieben Siegeln darstellen. Je früher diese Tests in einem Projekt stattfinden, desto größer ist die Wahrscheinlichkeit, dass Sie die gewonnenen Erkenntnisse noch kostenneutral umsetzen können.

Angemessenheit:

Seit der Veröffentlichung der WCAG 1.0 streiten Experten, ab wann der Punkt der Angemessenheit erreicht ist, oder ob dieser Zustand überhaupt erreicht werden *kann* (und ob somit WCAG 1.0 Level A bzw. BITV Priorität 1 überhaupt erfüllbar sind). Dazu gehört auch die Diskussion, ob die Forderung nach einer angemessenen Sprache auch das Angebot von Deutscher Gebärdensprache (DGS) beinhaltet, wenn das Angebot auch von gehörlosen Menschen genutzt werden soll. Die zurzeit in Entwicklung befindlichen WCAG der Version 2.0 fordern hier zumindest für multimediale Inhalte die Übersetzung in Gebärdensprache, wenn auch erst ab Level AAA.

Positive Beispiele:

Ein gelungenes Beispiel für die ›Rück-Übersetzung‹ komplexer Inhalte in eine allgemeinverständliche Sprache ist der sogenannte Virtuelle Beipackzettel des Pharmaunternehmens Pfizer. Die Packungsbeilagen in Medikamenten sollen einerseits juristisch wasserdicht sein, den ausgebildeten Mediziner oder Apotheker informieren und andererseits auch für den Endverbraucher verständlich sein. In der Regel fallen die Formulierungen jedoch zugunsten der Fachleute aus, was für Verbraucher im Ernstfall zu gesundheitsbedrohlichen oder sogar lebensgefährlichen Situationen führen kann, wenn entsprechende Warnhinweise nicht verstanden werden.

Die 2006 mit einer BIENE ausgezeichnete Website von Pfizer ist hier gleich in mehrfacher Hinsicht vorbildlich: die für die Mehrzahl der Patienten nicht mehr verständlichen Packungsbeilagen wurden in eine verständliche Sprache übersetzt; zudem werden die Fragen und Antworten in kleinen Filmsequenzen in Gebärdensprache gezeigt.

Mögliche Alternativen:

Selbst wenn sich Ihr Angebot vornehmlich an ein Fachpublikum richtet, so müssen sie doch immer mit der Nutzung durch Nicht-Fachleute rechnen. Für solche Fälle ein paralleles Angebot mit vereinfachten Texten zu erstellen und dauerhaft zu pflegen kann unter Umständen den Etat Ihres Web-Projektes sprengen. Um Besuchern, die nicht mit der Thematik Ihrer Site vertraut sind, eine Nutzung zu ermöglichen können und sollten Sie ein Glossar anbieten, das die häufigsten Fachbegriffe, Anglizismen und ähnliches auf eine verständliche Art erklärt.

Bedingung 14.2: Grafische Gestaltung (Prio. 2)

Text ist mit graphischen oder Audio–Präsentationen zu ergänzen, sofern dies das Verständnis der angebotenen Information fördert.« <zitat>

Was heißt das?

Die grafische Gestaltung ist ein wichtiger Bestandteil des gesamten Web-Angebotes und dient nicht nur der Selbstdarstellung des Anbieters. Sowohl der Besucher als auch der Anbieter haben ein Recht auf Gestaltung, aber Anbieter haben sogar die besondere Verantwortung und Pflicht, gut gestaltete und benutzbare Seiten zu veröffentlichen. Die BITV schließt hierbei Sonderwege wie reine Nur-Text-Versionen ausdrücklich aus, da diese nicht barrierefrei im Sinne der Verordnung für die Gesamtheit der Besucher sein können.

Eine gelungene grafische Gestaltung fördert jedoch nicht nur, wie in der BITV formuliert, »das Verständnis der angebotenen Information«, sondern dient der gesamten Wahrnehmung, Orientierung und Bedienung eines Angebots. Besonders für Menschen mit einer unterdurchschnittlichen Sprachkompetenz sind grafische Hilfen oder multimediale, interaktive Ergänzungen oftmals ein entscheidender Faktor für die erfolgreiche Bewältigung der selbstgestellten Aufgabe, für deren Erledigung sie auf Ihre Seiten gekommen sind. Hierunter fallen im Sinne der Verordnung vor allem Menschen mit Lernbehinderungen, Lese-Rechtschreib-Schwächen oder kognitiven Behinderungen; gerade im Bereich der Wissensvermittlung lassen sich durch alternative grafische oder multimediale Inhalte aber auch unterschiedliche Lernstile bei allen Nutzern abdecken.

Die Textlastigkeits des Mediums Web wird durch die vorhergehenden Vorgaben der BITV zur Bereitstellung von Textalternativen für sämtliche nicht-Text-Inhalte noch weiter verstärkt. Hierdurch können für Menschen mit kognitiven Behinderungen weitere Barrieren aufgebaut werden, wenn sie mit zu großen Textmengen konfrontiert werden.

Wie können Sie das testen?

Eigentlich gar nicht, zumindest nicht als ausschließlich mit der Produktion befasster Webdesigner: die Vorkehrungen zur Erfüllung der Bedingungen müssen bereits in der Konzeptions- und Designphase eines Angebotes getroffen werden.

Wie alle Bedingungen der Anforderung 14 ist auch diese nicht wirklich testbar, ohne ein Usability-Labor zu bemühen. Die Schwierigkeit besteht hierbei, dass sich diese Bedingung nicht wie die eher technisch orientierten Bedingungen nach einfachen Regeln testen lässt und die Tests kaum ein eindeutiges Ja-/Nein-Ergebnis bringen werden. Eine klare Grenze zwischen »verstehen« und »nicht verstehen« gibt es nicht und ist damit im Rahmen eines Prüfverfahrens nicht wirklich objektiv meßbar.

Hieraus ergeben sich Schwierigkeiten in der Abgrenzung, ab wann die verwendete Textmenge für die Zielgruppen eines Angebotes zu viel wird und alternative grafische oder multimediale Präsentationen nötig sind. Bei einer Website für eine Selbsthilfegruppe von Menschen mit kognitiven Behinderungen oder einem Angebot für funktionale Analphabeten ist dies noch relativ leicht zu definieren: hier **müssen** Sie auf die ganz speziellen Bedürfnisse dieser Nutzer reagieren; bei allgemeiner gefassten Angeboten hängt dies oftmals vom Problembewußtsein und Willen des Anbieters und den zur Verfügung stehenden Ressourcen ab.

Da kognitive Behinderungen jedoch nicht generell mit niedriger Intelligenz gleichzusetzen sind, kann man diese Nutzer als Anbieter von komplexen Inhalten, e-Learning-Plattformen und ähnlichem nicht einfach aus der Zielgruppe herausdefinieren. Dass die speziellen Bedürfnisse dieser Nutzer in der BITV mit niedriger Priorität behandelt werden ist zwar eine direkte Folge der Einsortierung bei Level AAA der WCAG 1.0, dies ist aber aus Sicht der Betroffenen weiterhin unzureichend. *Sämtliche* Anforderungen an die Accessibility muten zunächst kompliziert an, bis man sich als Anbieter mit den Techniken zum Abbau von Barrieren vertraut macht und diese umsetzt.

Bedingung 14.3: Durchgängiger Präsentationsstil (Prio. 2)

»Der gewählte Präsentationsstil ist durchgängig beizubehalten.« <zitat>

Was heißt das?

Eine gewisse Konsistenz hilft Nutzern beim Aufbau eines mentalen Modells der Architektur eines Angebots und minimiert den Lernaufwand für Nutzer mit Lernbehinderungen. Durch die Vorhersagbarkeit der Platzierung, Reihenfolge und Optik von Elementen wird die Orientierung und Bedienung eines Angebots vereinfacht. (Nota bene: weitergehende technische und gestalterische Vorgaben an Webinhalte werden in vielen der vorhergehenden Anforderungen behandelt; hier geht es ausschließlich um die Konsistenz des Aufbaus und der Präsentation. Hierzu gehört aber auch, dass eine Navigation auf allen Seiten eines Angebots aussehen sollte wie eine Navigation.)

Für stark sehbehinderte Nutzer, die einen verbliebenen Sehrest weiterhin zur Orientierung nutzen ist diese Bedingung von essentieller Bedeutung. Bei den teils enormen Vergrößerungsfaktoren, mit denen Lupenprogramme betrieben werden ist die

Effizienz, mit der sich immer wiederkehrende Seitenelemente wie Inhalte, Navigationsleisten etc. erreichen und identifizieren lassen, direkt von deren konsistenter Platzierung und Gestaltung abhängig.

Gleiches gilt für Nutzer von Screenreadern: eine konsistente Anordnung ermöglicht es dem Nutzer oder dem eingesetzten Hilfsmittel, Bereiche von Interesse zu identifizieren und uninteressante Bereiche zu überspringen. Screenreader geben den Inhalt als eine lineare Abfolge von Text wieder, mit dem sich räumliche Zusammenhänge nicht ohne weiteres erschließen lassen. Da diese sich an der Reihenfolge der Seiteninhalte im Quelltext orientieren sollten Sie einmal eingeführte relative Abfolgen nicht nur in der Gestaltung, sondern auch im Quelltext beibehalten.

Menschen mit motorischer Behinderung können durch diese Bedingung Strategien entwickeln, um Ihre Website mit möglichst wenigen Tastendrücken zu erkunden, wenn Ihre Inhalte auf allen Seiten in der gleichen Art und Weise angeordnet und erreichbar sind.

Grenzen bei der Bewertung

Problematisch ist diese Bedingung durch die fehlende genauere Defintion der Begriffe »durchgängig« und »beizubehalten« in den WCAG 1.0. Bei den in den Techniken aufgeführten Beispielen will uns beim besten Willen nicht einleuchten, was eine minimale Anzahl an externen Style Sheets mit der Barrierefreiheit von Webinhalten für Menschen mit Behinderung zu tun hat. Hier fehlt im Umkehrschluß der Beweis, dass mehr als zum Beispiel zwei oder drei externe Style Sheets für Menschen mit Behinderungen eine wie auch immer geartete Barriere darstellen.

Auch aus der konzeptionellen oder gestalterischen Sicht des Anbieters ist diese Bedingung problematisch: Bei einfachen Informationsangeboten oder bei Shopping-Seiten mit wenigen, ähnlichen Produkten ergibt sich eine Erfüllung schon fast zwangsläufig alleine aus ökonomischen Gründen; in einem komplexen eCommerce-Angebot sollten die Bestellfunktionen überall gleich gestaltet sein, auch wenn die eigentliche Waren in unterschiedlichen Warengruppen durchaus auch unterschiedlich präsentiert werden können. Bei einem Markenartikler mit vielen unterschiedlichen Produktlinien wird dieser Punkt der BITV jedoch *nicht durchsetzbar* sein. Eine zu strikte Auslegung dieser Bedingung würde einen Mischkonzern dazu zwingen, Informationen zu Waschmitteln auf die gleiche Art und Weise zu präsentieren wie die des Geschäftsbereichs Schokoriegel.