
13 Technischer Unterbau

Valide Dokumente zeigen, dass Webinhalte sorgfältig und zeitgemäß aufbereitet wurden. Valide Dokumente sind damit ein Zeichen für Standardkonformität und fördern die Kompatibilität. Zudem unterstützen Hilfsmittel eine Auszeichnungssprache besser, wenn diese entsprechend ihrer Spezifikation verwendet wird. Valider Code ist zugleich ein Qualitätsmerkmal und erleichtert die Arbeit nach Webstandards.

Valides HTML bedeutet aber nicht, dass der Inhalt barrierefrei ist, sondern ist sozusagen das i-Tüpfelchen der Barrierefreiheit, wenn inhaltliche, strukturelle und interaktive Komponente barrierefrei umgesetzt sind.

Validität ist die Übereinstimmung des Quellcodes mit der formalen Grammatik einer Sprache wie z.B. HTML und zeigt, dass die jeweilige Auszeichnungssprache auf gültige (valide) Weise verwendet wird. Ein weiterer Aspekt ist die Wohlgeformtheit, die beispielsweise für XML-basierte Dokumente berücksichtigt werden muss.

Letztlich können alle Aspekte, die den technischen Unterbau eines Dokuments und gleichzeitig die Zugänglichkeit und Nutzbarkeit betreffen, unter dem Gesichtspunkt der Validität gesehen werden. Aus dem Blickwinkel der Barrierefreiheit geht es u.a. um den sinnvollen Einsatz von HTML-Elementen und -Attributen sowie die logischen Beziehungen zwischen einzelnen, zusammenhängenden Inhalten.

13.1 Validität

Die Einhaltung aller Webstandards ist für eine Konformität mit den WCAG20 nicht zwingend erforderlich, gilt aber als »Best Practice«. Nicht immer kann aber genau nach Webstandards gearbeitet werden, wenn z.B. in den CSS zuweilen Hacks für eine browserübergreifend gleiche Darstellung nötig werden.¹ Auch wenn eine Validierung gegen alle technischen Standards erfolgt, kann weder eine Aussage über die grundsätzliche Zugänglichkeit noch die Nutzbarkeit einer Webseite getroffen werden. Sie ist jedoch Voraussetzung für die technische



1. Heller, St., CSS Hack, URL: <http://www.css-hack.de/> (Abruf 10.12.2010).

Aufbereitung von Dokumenten und damit für die Zugänglichkeit der Inhalte in verschiedenen Anwendungen.

Auf der technischen Ebene ist – trotz der nur eingeschränkten Aussagefähigkeit zur Barrierefreiheit – ein valides Dokument ein Qualitätsmerkmal. Der Einsatz von HTML-Validatoren zur Prüfung des technischen Unterbaus kann Fehler aufdecken, die die technische Zugänglichkeit beeinflussen.

Für die Barrierefreiheit selbst gibt es keine Validatoren. Obwohl die WCAG20 technisch überprüfbare Anforderungen enthalten, muss Barrierefreiheit in Abhängigkeit sowohl von den Inhalten als auch ihrem Kontext bewertet werden. Einige spezialisierte Tools erleichtern jedoch die Bewertung.

13.1.1 Kompatibilität sichert die Zugänglichkeit



Die Zugänglichkeit der Webinhalte ist nicht nur vom Browser abhängig. Hilfsmittel wie Screenreader, Vergrößerungssysteme, Spracherkennung oder Maus- und Tastaturergänzungen setzen den korrekten Einsatz verschiedener Techniken entweder voraus oder funktionieren dann besser. Die Schnittstelle zwischen Hilfsmittel und Dokument kann der Browser selbst oder ein Teil des Betriebssystems sein, wie etwa die MSAA-Schnittstelle in Windows-Systemen.² Je standardkonformer Dokumente aufbereitet sind, umso besser können die Schnittstellen zwischen Inhalt und Nutzer »gefüttert« werden.

Keine Software ist perfekt und nicht alle Spezifikationen zur Barrierefreiheit werden von den Anwendungen so aufbereitet oder unterstützt, wie es erforderlich wäre. Browser und Hilfsmittel entwickeln sich aber ebenso wie andere Software-Produkte weiter. Wenn Sie heute Webstandards berücksichtigen, dann ist auch die Bedienung der Schnittstellen in der Zukunft sichergestellt. Werden hingegen Techniken eingesetzt, die zwar heute funktionieren, aber nicht standardkonform sind, entstehen zwei Probleme:

1. Eine nicht standardkonforme Technik funktioniert möglicherweise in zukünftiger Software nicht oder nicht mehr so gut. So verfügen die meisten Browser über Korrektur-Algorithmen beim Rendering von HTML. Fehlerhafter Quellcode kann dadurch meist trotzdem angezeigt werden. Wenn aber das HTML nicht ordentlich in eine Datenstruktur umgewandelt werden kann, kann es sein, dass eine falsche oder gar keine Darstellung angezeigt wird. Eine korrekte Schreibweise ist robust und gewährleistet eine zuverlässige Interpretation durch Browser und weitere Zugangssoftware.
2. Nur der richtige Einsatz von Webstandards stellt die Unterstützung in Anwendungen langfristig sicher. Die Förderung der Zugänglichkeit ist selbstverständlich nicht nur Aufgabe der Webanbieter, sondern auch eine Aufgabe der Browserhersteller und der Entwickler von Redaktionssystemen. Die Zugänglichkeit muss durch den erzeugten Code in Richtung Barriere-

2. Christmann, G.; Hellbusch, J. E., Microsoft Active Accessibility, URL: <http://www.barrierefreies-webdesign.de/knowhow/msaa/> (Abruf 5.7.2010).

freiheit optimiert werden, da es sonst für die Software-Hersteller wenig Anreiz gibt, die Techniken an die Schnittstellen durchzureichen.

13.1.2 HTML-Validierung

Ein gültiger Code ist Voraussetzung einer korrekten Umsetzung nach Webstandards und wird durch die HTML-Validierung festgestellt.

Damit ein Webdokument mit einem Validator geprüft werden kann, muss zu Beginn des Dokuments eine Dokumententypdefinition (DTD) angegeben werden (vgl. Abschnitt 4.1.1 ab S. 118). Die DTD gibt dem Validator die Information über die anzuwendende Spezifikation und definiert zugleich den zulässigen (gültigen) Aufbau des Dokuments sowie die Verwendung von Elementen und Attributen. Durch die DTD können außerdem die Inhalte durch die Browser standardkonform aufbereitet werden. DTDs gibt es außer für HTML für zahlreiche weitere Sprachen.³



13.1.2.1 Validatoren für HTML-Dokumente

Validator sind Werkzeuge, mit denen Dokumente auf formale syntaktische Fehlerfreiheit geprüft werden können, indem der Quelltext mit der in der Dokumententypdefinition angegebenen HTML-Spezifikation abgeglichen wird. Bei der Validierung werden uneindeutige Zusammenhänge innerhalb des Codes aufgedeckt. Sie dient aber nicht der Konformitätsprüfung mit dem zugrunde gelegten Standard oder gar der Barrierefreiheit. Das Ergebnis ist entweder die Information, dass ein Dokument den formalen Anforderungen genügt, oder eine Fehlerliste (vgl. Abb. 13-1).

Zwei der zahlreichen Webanwendungen zur Validierung haben sich bewährt:

- W3C Markup Validation Service:
Der Validator des W3C bietet die Prüfung und Korrektur von HTML-Dokumenten. Die Originalversion ist zu finden auf
<http://validator.w3.org/>
- WDG HTML Validator:
Ein Werkzeug der Web Design Group, das sich für Dokumente eignet, in denen andere Dokumententypdefinitionen als die vom W3C spezifizierten verwendet werden. Außerdem ist es möglich, komplette Webangebote in einem Schwung zu validieren. Zu finden ist der Validator auf
<http://www.htmlhelp.com/tools/validator/>

3. Vgl. W3C, Recommended list of Doctype Declaration,
URL: <http://www.w3.org/QA/2002/04/valid-dtd-list.html> (Abruf 5.7.2010).

Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: [Congratulations - Icons](#)

This document was successfully checked as XHTML 1.0 Strict!

Result:	Passed	
Address:	<input type="text" value="http://www.cbr-rechtsberatung.de/"/>	
Modified:	(undefined)	
Server:	Apache/2.2	
Size:	(undefined)	
Content-Type:	text/html	
Encoding:	utf-8	<small>(detect automatically)</small> ▾
Doctype:	XHTML 1.0 Strict	<small>(detect automatically)</small> ▾
Root Element:	html	
Root Namespace:	http://www.w3.org/1999/xhtml	

The W3C validators rely on community support for hosting and development. [Donate](#) and help us build better tools for a better web.

Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: [Validation Output](#)

Errors found while checking this document as HTML 4.01 Transitional!

Result:	182 Errors, 52 warning(s)	
Address:	<input type="text" value="http://www.xxxxx.de/"/>	
Modified:	(undefined)	
Server:	Apache	
Size:	38917	
Content-Type:	text/html	
Encoding:	iso-8859-1	<small>(detect automatically)</small> ▾
Doctype:	HTML 4.01 Transitional	<small>(detect automatically)</small> ▾
Root Element:	HTML	

The W3C validators rely on community support for hosting and development. [Donate](#) and help us build better tools for a better web.

Abb. 13-1 Ergebnisse einer W3C-Validierung für ein valides und ein nicht valides HTML-Dokument

Ein weiteres hilfreiches Werkzeug ist Tidy, eine Anwendung, mit der die Syntax geprüft und aufgeräumt werden kann. Tidy kann auch bestimmte Aspekte der Barrierefreiheit prüfen und liefert hierzu Hinweise. Das Herzstück von Tidy ist

die TidyLib, es wird in diversen Anwendungen von HTML-Editoren bis Validatoren eingesetzt.

Für die Prüfung einzelner Seiten oder Seitenbereiche eignet sich einer der oben genannten Validatoren. Bei größeren Projekten sollte die Serverumgebung mit Tidy so erweitert werden, dass eine automatisierte HTML-Optimierung erfolgen kann.

Viele Editoren und andere Entwicklungsumgebungen bieten zusätzliche Validierungsmöglichkeiten. Dabei geht es selbstverständlich nicht nur um HTML, sondern auch um XML, SVG, SMIL, CSS oder eine der zahlreichen anderen Techniken zur Erstellung von Webdokumenten. Dass prinzipiell andere Techniken ebenso valide sein sollten, steht außer Frage. HTML spielt aber wegen seiner Funktion als technische Benutzungsschnittstelle für Hilfsmittel eine besondere Rolle.

Links zu verschiedener Validierungssoftware des W3C, einschließlich Tidy, finden Sie auf:

<http://www.w3.org/Status.html>

13.1.2.2 Strategie der Fehlervermeidung

HTML-Validierung ist vor allem eine Strategie der Fehlervermeidung und keine Optimierungsstrategie, da man »nur« die offensichtlichen Fehler auf der technischen Ebene feststellen kann.

Bei XML-basierten Auszeichnungssprachen ist auf die in den XML-Spezifikationen definierte Wohlgeformtheit des Dokuments zu achten. Dazu zählt, dass Elemente richtig verschachtelt werden, dass es genau ein Wurzelement gibt oder dass sonstige Regeln für Attribute oder Wertzuweisungen gemäß ihrer Spezifikation eingesetzt werden. Die XML-Regeln sind für XHTML ebenso wie für SVG, SMIL oder andere XML-basierte Auszeichnungssprachen einzuhalten.

Zunächst geht es um die Vermeidung nicht spezifizierter Schreibweisen. Einige HTML-Elemente und -Attribute waren nie Teil einer HTML-Spezifikation und sollten deshalb nicht eingesetzt werden. Ein Beispiel ist das Element MARQUEE zur Erzeugung von Laufschriften. Abgesehen davon, dass MARQUEE nicht von allen Browsern unterstützt wird, führt es gelegentlich zum Absturz von Screenreadern.

Viele Elemente und Attribute, die noch in den 1990er-Jahren genutzt wurden, lassen sich heute problemlos mit CSS umsetzen. Sie werden deswegen vom W3C als veraltet eingestuft. Dies betrifft vor allem solche, die das Layout beeinflussen. Zu den veralteten Elementen gehören u.a. das FONT- oder das U-Element (Unterstreichung). Veraltete Attribute sind beispielsweise hspace, bgcolor sowie das language-Attribut für das SCRIPT-Element. Einige Attribute wiederum sind nur für bestimmte Elemente veraltet, wie border für IMG und OBJECT oder value für LI. Zu beachten ist, dass bei der Validierung nicht alle veralteten Elemente und Attribute als Fehler beanstandet werden. Deswegen müssen Tests



mit weiteren Anwendungen, wie sie in speziellen Toolbars für Webentwickler vorhanden sind, durchgeführt werden.

Viele veraltete Elemente und Attribute sind in den strengen DTDs (strict) nicht zugelassen;⁴ es ist teilweise dem Einsatz dieser Elemente und Attribute zu »verdanken«, dass es alternative Übergangs-DTDs (transitional) gibt.



Die Validierung unterstützt bei der Klärung eindeutiger Beziehungen zwischen einzelnen Inhalten. Beispielsweise sind id-Attribute für bestimmte Elemente für die Zugänglichkeit des Dokuments wichtig. Werden komplexe Datentabellen über headers-Attribute für die einzelnen Zellen zugänglich gemacht, dann ist eine eindeutige Verknüpfung der Datenzellen mit den Zeilen- bzw. Spaltenüberschriften nötig, damit sie von Screenreadern erkannt werden. HTML-Validatoren stellen jedoch nur fest, ob die Werte für id-Attribute von Kopfzellen eventuell doppelt vergeben wurden. Ob die Verknüpfung von Kopfzellen und Datenzellen über headers-Attribute korrekt ist, hängt von den Tabelleninhalten ab und muss »per Hand« und sicherheitshalber direkt mit einem Screenreader ermittelt werden.



Die Verknüpfung von Formularbeschriftungen mit Steuerelementen ist ähnlich. Jedes Formularelement benötigt eine Beschriftung, die eindeutig mit dem Steuerelement verknüpft ist. Kann dieser Zusammenhang nicht über die Verknüpfung durch Screenreader erkannt werden, dann sind vor allem komplexe Formulare von blinden Nutzern kaum noch bedienbar. Voraussetzung ist auch hier die Vergabe eindeutiger Werte in den id-Attributen der Steuerelemente. Aber auch hier kann ein Validator nicht ermitteln, ob die Beschriftungen den richtigen Steuerelementen zugewiesen wurden.



Mit einer Validierung können außerdem kleinere Fehler aufgespürt werden, die zwar oft nicht auffallen, aber irgendwann zu Kompatibilitätsproblemen führen können. Ob HTML-Elemente nicht korrekt verschachtelt, zulässige Attribute eingesetzt oder geöffnete Tags nicht geschlossen sind: Bestimmte Fehler können dazu führen, dass die Inhalte durch die verwendete Software falsch aufbereitet werden. Auch Attribute ohne Anführungszeichen oder fehlende Trennung von Attributen mit Leerzeichen können zu Zugänglichkeitsproblemen führen.

Besteht ein HTML-Dokument die Validierung, ist es zwar formal korrekt aufgebaut, aber nicht zwingend fehlerfrei im Sinne der Zugänglichkeit und Nutzbarkeit. Ein HTML-Validator wird z. B. für den folgenden HTML-Schnipsel keine Fehlermeldung liefern:

```
<a href="unsinn">Linktext</a>
```

Weil in der Dokumententypdefinition als Wert für das href-Attribut ein Inhalt des Typs »CDATA« gefordert wird, ist die Schreibweise formal richtig. Nutzbar ist das allerdings nicht.

4. Vgl. SelfHTML, nicht erlaubte Elemente und Attribute bei Variante strict, URL: http://de.selfhtml.org/html/referenz/varianten.htm#strict_nicht_erlaubt (Abruf 5.7.2010).

Ein weiterer wichtiger Aspekt der Barrierefreiheit, der durch eine Validierung nicht aufgedeckt werden kann, ist die logische Reihenfolge der Inhalte im Quelltext (Linearisierbarkeit). Sie gehört zu den für viele Nutzer problematischen Aspekten und wird im Unterschied zu dem oben aufgeführten Linkbeispiel konkret in den WCAG20 angesprochen.



Wenn ein Tabellenlayout verwendet wird, dann kann es gemäß der HTML-Spezifikation aufgebaut und damit valide sein. Sind aber die Tabellen- oder Zelleninhalte nicht linearisierbar, dann ist die Seite nicht konform nach den Webstandards der Barrierefreiheit.

Bereits diese wenigen Beispiele zeigen, dass trotz formaler Gültigkeit und Einhaltung der HTML-Spezifikation für die Zugänglichkeit und Nutzbarkeit weitere Anforderungen erfüllt sein müssen.

13.1.2.3 HTML semantisch richtig anwenden

In den HTML-Spezifikationen wird nicht nur die richtige Syntax beschrieben, sondern auch die Semantik der Elemente und die korrekte Verwendung der Attribute. Erst die Berücksichtigung dieser Vorgaben stellt sicher, dass die Inhalte von einer Zugangssoftware einschließlich Hilfsmitteln korrekt aufbereitet werden.



Der standardkonforme und semantisch richtige Einsatz der HTML-Elemente und -Attribute wird in diesem Buch in verschiedenen Kapiteln gezeigt: Eine einfache und grundsätzliche Einführung befindet sich in Abschnitt 4.1 ab Seite 117. Die wichtigsten blockbildenden Elemente werden in Tabelle 4-1 auf Seite 124 als Übersicht und in Abschnitt 9.2.4 ab Seite 333 ausführlich behandelt. Der Aufbau von Navigationsleisten wird in Abschnitt 7.2.2 ab Seite 272 erläutert und Steuerelemente für Formulare in Tabelle 15-1 auf Seite 570 sowie Tabelle 15-2 auf Seite 574 aufgelistet.

Wichtig ist, dass HTML-Elemente und -Attribute ihrem Zweck nach eingesetzt werden, denn es geht nicht nur um die bloße sichtbare Darstellung der Inhalte, sondern auch um semantischen und damit bedeutsamen Quellcode. Seien es das Document Object Model (DOM) oder die Schnittstellen für Hilfsmittel, es kommt immer darauf an, dass die Werte, Status und Rollen der Inhalte technisch ermittelt werden können.

Für die rein visuelle Präsentation ist die in den Spezifikationen beschriebene Semantik meist irrelevant. Sehenden Nutzern erschließt sich die Bedeutung der Elemente in der Regel durch die visuelle Gestaltung, z. B. über Farbe, Schriftgröße oder Position. Semantisch korrekt und damit standardkonform ist ein Dokument aber erst, wenn die Bedeutung auf der Strukturebene und damit technisch ermittelbar ist, z. B. durch eine Zugangssoftware, die auf der Datenbasis eine Benutzungsschnittstelle erstellt, wie Screenreader, Vergrößerungssysteme oder diversen Eingabegeräte.

Ein typisches Beispiel ist der falsche Einsatz des BLOCKQUOTE-Elements: Obwohl es einfach erscheint, mit diesem Element z. B. einen normalen Absatz einzurücken (die meisten Browser fügen bei diesem Element einen Einzug ein),

so ist dieser Absatz nicht notwendigerweise ein Zitat. Da Screenreader mit BLOCKQUOTE ausgezeichnete Textpassagen als Zitate ankündigt, wird die Semantik des Absatzes nicht mehr korrekt übertragen und so eine inkorrekte Dateistruktur übermittelt. Selbstverständlich gilt das auch für das UL- oder das DD-Element, die ebenfalls nicht für normale Texteinrückungen verwendet werden sollten.

Ein weiteres Beispiel ist die Emulierung von Links und Steuerelementen mit JavaScript. Mit dem onclick-Attribut kann ein beliebiger Inhalt klickbar gemacht und als Link oder Steuerelement genutzt werden. Solche Texte sind jedoch meist keine Links oder Formularelemente im Sinne der Spezifikation, weshalb die Rolle des Inhalts oft nicht richtig an eine weiterverarbeitende Software übermittelt wird. Bei emulierten Links lässt sich oft feststellen, dass sie mit der Tastatur nicht bedienbar sind (vgl. Abschnitt 16.3.3.1 ab S. 631).



Entscheidend ist die Verwendung standardisierter Schnittstellen. Wird eine Checkbox mit dem INPUT-Element erzeugt, dann kann der Browser die Rolle (Kontrollkästchen) und den Status (aktiviert oder nicht aktiviert) des Elements identifizieren und an entsprechende Schnittstellen weiterreichen. Wenn hingegen eine Checkbox als verlinkte Grafik eingebunden wird, dann ist die Rolle des Elements ein Link. Der (visuell vermittelte) Status muss dann durch zusätzliche Maßnahmen technisch vermittelt werden, etwa durch Alternativtexte für die verlinkten Grafiken. Solche Maßnahmen können selbstverständlich vorgenommen werden, aber standardisierte Elemente erleichtern die Webentwicklung und fördern die Zugänglichkeit deutlich.



Auch weitere Informationen und ihre Beziehungen zueinander müssen im HTML berücksichtigt werden, damit sie auch dann verständlich sind, wenn sich die Präsentation ändert. Dies ist der Fall, wenn die Inhalte ohne CSS (z.B. mit einem Screenreader) gelesen werden oder eine Webseite mit benutzerdefinierten Bildschirmereinstellungen dargestellt wird, etwa eigenen Farbschemata oder Schriftschnitten, die die vorgegebenen CSS-Eigenschaften überschreiben. Auch hier geht es darum, dass sichtbare Informationen oder Beziehungen zwischen Informationen immer auch technisch ermittelt werden können. Richtige Verschachtelungen und logische Beziehungen dürfen nicht vernachlässigt werden. Diese Aspekte können meist nur durch praktische Tests und nicht mit einem Validator geprüft werden.



Dass zugänglichkeitsunterstützende Techniken nicht immer zu barrierefreien Ergebnissen führen, wird vor allem bei Datentabellen deutlich. Wird der Inhalt einer Datentabelle im summary-Attribut zusammengefasst, dann steht diese Information nur Screenreader-Nutzern zur Verfügung. Beschreibungen, vor allem von komplexen Datentabellen, können aber auch für andere Nutzer hilfreich sein. Hier gilt es abzuwägen, ob die Beschreibung als CAPTION-Element und somit für alle sichtbar oder sogar im Kontext der Tabelle stehen kann oder sollte. Im Einzelfall muss überlegt werden, welche Technik die barrierefreie Nutzung für alle am besten fördert. Während summary-Attribute oder CAPTION-Elemente

durch Software ermittelbare Techniken sind, ist die kontextuelle Beschreibung maschinell nicht überprüfbar, sodass auch dieses Beispiel die Grenzen eines Validators zeigt.

Auf einer technischen Ebene ist Barrierefreiheit nur mit den Grundsätzen des »Progressive Enhancement« (vgl. Abschnitt 6.1 ab S. 197) zweckmäßig. Dem Grunde nach geht es darum, die Inhalte so aufzubereiten, dass sie auf allen Endgeräten zumindest funktionieren, was die Aufbereitung von reinen HTML-Seiten ohne weitere Techniken bedeutet. CSS, JavaScript und andere Techniken werden als optionale Techniken angesehen und das Fehlen dieser Techniken in Endgeräten darf bei diesem Ansatz nicht dazu führen, dass die Seite ganz oder in Teilen nicht mehr funktioniert.⁵

13.1.3 Werkzeuge zur Prüfung der Barrierefreiheit

Automatisierte Prüfwerkzeuge für Barrierefreiheit gibt es nicht – zumindest keine, mit denen eine erfolgreiche Prüfung möglich wäre –, denn Barrierefreiheit ist keine formale Grammatik. Die vorhandenen Werkzeuge können zwar unterstützen (vgl. Abschnitt 2.4.1.1 ab S. 52), liefern aber keine genauen Ergebnisse. Am sichersten ist immer die Einbeziehung qualifizierter Prüfer und behinderter Nutzer.

Die wichtigsten Werkzeuge zur Prüfung von Barrierefreiheit sind die Web Accessibility Toolbar für Internet Explorer und Opera sowie die Web Developer Toolbar für Firefox. Mit diesen auch in deutscher Sprache verfügbaren Tools können verschiedene Elemente eines HTML-Dokuments identifiziert und hervorgehoben werden. Außerdem sind alternative Darstellungen einer Seite möglich. Die manuelle Prüfung wird dabei recht umfassend unterstützt. Für Firefox stehen außer der oben genannten Web Developer Toolbar weitere Werkzeuge zur Verfügung, wie die Juicy Studio Toolbar, die Firefox Accessibility Extension und WAVE.

Viele Aspekte der Barrierefreiheit können mit Firefox geprüft werden. Trotzdem darf der Internet Explorer in keiner Testsuite fehlen, insbesondere wenn es um das Zusammenspiel mit Hilfsmitteln geht. Obwohl Firefox die Zugänglichkeit unterstützt, funktionieren viele Hilfsmittel behinderter Nutzer immer noch besser mit dem Internet Explorer. Außerdem kann es in verschiedenen Browsern zu Unterschieden bei der Darstellung von Inhalten kommen, sodass auch eine Prüfung im Internet Explorer immer nötig ist. Dies betrifft u. a. das Verhalten einer Webseite bei Schriftvergrößerung und den sichtbaren Fokus bei Einsatz der Tabulatortaste. Auch weitere Aspekte der Tastaturbedienbarkeit, wie die logische Reihenfolge der Inhalte oder Tastaturfallen, müssen immer in verschiedenen Browsern getestet werden.

Manche Aspekte wie Überschriftenhierarchien und Alternativtexte lassen sich sowohl mit der Web Developer Toolbar für Firefox als auch mit der Web

5. Filament Group, Designing with progressive enhancement, URL: <http://filamentgroup.com/dwpe/> (Abruf 5.7.2010).

Accessibility Toolbar für Internet Explorer gut testen. Die Barrierefreiheit von einfachen und komplexen Datentabellen hingegen lässt sich besser mit den entsprechenden Funktionen der Web Accessibility Toolbar testen, da die Tabellenattribute dort deutlicher angezeigt werden.⁶

13.2 Allgemeine Sorgfaltspflichten

In der technischen Umsetzung gilt es zahlreiche Sorgfaltspflichten zu beachten. In vielen Fällen handelt es sich um Details. Oft führen aber nicht beachtete Details zu Schwierigkeiten. Das können ungenügende Angaben zur Zeichenkodierung sein oder – in internationalen Projekten – falsche Angaben zur Dokumentsprache oder Schriftrichtung. In den folgenden Abschnitten stellen wir einige teilweise sehr kurze Code-Schnipsel vor, die – trotz ihrer Kürze – die Zugänglichkeit eines Webauftritts stark beeinflussen können.

13.2.1 Zeichenkodierung

Die korrekte Angabe der Zeichenkodierung gehört nicht zu den Anforderungen der Barrierefreiheit, ist aber ein allgemeines Problem. Immer wieder werden z.B. Umlaute in dem einen Browser richtig, aber in einem anderen Browser falsch und gelegentlich sogar in allen Browsern falsch dargestellt.



Abb. 13-2 Schwer lesbarer Text aufgrund falscher Zeichenkodierung

6. Siehe Probiesch, K. (2009), Barrierefreiheit mit Firefox testen, in: Webstandards Magazin 3/2009. Vgl. auch <http://www.paciellogroup.com/resources/wat-ie-about.html>.