

DIPLOMARBEIT

BARRIEREFREIE, JAVA-SCRIPT-GESTÜTZTE WEBAPPLIKATIONEN IM PRAXISNAHEN UMFELD

Firma Namics (Deutschland) GmbH
Diplomand: Felix Nagel
Referent: Prof. Dr.-Ing. Burkhard Kampschulte
Korreferent: Prof. Dr. Stephan Euler
Betreuer: Markus Staeuble
 Alexander Stirn

Fachhochschule Gießen-Friedberg, Bereich Friedberg
Fachbereich IEM – Medieninformatik, 8. Semester – 2009

Inhaltsverzeichnis

Anmerkung des Verfassers	v
1 Einleitung	1
1.1 Vorwort	1
1.2 Allgemeine Problemdefinition	2
1.3 Zielsetzung und Aufbau	4
1.4 Abgrenzung	5
2 Theoretische Grundlagen	7
2.1 Zielgruppen	7
2.1.1 Sensorische Fähigkeiten	8
2.1.2 Motorische Fähigkeiten	9
2.1.3 Mentale Fähigkeiten	9
2.2 Gründe für Barrierefreiheit	10
2.2.1 Geschäftliche	10
2.2.2 Gesetzliche	11
2.2.3 Ethische	12
2.2.4 Technische	13
3 Technische Grundlagen	15
3.1 Unterstützungstechnologie	15
3.1.1 Screenreader	15
3.1.2 Bildschirmlupen	21
3.1.3 Textbasierte Browser	21
3.1.4 Braillezeile	24
3.1.5 Großschrift-Tastaturen	24
3.1.6 Mausersatz	25
3.2 jQuery	25
3.2.1 Barrierefreiheit	26
3.2.2 Alternativen	27
3.3 Schnittstellen zu Datenübertragung	27

4	Technische Richtlinien	29
4.1	ISO	32
4.1.1	ISO 9241-151	32
4.1.2	ISO 9241-171	35
4.2	BITV	38
4.2.1	Aufbau und Relevanz	39
4.2.2	Kritik an der BITV	40
4.2.3	BITV 2	40
4.3	WCAG 2.0	41
4.3.1	Aufbau und Relevanz	42
4.3.2	Kritik an der WCAG 2.0	50
4.4	WAI ARIA	51
4.4.1	Tastatur-Navigation und Fokus	52
4.4.2	Rollenvergabe	53
4.4.3	Beziehungen von Elementen	55
4.4.4	Live-Regionen	55
4.4.5	Formular Eigenschaften	56
4.4.6	Drag und Drop	57
5	Praktische Umsetzung	59
5.1	Programmierrichtlinien	59
5.1.1	Grundsätzliches	59
5.1.2	HTML	60
5.1.3	Java-Script	61
5.1.4	Cascading Stylesheets	61
5.2	Formular	62
5.2.1	Definition	62
5.2.2	Umsetzung	63
5.2.3	Dokumentation	66
5.3	Lightbox	79
5.3.1	Definition	79
5.3.2	Umsetzung	79
5.3.3	Dokumentation	81
5.4	Tabs	91
5.4.1	Definition	91
5.4.2	Umsetzung	92
5.4.3	Dokumentation	94
5.5	Sortierbare Tabellen	96
5.5.1	Definition	96
5.5.2	Umsetzung	97
5.5.3	Dokumentation	99

6 Fazit	109
7 Literaturverzeichnis	113
8 Abbildungsverzeichnis	123
9 Quellennachweise für Abbildungen	125
10 Quellcodeverzeichnis	127
11 Tabellenverzeichnis	129
12 Abkürzungsverzeichnis	131
A Erklärung der Selbstständigkeit	135
B Angefügte Dokumente	137
B.1 Formular	137
B.1.1 HTML Quellcode	137
B.1.2 JavaScript Quellcode	143
B.2 Lightbox	158
B.2.1 HTML Quellcode	158
B.2.2 JavaScript Quellcode	162
B.3 Tabs	174
B.3.1 JavaScript Quellcode	174
B.4 Sortierbare Tabellen	178
B.4.1 HTML Quellcode	178
B.4.2 JavaScript Quellcode	180
C Danksagung	195

Anmerkung des Verfassers

Aus Gründen der Lesbarkeit habe ich auf die weiblichen Formen bei Bezeichnungen von Personen und Personengruppen verzichtet. Mit „der Nutzer“, etc. ist immer auch „die Nutzerin“ gemeint.

Besonders im Bereich des Internets finden aufgrund seiner Entstehung meist englische Begriffe Verwendung. Ich habe versucht, diese mit deutschen Äquivalenten zu ersetzen.

Oftmals sind die Bezeichnungen für bestimmte Techniken oder Elemente nur schwer ins Deutsche zu übersetzen, haben übersetzt kaum Verbreitung oder treffen nicht den eigentlichen Sinn. Bestimmte Begriffe wurden deshalb nicht übersetzt. Sie werden in der Arbeit an entsprechenden Stellen näher erläutert.

KAPITEL 1

Einleitung

1.1 Vorwort

“The power of the web is in its universality. Access by everyone regardless of disability is an essential aspect!”¹

Sir Timothy John Berners-Lee

Direktor des [W3C](#)² und Erfinder des World Wide Web

Für viele Menschen mit Behinderung stellen einige alltägliche Aufgaben – wie das Haus verlassen, um mit dem Bus zum Einkauf zu fahren oder auch nur den Busfahrplan zu lesen – eine echte Herausforderung dar. Dies betrifft nicht nur Menschen mit einer anerkannten Behinderung sondern auch Personen, die zum Beispiel unter einer Lese- oder Aufmerksamkeitsschwäche leiden, auf eine Sehhilfe angewiesen sind oder Probleme mit der Hand-Auge-Koordination haben. Letzten Endes haben die meisten Menschen mit Problemen dieser Art zu kämpfen – spätestens mit fortschreitendem Alter.

Das Internet hat unser Leben verändert. Es bietet Unterhaltung in nie da gewesener Vielfalt und einen Informationsaustausch, der noch nie so schnell, umfassend und weitgehend unzensiert gewesen ist. Es bietet unzählige Chancen, Geld zu verdienen, und noch mehr, welches auszugeben. Man kann mit Menschen rund um den Globus in Kontakt treten, um zusammen aktiv zu werden oder nur, um den neusten Klatsch auszutauschen, ohne dabei auch nur das Haus zu verlassen.

Das Internet bietet für viele Menschen erstmals die Möglichkeit, wirklich aktiv am sozialen, gesellschaftlichen und auch politischen Leben teilzuhaben.

Obwohl Barrieren im Internet verhältnismäßig kostengünstig und unkompliziert abgebaut

1 Quelle: Startseite WAI der W3C [Berners-Lee \[2009\]](#)

2 World Wide Web Consortium; ein Gremium zur Standardisierung von Internettechnologien.

werden könnten, treffen Menschen mit verschiedensten Beeinträchtigung auf virtueller Ebene auf alte und neue Hürden.

Gerade mit dem Einzug des Web 2.0, das ja eigentlich für das Einbeziehen und die Interaktion der Nutzer stehen sollte, wurden viele Dienste im Internet unbenutzbar. Das liegt unter anderem an eben diesem „Mitmach-Internet“, in dem die meisten Inhalte eben nicht mehr redaktionell erstellt, sondern von Laien angeboten werden. Dies verstärkt altbekannte Probleme wie fehlende Alternativ-Texte für Bilder, eine undeutliche Sprache, schlechte Grammatik oder unlesbare Typografie. Die sehr beliebten multimedialen Inhalte wie Videos oder Flash-Animationen stellen ebenfalls eine Hürde für Menschen mit Behinderung dar. Außerdem wurden durch neue Technologien und deren Möglichkeiten – wie den massiven Einsatz von modernem Java-Script wie zum Beispiel [AJAX](#)¹ – Webapplikationen (eine im Browser laufende Software) geschaffen, die in ihrer Komplexität Desktopanwendungen (zum Beispiel dem Browser selbst) in nichts nachstehen. Das Fehlen bzw. die mangelhafte Verwendung von Standards haben für viele Menschen eine große Anzahl von Problemen geschaffen.

In meiner Arbeit möchte ich zeigen, dass es mit Hilfe der entsprechenden Technik und dem Einhalten gewisser Standards möglich ist, eine barrierefreie, technisch moderne Webseite bzw. Webapplikation zu entwickeln.

1.2 Allgemeine Problemdefinition

Mit der Veränderung des Internet vom Web 1.0, dem statischen Anbieten von Inhalten und Services, hin zum Web 2.0, dem Einbeziehen von sogenannten nutzergenerierten Inhalten und der verstärkten Nutzung von multimedialen Inhalten, haben viele neue Techniken Einzug gehalten.

Mit Techniken im Java-Script-Umfeld (wie [AJAX](#), Echtzeit-Berechnungen und [DOM](#)²-Manipulationen sowie auf [JS](#)³ und [CSS](#)⁴ basierten Animationen) werden moderne Webseiten heute immer mehr den bekannten Desktopanwendungen nachempfunden. Das bedeutet, dass zum Beispiel Dialogfelder nicht mehr als sogenanntes Pop-Up-Fenster angezeigt werden, sondern dass per Java-Script-Code ein mit [CSS](#) graphisch gestaltetes HTML-Konstrukt in den [DOM](#) injiziert wird und dieses dann an geeigneter Stelle im Browser angezeigt wird. Sehr beliebt sind auch auch sogenannte *Lightbox*-Applikationen,

-
- 1 Asynchronous JavaScript and XML; ein Konzept der asynchronen Datenübertragung, das es ermöglicht, vom Server geladene Daten im Browser verfügbar zu machen, ohne die Seite neu zu laden (umgs. „nachladen“).
 - 2 Document Object Model; eine Spezifikation einer Schnittstelle für den Zugriff auf HTML- oder XML-Dokumente.
 - 3 Java-Script
 - 4 Cascading Style Sheets

die einzelne Bilder oder Serien von Bildern nach Klick auf das entsprechende Vorschaubild in einer größeren Version darstellen. Weitere Beispiele wären die aus Windows-Einstellungsfenstern bekannten Reiter (sog. *Tabs*) zum Anzeigen verschiedener Ebenen oder die Echtzeit-Validierung von Formularfeldern.

Neben diesen vergleichsweise einfachen Anwendungen gibt es auch noch richtige Applikationen mit großem Funktionsumfang – die sogenannten **RIA**¹. Viele dieser **RIA** Anwendungen erfreuen sich großer Beliebtheit. Google Mail, DeviantArt², Twitter³ und viele andere nutzen exzessiv neue Technologien, um eine möglichst gute Benutzerfreundlichkeit zu gewährleisten.

Problematisch ist aber, dass diese oft nicht barrierefrei zugänglich sind. Das liegt unter anderem an der bisherigen Limitierung der technischen Hilfsmittel, der sogenannten **AT**⁴. Für Bildschirmleseprogramme (sog. *Screenreader*) beispielsweise war es bis vor einiger Zeit – und ist es ohne Zutun der Webentwickler bis heute – kaum möglich, **DOM**-Manipulationen zu bemerken. Das heißt, dass die Aktualisierung im **HTML**⁵ zwar vom Browser wahrgenommen wird, das Hilfsprogramm diese Änderung aber nicht erkennt und dem Nutzer daher nicht mitteilen kann. Das kann bedeuten, dass es nicht möglich ist, einen wichtigen Dialog zu beantworten – der Nutzer bricht die Verwendung vermutlich ab.

Ein Beispiel für schlechte technische Umsetzung dieser Art wäre Facebook⁶, eines der weltweit am meisten genutzten sozialen Netzwerke⁷. Aufgrund seiner Beschaffenheit ist es so gut wie nicht barrierefrei und mit einem Bildschirmleseprogramm⁸ kaum nutzbar. Auch die Videos auf der bekannten Plattform YouTube⁹ lassen sich standardmäßig nur durch die ins Flash eingebauten Bedienfelder steuern. Flash an sich ist zwar unter bestimmten Voraussetzungen barrierefrei, aber nur bei entsprechender Sorgfalt während der Entwicklung. Chris Heilmann, Entwickler bei Yahoo, erstellte mit Hilfe der YouTube **API**¹⁰ im Nachhinein eine barriearme Bedienoberfläche für die Videointhalte¹¹.

Google Mail ist von sich aus barriearm – es kann fast vollständig genutzt werden¹².

1 Rich Internet Application; komplexe, meist benutzerfreundliche Webapplikationen, die mit modernen Techniken funktionieren.

2 <http://www.deviantart.com>

3 <http://www.twitter.com>

4 Assistive Technology; Unterstützungstechnologie; befasst sich mit technischen Hilfen speziell für Menschen mit Behinderungen, zum Beispiel Braillezeilen oder Bildschirmleseprogramme

5 Hypertext Markup Language

6 <http://www.facebook.com>

7 vgl. comScore Pressemeldung [comScore \[2009\]](#)

8 vgl. Äußerung C. Heilmann ab Minute 00:49 auf der EAFRA Konferenz [Heilmann \[2009\]](#)

9 <http://www.youtube.com>

10 Application Programming Interface; eine Schnittstelle, die von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird.

11 <http://icant.co.uk/easy-youtube/>

12 vgl. <http://mail.google.com/support/bin/answer.py?hl=en&answer=90559>

Die barrierefreie Gestaltung einer modernen *State-of-the-Art*-Internetseite ist also mit dem Einhalten bestimmter Richtlinien und der Nutzung von modernen Techniken durchaus möglich.

1.3 Zielsetzung und Aufbau

Ziel dieser Arbeit ist eine Toolbox – eine Art Best-Practice – zur Erstellung von barrierefreien Internetseiten. Sie soll für Projekte im internationalen Einsatz konzipiert werden, allerdings mit speziellem Augenmerk auf den deutschen Markt.

Vorab erfolgt eine Definition der Zielgruppen und die Erörterung von Gründen für die Erstellung von barrierefreien bzw. barrierearmen Webseiten und deren Applikationen. Dies schließt eine Abwägung der geschäftlichen sowie ethischen Aspekte – anhand von Statistiken und Gesetzen bzw. Richtlinien – und technische Aspekte, zum Beispiel SEO¹ und mobiles Internet ein.

Dem schließt sich die Analyse der von den jeweiligen Zielgruppen verwendeten Hilfsmittel und der technischen Spezifikationen und Richtlinien an. Daraus ergeben sich mögliche Best-Practice, die für die praktische Umsetzung mit technischen Restriktionen verglichen werden.

Diesem theoretischen Teil folgt die Beschreibung der praktischen Umsetzung, aus der die angestrebte Tool-Box entwickelt werden soll. Die einzelnen umzusetzenden Design Pattern² und deren Probleme werden erörtert. Diese werden dann mit den Richtlinien und Spezifikationen verglichen und Lösungsmöglichkeiten werden erarbeitet.

Technisch sollen diese in validem XHTML³ 1.1 Strict, validem CSS 2.0 sowie dem von Namics⁴ zumeist verwendeten Java-Script Framework JQuery 1.3.x umgesetzt werden. Die Arbeit soll den Web Content Accessibility Guidelines 2.0 Richtlinien des W3C entsprechen.

Falls nötig sollen die umzusetzenden Muster-Anwendungen eine einfache Möglichkeit zum Datenaustausch mit dem von Namics am meisten verwendeten Content-

1 Search Engine Optimization

2 dt.: Entwurfsmuster; sind bewährte Lösungs-Schablonen für wiederkehrende Entwurfsprobleme der Softwarearchitektur und Softwareentwicklung

3 Extensible HyperText Markup Language

4 **Namics (Deutschland) GmbH**; der Internetdienstleister mit deren Hilfe der Autor seine Diplomarbeit schreibt.

Management-System [CQ¹](#) von Day ermöglichen. Zusätzlich soll eine universelle [XML²](#)-Schnittstelle implementiert werden.

Die Standard-Applikationen Formular (bzw. dessen Live-Validierung), Lightbox, Accordion und Tabs sollen umgesetzt werden. Wenn diese bereits in der *jQuery UI* Bibliothek vorhanden sind, ist zu erörtern, in wie weit Barrierefreiheit bereits implementiert wurde.

Am Ende soll ein Paket aus einfachen Muster-Templates zur Verwendung beim Entwickeln von Webseiten stehen, bestehend aus HTML und CSS sowie Java-Script-Klassen und Code-Elementen, die dann entsprechend angepasst werden können, um die jeweiligen Projekt-Spezifikationen zu erfüllen.

1.4 Abgrenzung

In dieser technisch orientierten Arbeit soll auf die Möglichkeit einer barrierefreien Umsetzung von Internetapplikationen eingegangen werden. Dies beinhaltet die Umsetzung von einigen Use-Cases anhand von Erfahrungswerten, Statistiken und bestehenden Richtlinien. Barrierefreiheit umfasst viele zu beachtende Felder, auf die in dieser Arbeit nicht in ihrer Gesamtheit eingegangen werden kann. Im folgenden wird eine Abgrenzung dieser Felder zu den bearbeiteten Aspekten vorgenommen.

Um einen Internetauftritt bzw. dessen Seiten barrierefrei zu gestalten, sind neben den technischen Bedingungen auch visuelle Vorgaben zu beachten. Dies beinhaltet beispielsweise das Einhalten von Farb- und Helligkeitskontrasten, um Menschen mit verschiedenen Sehbehinderungen das Erfassen der Inhalte zu erleichtern. Ebenfalls relevant für visuell beeinträchtigte Menschen ist eine oft schlechte Typografie und der problematische Einsatz von Text als Grafik. Auch die Nutzbarkeit für Menschen mit eingeschränkten motorischen Fähigkeiten hängt unter anderem vom Design ab, da zum Beispiel die Größe von Schaltflächen gestalterisch festgelegt wird.

Diese Aspekte werden so weit wie möglich beachtet, aber nicht näher beleuchtet. So wird auf Farben nur eingegangen, insofern sie die technische Umsetzung betreffen. Dies kann zum Beispiel bei der Art einer Kennzeichnung von Belang sein.

Die Gestaltung und das damit einhergehende Navigationskonzept sowie die Informationsarchitektur beeinflussen die Nutzbarkeit.

Eine konsistente Navigation mit klarer Auszeichnung und sinnvoller Navigationstiefe ist wichtig, um eine intuitive und zielgerichtete Benutzerführung zu ermöglichen. Ein

1 Communiqué; ein auf Java basierendes Enterprise-Content-Management-System der Firma Day; aktuell in Version 5 (CQ5).

2 Extensible Markup Language

definierter und konsistenter Seitenaufbau trägt ebenfalls dazu bei, die Nutzung zu erleichtern.

Da in dieser Arbeit spezifische Applikationen umgesetzt werden, ist die Gestaltung (die ohnehin im späteren Einsatz abgeändert wird) der gesamten Seite nachrangig. Bei den Benutzeroberflächen soll hingegen auf eine Gestaltung nach barrierefreien Konzepten geachtet werden.

In modernen Internetauftritten werden verschiedene Techniken für multimediale Inhalte genutzt. Video und Audio werden in verschiedenen Formaten wie *flv*, *mp3*, *ogg*, *avi* und vielen anderen angeboten. Die Möglichkeiten, diese Formate abzuspielen, sind noch vielfältiger. Als Beispiele wären *Adobe Flash*, *Apple Quicktime*, *Real Player* oder *Microsoft Silverlight* zu nennen. Diese eingebetteten Formate stellen oft eine ernst zu nehmende Hürde dar – sei es aufgrund ihrer Architektur oder ihres Inhalts. Auf diese wird in dieser Arbeit jedoch nicht eingegangen, da diese Art von Inhalten keine Verwendung findet.

Nachdem nun die Problemdefinition, die Zielsetzung und die Abgrenzung erfolgt sind, sollen im folgenden die theoretischen Grundlagen für das Verständnis von Barrierefreiheit im Internet ausgearbeitet werden.

KAPITEL 2

Theoretische Grundlagen

2.1 Zielgruppen

Die Zielgruppe von Barrierefreiheit wird oft auf Menschen mit Behinderung reduziert. Dabei wird oft vergessen, was Barrierefreiheit im Internet eigentlich bedeutet. Barrierefreiheit wird im englischen als *Accessibility*, als Zugänglichkeit, bezeichnet und trifft damit den Kern der Sache. Eine Internetseite sollte einem möglichst breitem Publikum zugänglich gemacht werden und von diesem auch bedient werden können. Dabei sind beide naturgemäß eng mit dem Begriff der Nutzbarkeit verwandt. Eine bessere Nutzbarkeit – und nichts anderes wird mit dem Einhalten der verschiedenen Richtlinien für Barrierefreiheit erreicht – hilft jedem Benutzer, sein Ziel schnell, einfach und erfolgreich zu erreichen. Besonders für unerfahrene Nutzer, oftmals auch für ältere Nutzer mit wenig Erfahrung mit dem Medium Internet, ist eine gute Nutzbarkeit unerlässlich. Gute Lesbarkeit, bedienbare Navigation, durchdachte Benutzerführung, konsistente Gestaltung, um nur einige Beispiele zu nennen, kommen jedem Nutzer zu Gute. Somit hilft das Einhalten von bestimmten Richtlinien allen Zielgruppen.

Auch Richtlinien spezifisch für Behinderungen sind in ihrer potenziellen Zielgruppe weiter gefasst als oftmals angenommen. Eine Beeinträchtigung kann, muss aber nicht, eine Behinderung in der umgangssprachlichen Definition sein. Das Tragen einer starken Sehhilfe, Farbblindheit, ein gebrochener Arm, starke Migräne oder schlicht schlechte Sprachkenntnis stellen ebenfalls eine Behinderung dar, die von Barrierefreiheit profitieren kann. Menschen mit Lese- und Schreibschwächen oder geringem Bildungsstandard profitieren ebenfalls von einer barrierefreien, nutzbaren Gestaltung von Benutzeroberflächen und Inhalten.

Eine Behinderung, wie man sie im allgemein definiert wird, kann verschiedene Ursachen haben. Sie kann von Geburt an vorhanden sein oder durch Unfälle oder Krankheiten verursacht werden. Spätestens im hohen Alter sind die meisten Menschen von verschiedenen Gebrechen geplagt, die eine anerkannte Behinderung darstellen.

Behinderungen können grundsätzlich zeitweise oder dauerhaft auftreten und Kombinationsbehinderungen sind möglich bzw. oftmals der Regelfall. Ein Beispiel hierfür wäre das Auftreten einer motorischen Störung des Bewegungsapparats zusammen mit Alterssichtigkeit in zunehmendem Alter.

Grundsätzlich unterscheidet man zwischen erworbenen Behinderungen (perinatale entstandene Schäden, Krankheiten, körperliche Schädigungen oder Alterungsprozesse) und angeborenen Behinderungen (durch Vererbung, durch pränatale Schädigungen).

Im nachfolgenden sollen nun die verschiedenen möglichen Behinderungsformen erläutert werden und auf ihre mögliche Beeinträchtigung bei der Nutzung von Internetapplikationen hingewiesen werden. Hierbei werden nur für diese Arbeit relevante Behinderungsformen berücksichtigt und es wird kein Anspruch auf Vollständigkeit erhoben.

Der Autor bezieht sich bei der Einteilung auf die ICF¹-Klassifikation der Weltgesundheitsorganisation (WHO).

2.1.1 Sensorische Fähigkeiten

Als sensorische Wahrnehmung werden Sinneswahrnehmungsvorgänge von Lebewesen, also Sehen, Hören, Schmecken, Riechen und Fühlen bezeichnet. Für Internetapplikationen sind derzeit nur Sehen und Hören relevant, wobei das Sehen aufgrund des stark visuell geprägten Mediums Internet am wichtigsten ist.

Eine sensorische Behinderung des Sehens kann ein vorübergehender oder dauerhafter Verlust der Sehkraft oder Teile derselben sein. Ausprägungen reichen von gewöhnlicher Fehlsichtigkeit, selektiver Farbwahrnehmung und eingeschränkter Wahrnehmung von Helligkeitsunterschieden über den Verlust des Sehvermögens in verschiedenen Formen bis hin zur Vollblindheit (Amaurosis). Auch das Fehlen einer nötigen Sehhilfe (wie durch Vergessen der Brille) stellt eine solche Beeinträchtigung dar.

Hilfsmittel können von optischen Sehhilfen, softwaregestützten Bildschirmlupen bis hin zu Braillezeilen und Screenreadern reichen. Visuelle Informationen können nicht oder nur schwer wahrgenommen werden und müssen daher anders erfassbar gemacht werden, zum Beispiel durch Audiodeskription oder Alternativ- bzw. Beschreibungstexte. Auch Hilfen wie das Anbieten von invertierten Darstellungen, lesbarer Typografie und ausreichendem Kontrast sind nützlich.

Ebenfalls zu den sensorischen Fähigkeiten zu zählen ist das Hören. Die Beeinträchtigung

¹ International Classification of Functioning, Disability and Health; eine von der WHO erstellte und herausgegebene Klassifikation zur Beschreibung des funktionalen Gesundheitszustandes, der Behinderung, der sozialen Beeinträchtigung sowie der relevanten Umweltfaktoren von Menschen.

dieser Fähigkeit kann von einer leichten Schwerhörigkeit bis zur vollständigen Gehörlosigkeit reichen. Im Gegensatz zur immer noch verbreiteten Annahme sind Gehörlose mitnichten grundsätzlich auch stumm, da Sprechen auch ohne Gehör erlernt werden kann.

Hilfsmittel sind Hörgeräte in ihren verschiedenen Formen.

Personen mit Schwierigkeiten beim Hören haben Probleme damit rein akustische Informationen wahrzunehmen. Beispielsweise sind sie nicht in der Lage, einen Warnton zu hören bzw. zu interpretieren oder diesen ggf. zu orten. Deshalb müssen relevante Informationen auch visuell kenntlich gemacht werden und zum Beispiel mit Untertiteln ausgestattet werden.

Neben körperlichen Gründen für verminderte Hörfähigkeit sollten auch Personen mit Gehörschutz oder allgemein in lauter Umwelt befindliche Personen berücksichtigt werden.

2.1.2 Motorische Fähigkeiten

Einschränkungen der motorischen Fähigkeiten können verschiedene Formen und Ursachen haben. Zur Benutzung von Internetapplikationen ist die allgemeine Bewegungsfähigkeit des ganzen Körpers nicht erforderlich. Für gewöhnlich sollte die Mobilität und Stabilität von Gelenken und Knochen, die Kraft-, Tonus- und Ausdauerfunktion der Muskeln sowie die bewusste Steuerung des Armes und der Hand ausreichend für eine Interaktion mit dem Ein- bzw. Ausgabegerät sein. Oft werden Betroffene durch Koordinationsschwierigkeiten, Schwäche, begrenzte Reichweite (zum Beispiel durch Kleinwüchsigkeit, aber auch bei Kindern), unwillkürliche Bewegungen und Schmerzen an der Nutzung von Applikationen gehindert oder dieselbe wird zumindest erschwert. Ausprägungen reichen von temporären oder chronischen Gelenkschmerzen und gebrochenen Gliedmaßen über krankheitsbedingtes Zittern und Sprechbehinderungen bis hin zu fehlenden Gliedmaßen und Querschnittslähmung.

Diese Behinderungen sind oft auch in höherem Alter in Form von Verlangsamung der Reaktionszeit und Abnahme der Geschwindigkeit von motorischen Aktionen anzutreffen. Hilfsmittel sind sehr spezifisch und helfen individuell. Beispiele für technisch anspruchsvollere AT wären Blickbewegungsregistrierung, Maus- und Tastatureratz oder auch Spracherkennungssoftware. Hilfestellung kann zum Beispiel durch ausreichend große Elemente und Tastaturbedienbarkeit gegeben werden.

2.1.3 Mentale Fähigkeiten

Die Aufnahme von Inhalten und die Interaktion mit einer Internetseite oder einer Applikation werden maßgeblich von den kognitiven Fähigkeiten beeinflusst. Diese Fähigkeiten sind sehr unterschiedlich ausgeprägt und ändern sich fortwährend.

Wichtig für die Nutzung von Internetapplikationen sind Lern-, Abstraktions- und Assoziationsfähigkeiten, die Aufmerksamkeitsspanne und sprachliche Fähigkeiten. Probleme

können durch mangelnde Konzentrationsfähigkeit, Einschränkungen in Bezug auf Kurz- und Langzeitgedächtnis, eingeschränkte Kombinationsfähigkeit, aber auch durch fehlendes Sprachverständnis ausgelöst werden.

Eine solche Beeinträchtigung kann verschiedene Gründe haben, beispielsweise kann sie durch eine psychische Störung, eine Sprachbehinderung, geistige Behinderung (mentale Retardierung) oder eine Lernbehinderung begründet sein.

Auch Aspekte wie die Erfahrung mit dem Medium Internet und der zugrunde liegenden Technologie, die Allgemeinbildung und der Bildungsstand haben Auswirkungen auf das Verständnis und die Interaktionsfähigkeit eines Nutzers.

Hilfsmittel sind in technischer Form nicht gegeben, Unterstützung kann aber zum Beispiel in Form guter Nutzbarkeit der Navigation und eines konsistenten Seitenaufbaus, einer lesbaren Typografie und einfachen, klaren Sprache und Anbieten von Zusammenfassungen gegeben werden.

2.2 Gründe für Barrierefreiheit

2.2.1 Geschäftliche

Weltweit leben mindestens 650 Millionen Menschen mit Behinderung¹. Davon sind ca. 314 Millionen in ihrer Sehkraft beeinträchtigt (45 Millionen davon sind Blind)² und ca. 278 Millionen hörgeschädigte Menschen³.

Allein in Europa leben ca. 50 Millionen Menschen mit Behinderung. In den USA machen Menschen mit Behinderung ungefähr 15 Prozent der Gesamtbevölkerung (ca. 41 Millionen Personen) aus⁴. Über 21 Millionen U.S. Bürger haben eine signifikante Sehschwäche und 1,3 Millionen davon sind blind⁵.

In Deutschland leben ca. 6,9 Millionen schwerbehinderte Menschen, das sind rund 8,4 Prozent der Bevölkerung⁶. Die meisten davon (63 Prozent) beziehen Rente oder Pension, was auf die Menge der älteren Personen (71 Prozent ist älter als 55 Jahre) zurückzuführen ist. Weitere 19 Prozent bestreiten ihren Lebensunterhalt selbst⁷.

Anhand dieser Zahlen wird deutlich, dass die Zielgruppe *Menschen mit Behinderung* eine wirtschaftlich durchaus interessante darstellt.

1 vgl. Vortrag C. McClain-Nhlapo (Weltbank) auf der EAFLA Konferenz; ab 2:30 [McClain-Nhlapo \[2009\]](#)

2 vgl. WHO - Fact Sheet N°282 - Visual impairment and blindness [Organization \[2009\]](#)

3 vgl. WHO - Fact Sheet N°300 - Deafness and hearing impairment [Organization \[2006\]](#)

4 vgl. U.S. Department of Education - 2007 Disability Status Report [U.S. Department of Education u. Research \[2007\]](#)

5 vgl. AFB - Facts and Figures on Americans with Vision Loss [for the Blind \[2008\]](#)

6 laut: DESTATIS - Pressemitteilung Nr.258 vom 17.07.2008; Daten von 2007 [Deutschland \[2008a\]](#)

7 laut: DESTATIS - Pressemitteilung Nr.406 vom 30.10.2008; Daten von 2005 [Deutschland \[2008b\]](#)

Durch das konsequente Umsetzen von Barrierefreiheit kann im besten Fall ein Alleinstellungsmerkmal erreicht werden aber zumindest ein enormes Kundenpotential freigesetzt werden. Aufgrund der demographischen Entwicklung ist davon auszugehen, dass die Zielgruppe der altersbedingt auf Hilfestellung angewiesenen Nutzer weiter wachsen wird. In Deutschland beispielsweise werden 2050 über 40 Prozent (derzeit ein Drittel) des privaten Konsums von den über 60 Jährigen getätigt¹. Ähnlich wird die Lage auch für Europa eingeschätzt².

Außerdem werden durch bessere Auffindbarkeit der Inhalte durch Suchmaschinen mehr potentielle Interessenten auf den Internetauftritt geleitet. Dies verstärkt zusätzlich eine Erhöhung der Konversionsrate, die durch bessere Nutzbarkeit erreicht werden kann. Durch gute Nutzbarkeit, die mit der barrierefreien Gestaltung einhergeht, kann sich ein Shop positiv aus dem Angebot hervorheben. Durch Mundpropaganda können so langfristig mehr Nutzer auf das Internetangebot gelangen, was sich durch PR³-Bemühungen noch verstärken lässt.

2.2.2 Gesetzliche

Eine Vorreiter-Stellung⁴ auf dem Gebiet der Gleichstellung behinderter Mitmenschen ist die USA. Mit ihrem bereits 1986 in Kraft getretenen und 1998 für das Internet überarbeitetem Gesetz *Section 508*⁵ setzte die amerikanische Regierung Maßstäbe in der Gestaltung gesetzlich verbindlicher Vorgaben für die Bundesverwaltung zur Gleichstellung behinderter Menschen.

Viele weitere Staaten haben mittlerweile Gesetze erlassen die (oftmals nur für behördliche Angebote) Barrierefreiheit im Internet fordern. Hierzu zählen zum Beispiel Deutschland⁶, Österreich⁷, Schweiz⁸, England⁹, Niederlande¹⁰, Schweden¹¹, Spanien¹²

1 vgl. Pressemitteilung des Auswärtigen Amts vom 17.04.2007 [der Bundesrepublik Deutschland \[2007\]](#)

2 vgl. Pressemitteilung der europäischen Kommission - IP/05/322 [Gemeinschaften \[2005\]](#)

3 Public Relations; Öffentlichkeitsarbeit

4 Vgl. Wikipedia - Barrierefreies Internet [Wikipedia \[2009a\]](#)

5 Beschreibt die Mindestanforderungen an Informationstechnik, Teil des „Workforce Rehabilitation Act“

6 nähere Informatationen unter Punkt 4.2 BITV

7 vgl. Kabinet Nachrichten [kabinet nachrichten \[2006b\]](#)

8 vgl. Richtlinien des Bundes für die Gestaltung von barrierefreien Internetangeboten - 2.1 Prioritäten des WCAG 1.0 [Tina Kohler \[2005\]](#)

9 vgl. Naming and registering websites - The conditions of use for a .gov.uk name, 78 [UK \[2007\]](#)

10 vgl: Votrag Raph de Rooij auf der EAFRA Konferenz [de Rooij \[2009\]](#)

11 vgl: Swedish National Guidelines for Public Sector Websites - Executive Summary [Agency \[2006\]](#)

12 vgl: Gesetzliche Situation auf europäischer Ebene - Spanien [Österreich](#)

und Kanada¹, um nur einige Staaten zu nennen.

Auch die Europäische Union hat eine Vereinbarung getroffen, laut der die Mittelvergabe künftig von der Barrierefreiheit abhängig ist². Außerdem wurde im Rahmen der *eEurope*-Initiative im Dezember 1999 die Einführung der Richtlinien der [WAI](#)³ bis 2002 beschlossen. Dieser Aufforderung wurde im April 2002 mit einer Entschließung nochmals Nachdruck verliehen⁴. Eine ähnliche Aufforderung wurde auch im März 2009 ausgesprochen⁵, was zeigt, dass die Umsetzung noch einige Lücken aufweist.

Diese gesetzlichen Vorgaben gelten bisher meistens nur für staatseigene Institutionen, nur in sehr wenigen Ländern auch für die Privatwirtschaft. Freilich wird sie von den meisten Regierungen auch für diese empfohlen.

2.2.3 Ethische

Das Internet ist von Anfang an im Glauben an freie Rede, freien Zugang und das gegenseitige Helfen entstanden. Um diesen Prinzipien genüge zu tragen, sollte jedes Internetangebot so vielen Nutzern wie möglich zugänglich gemacht werden.

In Verbindung mit der Bedeutung, die das Internet mittlerweile für Informationsbeschaffung, Kommunikation, Unterhaltung und Konsum innehaltet, stellt der Zugang zu diesem Medium einen wichtigen Aspekt der Gleichberechtigung und damit der Menschenwürde dar.

Neben der konsequenten Ausführung etwaiger unternehmerischer Prinzipien kann durch das Anbieten eines barrierefreien Internetauftritts (und dessen Zertifizierung bzw. Prämierung) das öffentliche Ansehen einer Firma oder Institution nachhaltig positiv beeinflusst werden. Dieser Effekt kann durch gezielte [PR](#) weiter ausgebaut und genutzt werden und verstärkt sich unter Umständen durch Mundpropaganda.

1 vgl. Common Look and Feel Standards for the Internet [of Canada \[2007\]](#)

2 vgl. Kabinet Nachrichten Mai 06 - [kabinet nachrichten \[2006a\]](#)

3 Web Accessibility Initiative; treibt die Barrierefreiheit im Internet voran und entwickelt dafür Standards. Sie möchte u. a. ein internationales Forum für Industrie, Forschung, Behindertenverbände, Regierungen und andere Interessierte bieten.

4 vgl. Wikipedia - Barrierefreies Internet, Richtlinien zur Barrierefreiheit von Online-Inhalten [Wikipedia \[2009a\]](#)

5 vgl. Rat der europäischen Union - Mitteilung an die Presse - 2935. Tagung des Rates - Verkehr, Telekommunikation und Energie - Telekommunikation 12345 12345 12345 [der europäischen Union \[2009\]](#)

2.2.4 Technische

Durch die strikte Trennung von Darstellung und Inhalten wird eine erhöhte Wartbarkeit erreicht. Dieser Effekt wird zum Beispiel bei einem Relaunch oder einem Facelift besonders deutlich. Durch diesen Effekt können sich die zusätzliche Kosten für redaktionelle Arbeiten und die Erstanschaffungskosten ausgleichen. Grundsätzlich sind die zusätzlichen Kosten bei der kompletten Entwicklung eines Internetauftritts weniger hoch als bei einer nachträglichen Implementation. Die jeweiligen Kosten hängen zusätzlich stark von der Gründlichkeit ab, mit der die Barrierefreiheit umgesetzt werden soll (bei der WCAG¹ 2.0 zum Beispiel in Level A, AA, AAA definiert).

Durch die Trennung von Darstellung und Inhalt sowie die Einhaltung von Standards können auch die zu übertragenden Daten minimiert werden. Das sorgt für schnellere Ladezeiten beim Nutzer und kann langfristig Bandbreiten-Kosten senken. Oft wird auch angeführt, dass die Internetseite durch die Trennung kompatibler zu verschiedenen Ausgabemedien wird. Richtig ist, dass zumindest eine einfache Anpassung an diese möglich wird und die Anwendung generell besser auf zukünftige Technologien vorbereitet ist.

Das Einhalten von gängigen Standards und das Bereitstellen von validem Quellcode (wie von den Richtlinien gefordert) erleichtert außerdem Suchmaschinen die Erfassung des Inhalts und die Indexierung des gesamten Internetauftritts. Das Anbieten von alternativen Darstellungsformen bei Inhalten, die nicht zugänglich sind (wie Flash, Bilder und unter Umständen Java-Script), hilft ebenfalls, diese zu indexieren.

Mit diesen Ausführungen sollen die theoretischen Grundlagen abgeschlossen sein, so dass im nachfolgenden Kapitel die technischen Grundlagen erarbeitet werden können.

1 Web Content Accessability Guidelines

KAPITEL 3

Technische Grundlagen

3.1 Unterstützungstechnologie

Um Menschen mit Behinderungen das Arbeiten mit Computern und damit auch Browsern zu ermöglichen, gibt es eine Vielzahl von Hilfsmitteln. Für die Software und im Besonderen die Hardware entstehen normalerweise hohe Anschaffungskosten. In Deutschland werden diese je nach Art der Behinderung und des Hilfsmittels normalerweise von den Krankenkassen oder Ämtern übernommen¹.

Nachfolgend werden die wichtigsten Hilfsmittel zur Nutzung von Internetseiten vorgestellt. Dies umfasst Software wie Screenreader oder Bildschirmschriften und Hardware. Wenn es für das Hilfsmittel relevant ist, wird dargelegt in wiefern [WAI ARIA](#)² unterstützt wird. [ARIA](#) ist eine technische Spezifikation zur Erhöhung der Barrierefreiheit bei Internetanwendungen. Sie wird unter Punkt [4.4](#) genauer vorgestellt.

3.1.1 Screenreader

Ein Screenreader bzw. Bildschirmleseprogramm ist ein Programm, das blinden oder sehbehinderten Menschen, aber auch Menschen mit einer kognitiven Behinderung, die Möglichkeit geben soll, das Betriebssystem und andere Programme zu bedienen. Dazu bietet es eine alternative Benutzerschnittstelle zur grafischen Benutzeroberfläche, meist in Form einer akustischen Sprachsynthese oder in einer taktilen Form als Braillezeile. Screenreader sind teilweise bereits vorinstalliert werden aber für gewöhnlich installiert. Mittlerweile sind auch mobile Screenreader auf dem Markt, die teilweise auch ohne Installation betrieben werden können.

Die Funktionsweise mit Internetbrowsern soll an dieser Stelle näher erläutert werden. Normalerweise meldet der Browser bestimmte Inhalte, Eigenschaften oder Funktionen

1 vgl. Webseite INCOBS [für Blinde und Sehbehinderte](#) [2009b] und folgend

2 Accessible Rich Internet Applications; mehr Informationen siehe Punkt [4.4](#)

einer Webseite weiter an das OS¹. Das OS wiederum leitet beispielsweise die Änderungen im Inhalt weiter an die AT, die dem Nutzer Rückmeldung gibt. Somit gibt es verschiedene Ebenen, die einer reibungslosen Kommunikation im Weg stehen können.

Neben den verschiedenen Browsern und Screenreader-Produkten gibt es je nach Plattform verschiedene Accessability APIs. Auf Windows-Systemen sind die ältere MSAA² und die weiterentwickelte UIA³ – die wesentlich mehr Eigenschaften von ARIA unterstützt⁴, aber erst ab Windows Vista nutzbar ist – verfügbar. Außerdem wurde von IBM⁵ eine Open-Source Alternative namens IAccessible2 entwickelt, die die MSAA erweitern soll. Firefox ist im Moment dabei, eine vollständige Implementation zu verwirklichen⁶. Dies verschafft Firefox bei der Kombination von bestimmten OS und AT einen Vorteil gegenüber anderen Browsern wie dem MS⁷ Internet Explorer, da dieser IAccessible2 nicht unterstützt und dessen UIA zumindest zur Zeit weniger Eigenschaften abbilden kann. Daneben gibt es auf Linux-Systemen die AT-SPI⁸, ein Macintosh eigenes System (Mac OS X Accessibility Protocol⁹) und das plattformunabhängige, javabasierte Access Bridge.

Screenreader legen sich für gewöhnlich einen Virtuellen Speicher (virtual Buffer) an, mit dem der Nutzer interagieren kann. Bei Änderungen im DOM zum Beispiel durch AJAX muss dieser Buffer aktualisiert werden, um diese Änderungen abzubilden. Um zu verstehen, wann die jeweilige AT den Buffer aktualisiert, muss man sich vor Augen führen, dass ein Screenreader für gewöhnlich zwei Modi kennt: einen Read-Modus (JAWS¹⁰ nennt diesen Virtual PC Cursor und Window Eyes Browse Mode) und einen Form-Modus.

Im Read-Modus kann der Nutzer über die Seite und deren Elemente navigieren, also zum Beispiel durch Überschriften und Absätze. Im Form-Modus kann der Nutzer hauptsächlich durch Eingabeelemente navigieren und mit diesen interagieren, was eine Aktualisierung des Buffers auslöst¹¹.

Ältere Versionen von JAWS (inklusive Version 7.1) und anderen Screenreadern aktualisieren den Virtuellen Speicher nicht automatisch, selbst wenn die entsprechenden

1 Operating System; Betriebssystem

2 Microsoft Active Accessability; Schnittstelle des Betriebssystems zur Bereitstellung von behindertengerechten Benutzeroberflächen

3 UI Automation; Weiterentwicklung der MSAA

4 vgl. Comparison of ARIA roles exposed via MSAA and UI Automation in IE8 Faulkner [2009a] und vgl. Mapping ARIA Roles, States, and Properties to UI Automation Corporation [2009a]

5 International Business Machines Corporation

6 vgl. Bugzilla Team [2009c]

7 Microsoft

8 Assistive Technology Service Provider Interface; Schnittstelle des Betriebssystems zur Bereitstellung von behindertengerechten Benutzeroberflächen

9 vgl. WAI-ARIA 1.0 User Agent Implementation Guide - 3.3 WAI [2009e]

10 Job Access with Speech; Screenreader Software

11 vgl. WEB 2.0: Blind to an Accessible New World Joshua Hailpern [2009]

[ARIA](#) Eigenschaften genutzt werden¹. Aufgrund dessen sollte der Screenreader dazu gezwungen werden, den Buffer zu aktualisieren². Das kann zum Beispiel durch die Veränderung eines Formfeldes erreicht werden³.

Mittlerweile greifen neuere Versionen der verschiedenen Screenreader auch direkt auf den [DOM](#) zu⁴ und können somit die falschen oder nicht implementierten Eigenschaften umgehen. Da aber die meisten kommerziellen Screenreader sehr teuer sind, kann man davon ausgehen, dass viele Nutzer noch ältere Versionen im Betrieb haben.

Leider sind weder Statistiken zur Verbreitung noch zur jeweiligen Version der Software erhältlich, wie es bei Browsern seit langem der Fall ist. Einzige Informationsquelle sind oftmals eine Vielzahl von Blogs und Foren, in denen Entwickler und Betroffene ihre Erfahrungen austauschen.

[JAWS](#) gilt als Marktführer, was eine in Amerika durchgeführte Umfrage bestätigt⁵. Allerdings lassen sich hieraus kaum Rückschlüsse für Deutschland oder Europa ziehen, da in Europa generell eine höhere Akzeptanz für freie Software herrscht.

Im Folgenden soll ein Überblick über die wichtigsten Screenreader und ihre Unterstützung gegeben werden. Hierbei werden vor allem aktuelle Versionen berücksichtigt. Erwähnt, aber nicht berücksichtigt, sei hier, dass es auch Screenreader speziell für mobile Endgeräte und für ausländische Schriftsysteme wie das Japanische gibt.

Vorwegnehmend sei gesagt, dass es schwierig ist, eine abschließende Aussage zum Thema Screenreader zu treffen, da bisher wenig verlässliche Informationen dazu dokumentiert sind. Aufgrund der vielen verschiedenen Faktoren die eine echte Nutzbarkeit erst möglich machen (Betriebssystem, Browser und Hilfsmittel), muss angeführt werden das die Unterstützung noch nicht befriedigend ist. Grundsätzlich kann man sagen, dass Firefox und der neuste [MS](#) Internet Explorer recht gut unterstützt werden. Die Unterstützung von Opera und Safari ist weniger ausgereift.

Der Versuch einer übersichtlichen Darstellung von funktionsfähigen Kombinationen von Browsern und Screenreadern wird bei CodeTalks⁶ unternommen.

1 vgl. The Virtual Buffer [Gez Lemon](#) [2007]

2 vgl. WEB 2.0: Blind to an Accessible New World - 7.3 Synchronizing with Virtual Buffers [Joshua Hailpern](#) [2009]

3 vgl. The Virtual Buffer [Gez Lemon](#) [2007]

4 vgl. Comparison of ARIA roles exposed via MSAA and UI Automation in IE8 [Faulkner](#) [2009a]

5 [WebAIM](#) [2009]

6 vgl. Set of ARIA Test Cases [Community](#) [2009]

NVDA

NVDA¹ ist ein freier und quelloffener Screenreader unter [GPL²](#) 2 Lizenz und läuft offiziell unter [MS Windows](#) XP 32-Bit oder höher. Lauffähigkeit ist auch unter MS Windows 2000 und auf 64-Bit Systemen möglich, wird aber nicht unterstützt.

NVDA unterstützt verschiedene Accessability APIs wie die [MSAA](#), die neuere [UIA](#) oder die Java Access Bridge und arbeitet mit der freien Sprachsynthese-Software eSpeak oder [SAPI³](#)⁴ und [SAPI5](#) von [MS Windows](#), um multilinguale Sprachausgabe zu ermöglichen. Seit der aktuellen Version 0.6p3 ist die Verwendung einer Braillezeile möglich⁴.

Trotz der Unterstützung des [MS Internet Explorers](#) wird für die bestmögliche Nutzbarkeit die Verwendung der aktuellen, stabilen Firefox Version (3.0.X) empfohlen⁵. Um alle Funktionen nutzen zu können, sollte die neusten Beta-Version 3.X.X (Shiretoko, Entwicklervorschauversion von Firefox 3.5)⁶ verwendet werden.

NVDA wurde ursprünglich privat entwickelt, wird mittlerweile aber von der Mozilla Foundation finanziell unterstützt⁷, um diesen freien Screenreader für Firefox zu optimieren. Seitdem sind viele Funktionen wie zum Beispiel die partielle [ARIA](#) Unterstützung⁸ hinzugekommen. Trotz allem sind auch hier noch nicht alle Funktionen und Eigenschaften der [ARIA](#) perfekt abgebildet. Noch nicht vollständig ist die Implementierung von *Live-Regions*, von *Landmarks*, von *Drag-and-Drop* und einigen allgemeinen Verbesserungen sowie der Unterstützung für eingebettetes Flash⁹.

NVDA kann direkt auf den [DOM](#) zugreifen¹⁰ und umgeht so Probleme mit unzureichenden Accessability APIs.

JAWS

JAWS (aktuell JAWS 10.0.1151) ist ein kommerzielles Screenreader-Programm der Firma Freedom Scientific, das multilinguale Sprachausgaben und Braillezeilen unterstützt. Das Programm funktioniert unter [MS Windows](#) XP und Vista, die Professional Version

1 Nonvisual Desktop Access; Screenreader Software

2 General Public License; eine von der Free Software Foundation herausgegebene Lizenz mit Copyleft für die Lizenzierung freier Software.

3 Speech Application Programming Interface; Schnittstelle zur Anbindung von Bibliotheken zur Sprachsynthese und Spracherkennung unter dem Betriebssystem MS Windows.

4 vgl. NVDA Blog - [Teh](#) [2009b]

5 vgl. NVDA Blog [Teh](#) [2009a]

6 vgl. Blog von Marco Zehe [Zehe](#) [2009]

7 vgl. NVDA Blog [Team](#) [2009d]

8 vgl. NVDA Blog [Teh](#) [2009a]

9 vgl. NVDA Wiki [Team](#) [2009d]

10 vgl. Comparison of ARIA roles exposed via MSAA and UI Automation in IE8 [Faulkner](#) [2009a]

auch mit 64-Bit Systemen¹.

Aufgrund des hohen Preises (859 US Dollar² in der Standard Version) ist davon auszugehen, dass einige Nutzer mit veralteten Versionen arbeiten, die noch wenig ARIA-Unterstützung implementiert haben.

JAWS ist für seine Funktionsfähigkeit mit vielen Programmen bekannt und hat einen hohen Verbreitungsgrad. Es gilt als Marktführer, allerdings sind hierzu keinerlei verlässliche Quellen verfügbar.

JAWS unterstützt viele ARIA Eigenschaften (nach Version 7.1 auch einen korrekten Umgang mit dem Virtuellen Speicher) in Verbindung mit Firefox 3 und IE 8³. Ab Version 10 unterstützt die Software auch *Live-Regions*⁴.

Window-Eyes

Windows-Eyes (aktuell 7.02) ist ein von der Firma GW Micro, Inc. entwickelter multilingualer Screenreader zum Preis von 895 US-Dollar⁵.

Die Software arbeitet auf allen 32-Bit basierenden MS Windows Systemen inklusive Vista und kann Sprachausgaben sowie Braillezeilen-Ausgabe bereitstellen.

Die Software unterstützt Internet Explorer und Firefox, arbeitet aber nur mit der veralteten MSAA zusammen⁶. Landmarks zum Beispiel werden unvollständig abgebildet⁷.

HAL

HAL (oder die Version mit integrierter Bildschirmlupe namens Supernova) ist ein Screenreader mit Braille-Unterstützung der Firma Dolphin Computer Access Ltd. zum Preis von ungefähr 795 US-Dollar⁸.

Die aktuelle Version wird von allen 32 und 64-Bit Versionen von XP bis Vista unterstützt.

HAL unterstützt Firefox und Internet Explorer.

1 JAWS Webseite Freedom Scientific [2009a]

2 JAWS Webseite Freedom Scientific [2009a]

3 vgl. JAWS Internetseite Freedom Scientific [2009b]

4 vgl. JAWS Internetseite Freedom Scientific [2009b]

5 Preis laut GW Micro [2009]

6 GW Micro Webseite GW Micro [2009]

7 vgl. Paciello Group Blog Faulkner [2009b]

8 Preis laut Ltd [2009]

Orca

Für Linux-Distributionen gibt es für die Arbeitsumgebung GNOME den Open-Source-Screenreader Orca. Die Entwicklung wird vom Accessibility Program Office of Sun Microsystems, Inc. geleitet und von der Community unterstützt. Orca nutzt die Daten aus der [AT-SPI API](#), die vom GNOME Accessibility Project entwickelt wird. Orca beinhaltet eine Kombination aus Sprachausgabe, Braillemodul und Bildschirmlupe.

Die aktuelle Version (laut Repository Version 2.27.3) ist abhängig von der jeweiligen Linux-Distribution, da Orca seit GNOME 2.16 mitgeliefert wird.

Orca bietet eine gute Unterstützung von Firefox 3¹, bei der bereits einige [ARIA](#) Eigenschaften und Funktionen (teilweise unvollständig) unterstützt werden. Beispiele wären die Unterstützung von Rollen ² (seit Version 2.20) und die gute Unterstützung von *Live-Regions*³

FireVox

Diese freie und quelloffene Software stellt eine Ausnahme dar, da sie direkt im Firefox arbeitet und somit auch nur für diesen Browser Hilfestellung anbietet. Diese Funktionsweise bietet aber auch den Vorteil, dass FireVox direkt im Browser arbeitet und somit gut mit dynamischen Internetseiten funktioniert. Die Software ist unter [GPL](#) lizenziert.

Im Prinzip greift FireVox als Erweiterung zum Browser nur auf eine Sammlung von Java-Script-Funktionen namens [CLC-4-TTS](#)⁴ zurück. Diese Bibliothek gibt es dann an eine der Sprachausgabe-[APIs](#) weiter. Diese arbeitet auf allen Plattformen, da sie verschiedene Sprachausgabe-Standards ([APIs](#)) unterstützt, zum Beispiel [MS SAPI5](#) für Windows, Apple's [TTS](#)⁵ und FreeTTS auf Java Basis⁶. Braillezeilen werden nicht unterstützt.

FireVox unterstützt einige [ARIA](#)-Funktionen wie die *Live-Regions*⁷, Rollen und andere Eigenschaften⁸.

1 vgl. Orca Internetseite [Project](#) [2009a]

2 vgl. Orca Internetseite [Project](#) [2009b]

3 vgl. Orca Internetseite [Project](#) [2009c]

4 Core Library Components for Text-To-Speech

5 Text-To-Speech

6 vgl. FireVox Projektseite [Team](#) [2009b]

7 vgl. FireVox Projektseite [Team](#) [2009a]

8 vgl. FireVox Projektseite - 5. Changes [Chen](#) [2009]

VoiceOver

Apple liefert für sein Macintosh-Betriebssystem seit Version 10.4 eine eingebaute, multilinguale Screenreader-Software mit, die neben Sprachausgabe auch Braillezeile unterstützt.

Diese Software funktioniert mit Apples eigenem Browser Safari und mit Opera¹. Allerdings ist die Unterstützung von [ARIA](#) erst innerhalb des Safari Version 4 vorgesehen. Die letzte Beta wies noch einige Lücken in der Implementation².

Auch Opera unterstützt erst seit Version 9.27 einige wenige [ARIA](#)-Funktionen³, zu denen bis zur der aktuellen Version 9.64 keine hinzukamen.

3.1.2 Bildschirmlupen

Softwaregestützte Bildschirmlupen vergrößern einen Ausschnitt bzw. den ganzen Bildschirminhalt um einen einstellbaren Faktor. Für gewöhnlich kann nicht nur vergrößert werden, sondern auch der Kontrast und die Farbdarstellung verändert werden. Außerdem wird der nicht sichtbare Bereich überwacht. Stark vergrößerter Text kann zur besseren Darstellung mit Anti-Aliasing (Kantenglättung) dargestellt werden. Oft können Bildschirmlupen mit Screenreadern verbunden werden oder sind in diese integriert. Teilweise werden auch Bildschirmtastaturen mitgeliefert.

Die Kosten können je nach Ausführung von 500 Euro bis über 1700 Euro reichen.

Es gibt eine Vielzahl von Softwareanbietern im kommerziellen und kostenlosen, quelloffenen Bereich. Da sie keinen direkten Einfluss auf die Nutzbarkeit einer Webseite haben, werden hier nur einige wichtige erwähnt: MAGic von Freedom Scientific (aktuelle Version 10.0.1154), ZoomText von Ai Squared (aktuelle Version 9.1), Lunar von Dolphin (aktuelle Version 11.01), Galileo von Baum (aktuelle Version 4.5), Cobra von Baum (aktuelle Version 8.1; Nachfolger des bekannten Blindows 4.12) sowie Virtual Magnifying Glass von Harri Pyy, Chris O'Donnell und Felipe Monteiro de Carvalho (aktuelle Version 3.3.2) und Kmagnifier von Sarang Lakare (aktuelle Version 8.2).⁴

3.1.3 Textbasierte Browser

Textbasierte Browser reduzieren die Anzeige von Internetseiten auf den reinen Inhalt und verzichten auf alle gestalterischen Elemente sowie Java-Script und eingebettete

1 vgl. Apple Internetseite [Inc. \[2009\]](#)

2 vgl. Webseite von Matt Machell [Machell \[2009\]](#) und Webseite von Roger Johansson [Johansson \[2009\]](#)

3 vgl. Opera Internetseite [ASA \[2009\]](#)

4 Die Informationen dieses Abschnitts stammen aus Wikipedia ([Wikipedia \[2009c\]](#)) und der Webseite von INCOBS ([für Blinde und Sehbehinderte \[2009c\]](#) und folgend).

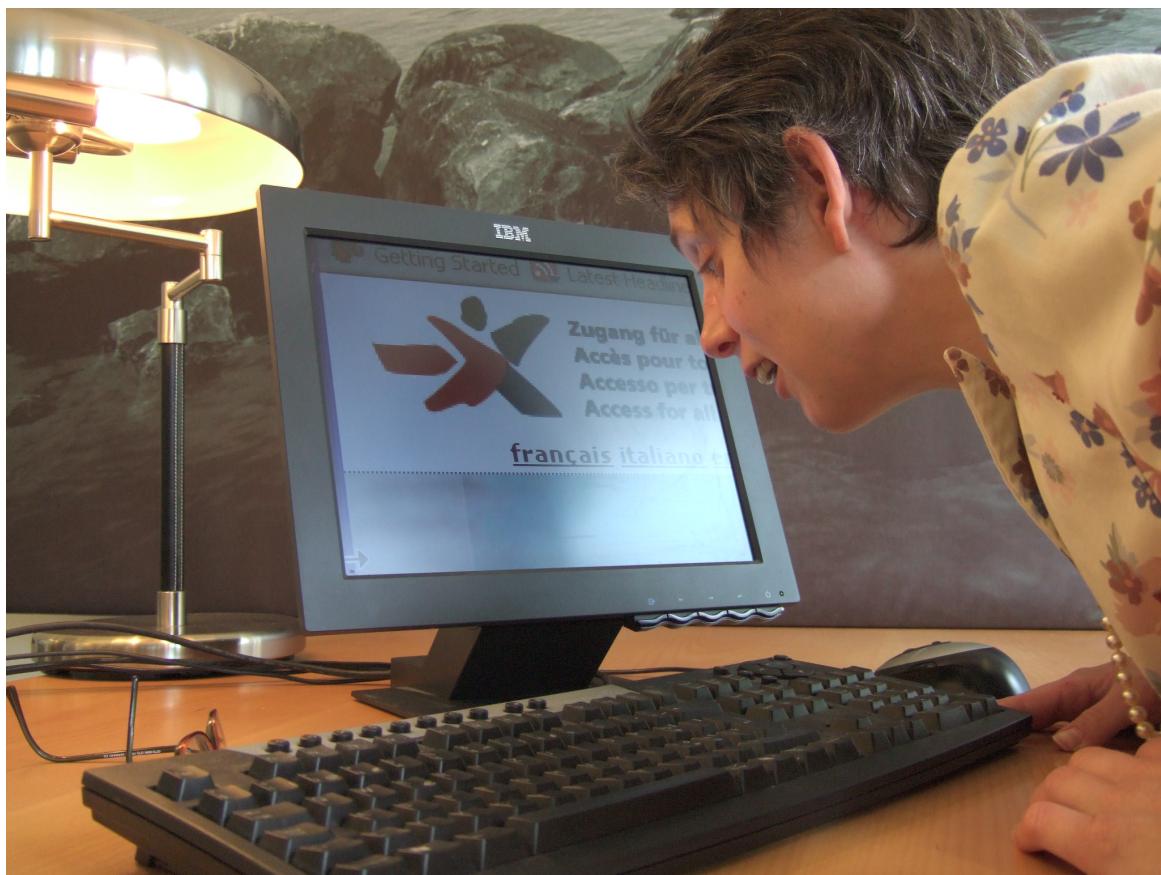


Abbildung 3.1: Nutzung einer softwaregestützten Bildschirmlupe

Inhalte wie Java, Flash oder Silverlight.

Diese Art Browser wurde ursprünglich für wissenschaftliche Zwecke entwickelt, wird aber auch von Menschen mit Behinderung genutzt. Aufgrund der fehlenden Darstellung können Informationen, die auf rein visueller Darstellung beruhen, nicht erfasst werden.

Wichtige Vertreter dieser meist unter freier Lizenz stehenden Software sind Lynx, w3m, Links sowie dessen Weiterentwicklungen Elinks und Links2.

Die Browser unterscheiden sich in ihrem Funktionsumfang. Einige sind zum Beispiel nicht in der Lage, *iframes* oder Tabellen anzuzeigen. Andere hingegen haben ihren Funktionsumfang auch auf die Anzeige von Bildern und Farben, das Anlegen von Lesezeichen und ähnliches erweitert.¹

¹ Die Informationen dieses Abschnitts stammen aus Wikipedia (Abschnitt Textbasierte Browser - [Wikipedia \[2009d\]](#)) und der Webseite Tec Channel (Seite 4 [Rieske \[2009\]](#)).

Hauptseite

aus Wikipedia, der freien Enzyklopädie

Wechseln zu: [Navigation](#), [Suche](#)

Willkommen bei Wikipedia

Die Wikipedia ist ein Projekt zum Aufbau einer freien Enzyklopädie in mehr als 200 Sprachen. Jeder kann mit seinem Wissen beitragen. Seit Mai 2001 entstanden so 419.349 Artikel in deutscher Sprache. Gute Autorinnen und Autoren sind stets willkommen.



Geographie



Geschichte



Gesellschaft



Kunst



Religion



Sport



Technik



Wissenschaft

[Artikel nach Themen](#) · [Alphabetischer Index](#) · [Artikel nach Kategorien](#)

[Kontakt](#) · [Presse](#) · [Statistik](#) · [Andere Sprachen](#) · [Andere Ausgaben](#)

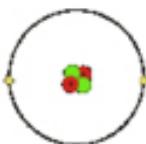
Wikipedia aktuell

Die Frühbuchung für [Wikimania 2006](#), die 2. internationale [Wikimedia](#)-Konferenz, läuft bis zum 9. Juli. Es stehen [Stipendien](#) zur Verfügung; Anträge hierauf sind bis zum 28. Juni einzureichen.

In den Nachrichten

- [Braunbär JJ1](#) · [Fußball-WM](#) · [Union islamischer Gerichte](#)

Artikel des Tages



[Helium](#) (von altgriechisch ἥλιος hélios = Sonne) ist ein farbloses, geruchloses, geschmackloses, ungiftiges, einatomiges chemisches Element. Helium gehört zur Gruppe der Edelgase, seine Ordnungszahl

- Die [Internationale Rotkreuz- und Rothalbmond-Bewegung](#) akzeptierte mit großer Mehrheit den Roten Kristall als drittes Kennzeichen der Bewegung.
- In [Klagenfurt](#) begann der Wettbewerb um den 30. [Ingeborg-Bachmann-Preis](#), eine der wichtigsten Auszeichnungen für deutschsprachige Autoren.

[◀ | Viewing <Hauptseite - Wikipedia>](#)



Abbildung 3.2: Screenshot des textbasierten Browsers w3m

3.1.4 Braillezeile

Braillzeilen erzeugen aus schriftlichen Daten eine Ausgabe in Blindenschrift (Brailleschrift). Dank eines piezoelektrischen Effekts speziell gezogener Kristalle, die auf elektrische Spannung reagieren, werden Erhebungen erzeugt, die der Nutzer mit den Fingerkuppen abtasten kann.

Braillezeilen gibt es in verschiedenen Ausführungen mit 20, 40 oder 80 darstellbaren Zeichen, welche der Menge der darstellbaren Zeichen in einer Zeile entsprechen. Für gewöhnlich besitzen Braillezeilen weitere Steuertasten, mit denen der dargestellte Ausschnitt verschoben oder auch der Screenreader bedient werden kann.¹

Die Geräte werden mit und ohne eingebaute Tastatur hergestellt und sind mittlerweile auch portabel verfügbar. Sie werden für gewöhnlich per [USB](#)² angeschlossen, es sind aber auch neuere Geräte mit Bluetooth verfügbar. Die Geräte kosten zwischen ca. 5000 Euro bis über 10.000 Euro.

Braillezeilen werden meist durch Screenreader mit Daten versorgt und haben im Gegensatz zur Sprachausgabe den Vorteil, dass sie die Worte exakt wiedergeben. Die meisten Screenreader unterstützen eine breite Palette von Braillezeilen. Hersteller von Braillezeilen sind zum Beispiel Baum, Papenmeier, Optelec Tieman, Freedom Scientific, B&M, Hedo, Deiniger, fluSoft, VERTICAL-Technologie, Handy Tech und viele weitere. Es gibt also eine große Menge an Anbietern. Wichtig ist aber vor allem die gute Unterstützung des Screenreaders.

Da Braillezeilen meistens über Screenreader angesteuert werden und somit ein reines Ausgabegerät darstellen sind sie für diese Arbeit nicht relevant.

3.1.5 Großschrift-Tastaturen

Großschrift-Tastaturen entsprechen einer Standard-Tastatur, haben aber eine deutlich größere Beschriftung, unterschiedliche Gehäuse und Schriftfarben und teilweise fühlbare Markierungen. Die Geräte sind normalerweise für Desktop-Computer entworfen, es werden aber auch Tastaturen für Laptops angeboten. Neben diesen modifizierten Standard-Tastaturen gibt es auch Spezial Tastaturen zur Einhandbedienung, Großfeld-Tastaturen sowie Kompakt-Tastaturen und Mini-Tastaturen.

Wichtige Anbietern sind Weißenstein, Gorlo & Todt, Deininger, INCAP, CareTec, ergoMotix und Reha Vista.

1 Die Informationen zu diesem Abschnitt stammen aus Wikipedia ([Wikipedia \[2009b\]](#)), der Webseite von INCOBS ([für Blinde und Sehbehinderte \[2009a\]](#) und folgend) und der Webseite von Matthias Haenel ([Haenel \[2009\]](#))

2 Universal Serial Bus



Abbildung 3.3: Tastatur mit Braillezeile der Firma EuroBraille

3.1.6 Mausersatz

Der Vollständigkeit halber erwähnt seien hier die analog zu Großschrift- und Spezial-Tastaturen erhältlichen Mausersatz-Geräte. Die Geräte gibt es für unterschiedliche Behinderungen in Form eines Joysticks, als empfindliche Touchpads, als eine Art Pfeiltasten-Ersatz oder als mit dem Mund bzw. Kopf steuerbare Eingabegeräte (wie kopfgetragene Maus, Headtracker, *Sip-and-Puff*-Systeme) .

3.2 jQuery

jQuery ist ein ursprünglich von John Reisig entwickeltes, freies Java-Script-Framework, das komfortable Funktionen zur Selektion von DOM-Elementen und deren Manipulation, für erweiterte Events, Hilfsfunktionen (zum Beispiel Iteration oder Array Funktionen), Effekte und Animationen sowie AJAX Funktionen bereitstellt. Es existieren einige offizielle Erweiterungen – die jQuery UI Widgets¹ – und eine Vielzahl von frei

1 <http://ui.jquery.com>

verfügbarer Plugins¹. Auch das Schreiben eigener Erweiterungen ist durch zahlreiche Anleitungen² und Dokumentationen³ einfach zu bewerkstelligen.

Das Framework wurde 2006 auf dem BarCamp⁴ in New York vorgestellt und ist unter den Lizenzen [MIT](#)⁵ und [GPL](#) verfügbar. Es kann somit auch für kommerzielle Projekte sehr gut verwendet werden. jQuery steht unter ständiger Weiterentwicklung.

In dieser Arbeit wird jQuery in erster Linie aus dem Grund eingesetzt, dass es bei Namics das Standard Java-Script Framework ist.

3.2.1 Barrierefreiheit

Bis jetzt wurde Barrierefreiheit in jQuery nur innerhalb einiger Widgets umgesetzt⁶, es sollen aber bis Version 1.8 zumindest Tickets im Bug-Tracker⁷ erstellt werden⁸.

Es existiert ein Plugin für den jQuery Core namens *jARIA*, das jQuery um einige Methoden zur [ARIA](#)-Implementation erweitert⁹. Die Entwicklung wurde aber eingestellt, da diese Funktionalität mittlerweile im jQuery Core enthalten ist¹⁰.

Bei dem zur Umsetzung angedachten Widget *Accordion* wurde bereits Barrierefreiheit umgesetzt und unter anderem [ARIA](#)-Eigenschaften implementiert¹¹. Somit wird eine Umsetzung überflüssig. Stattdessen sollen sortierbare Tabellen umgesetzt werden, da diese auch oft eingesetzt werden.

Für die angedachte Entwicklung einer Reiteranwendung ist festzustellen, dass diese bereits mit dem jQuery UI *Tabs* Widget umgesetzt ist. Da diese Implementierung aber nicht barrierefrei ist, soll das Widget erweitert werden.

1 <http://plugins.jquery.com>

2 unvollständige Sammlung auf <http://docs.jquery.com/Tutorials>

3 http://docs.jquery.com/Main_Page

4 Ein BarCamp ist eine offene, partizipative Unkonferenz, deren Ablauf und Inhalte von den Teilnehmern bestimmt werden.

5 MIT-Lizenz; eine aus dem Massachusetts Institute of Technology stammende Lizenz für die Benutzung verschiedener Arten von Computersoftware. Sie erlaubt die Wiederverwendung der unter ihr stehenden Software sowohl für Software, deren Quelltext frei einsehbar ist, als auch für Software, deren Quelltext nicht frei einsehbar ist.

6 vgl. Blog der Paciello Group [Faulkner \[2009c\]](#)

7 Fallbearbeitungssysteme für die Softwareentwicklung.

8 vgl. <http://docs.jquery.com/UI/ScheduledGoals>

9 vgl. Webseite von Chris Hoffmann [Hoffman \[2009\]](#)

10 vgl. Mailingliste jQuery Accessibility - Overview ARIA support [Link](#)

11 vgl. <http://jqueryui.com/demos/accordion/>

3.2.2 Alternativen

Es existieren viele verschiedene Java-Script-Frameworks mit unterschiedlichen Vor- und Nachteilen. Sie basieren alle auf dem ersten derartigen JS Framework *Prototype* bzw. *script.aculo.us*, welches mittlerweile eher als veraltet und zu wenig komfortabel gilt.

Im nachfolgenden soll eine Übersicht über die Frameworks mit der höchsten Verbreitung gegeben werden. Hierbei soll auch eine grobe Einschätzung¹ der Barrierefreiheit vorgenommen werden. (Stand 1. Juli 2009)

Name	Version	Lizenz	Barrierefreiheit	Anmerkung
Dojo Toolkit	1.3.1		ja ²	
Ext Core	3.0	MIT v3 oder kommerziell	-	keine fertigen Widgets
Ext JS	2.2.1	GPL v3 oder kommerziell	kaum, aber geplant	beinhaltet komplexe RIA
MooTools	1.2.3	MIT	nein	
script.aculo.us	1.8.2	MIT	k.A.	nutzt Prototype
YUI	2.7.0	BSD ³	ja	

Tabelle 3.1: Java-Script Frameworks

Das Kürzel *k.A.* bedeutet, dass keine Angaben gefunden werden konnten und auch beim testen keine besondere Unterstützung (zum Beispiel *ARIA*) gefunden wurde. Es kann davon ausgegangen werden, dass das Framework nicht barrierefrei ist. Die Angabe *ja* bezieht sich auf Teile des Frameworks, da eine komplette Implementation noch in keinem Framework realisiert ist.

Eine gute Übersicht zum Thema findet man im Blog der Parciello Group im Artikel *WAI-ARIA Implementation in JavaScript UI Libraries*⁴.

3.3 Schnittstellen zu Datenübertragung

In dieser Arbeit soll unter anderem eine Formularanwendung erstellt werden, die die eingegebenen Daten an den Server übertragen soll. Um diese Kommunikation zwischen clientseitiger Applikationen und Server zu ermöglichen, war in dieser Diplomarbeit zunächst eine spezifische Lösung für *CQ* von der Firma Day sowie eine universelle Schnittstelle in *XML* angedacht. Nach Rücksprache mit Spezialisten für *CMS*⁵-Systeme

¹ gestützt auf die Angaben der jeweiligen Entwickler

⁴ siehe Faulkner [2009c]

⁵ Content Management System

stellte sich heraus, dass keine spezielle Anpassung für [CQ](#) nötig ist. Wie alle [CMS](#)-Systeme besitzt auch [CQ](#) eine native Unterstützung für mit *POST* gesendete Formular-Daten.

Die per *POST*-Methode, genauer per [HTTP](#)¹ *POST*-Methode, gesendeten Daten werden von der jeweiligen Applikation auf dem Server entgegengenommen, aufbereitet und in der jeweiligen Programmiersprache zur Verfügung gestellt. Im Falle von [CQ](#) ist dies ein Servlet, genauer ein [CQ Application Server](#), der, wie die meisten JAVA basierten Systeme, eine Implementierung des nativen *JAVA Servlet getParameter Interface* zur Vorhaltung der Daten nutzt. Eine solche Implementierung der *JAVA* Spezifikation gibt es in jedem auf *JAVA* basierenden [CMS](#)-System.

Die [CQ](#)-Implementierung, die prinzipiell der des Apache *Tomcat* entspricht, ergänzt das Interface noch um einige [CQ](#) spezifische Methoden.

Da eigentlich alle Systeme – unabhängig von ihrer verwendeten Technologie (wie *JAVA* oder [PHP](#)²) – eine solche *POST*-Schnittstelle zum Empfang und zur Weiterverwendung von *POST*-Daten haben, sollte diese auch genutzt werden. Außerdem müsste eine [XML](#)-Datei ebenfalls über *POST* übermittelt und anschließend – mit einem oftmals nicht nativ vorhandenen [XML](#)-Parser – verarbeitet werden.

Nicht nur die einfachere Verwendung der vorhandenen *POST*-Schnittstelle, sondern auch der nicht unerhebliche Zuwachs der Dateigröße, die eine Erzeugung von [XML](#) in *Java-Script* mit sich bringen würde, spricht gegen eine Implementierung der [XML](#)-Schnittstelle. Aus diesem Grund wird auf eine solche Implementierung, wie sie in der Zielsetzung festgelegt wurde, verzichtet.

1 Hypertext Transfer Protocol

2 Hypertext Preprocessor

KAPITEL 4

Technische Richtlinien

Barrierefreiheit im Internet ist ein ein komplexes Feld, das einiges an Kenntnissen und Einarbeitungszeit verlangt. Neben dem umfangreichen theoretischem Wissen, das zum Verständnis und der korrekten Umsetzung nötig ist, muss eine Vielzahl von Techniken und *Best Practice* erlernt werden. Um bei der Entwicklung von Internetseiten und deren Applikationen effektiv und effizient arbeiten zu können, brauchen die Gestalter (die *Konzepter*) und die Entwickler konkrete Richtlinien, an Hand derer die Umsetzung von stattet geht.

Durch die Bereitstellung von praxistauglichen und, wenn möglich, auch meßbaren Standards wird die Verbreitung von barrierefreien Techniken und deren Implementation in internetnahe Software gefördert.

Außerdem helfen Internationale Standards Entwicklern und Firmen, in anderen Ländern arbeiten zu können und Produkte oder Dienstleistungen zu verkaufen. So werden durch solch internationale Standards zum Beispiel nationale gesetzliche Mindestanforderungen und Richtlinien ohne Weiteres erfüllt. Sie beschleunigen damit die Entwicklung von Software, indem sie Weitergabe und Verwendung vereinfachen.

Die wichtigsten Richtlinien für barrierefreie Internetseiten sind die [WCAG](#) der [WAI](#) Arbeitsgruppe innerhalb des [W3C](#). Die [W3C](#) ist das Gremium zur Standardisierung der das World Wide Web betreffenden Techniken und bestimmt maßgeblich die Entwicklung des Internets.

Daneben gibt es die von der [ISO](#)¹ herausgegebene Reihe *9241 - Ergonomie der Mensch-System Interaktion*, unter der mehrere für das Thema Internet relevante Leitlinien erschienen sind. Die wichtigsten sind *Teil 151: Leitlinien zur Gestaltung von Benutzerschnittstellen für das World Wide* und *Teil 171: Leitlinien für die Zugänglichkeit von Software*.

1 Internationale Organisation für Normung

Neben dieser international verwendbaren bzw. gültigen Richtlinien gibt es einige national gültige Richtlinien. Die meisten westlichen Industrienationen verfügen mittlerweile über in Regierungsauftrag oder von gemeinnützigen Organisationen entwickelte technische Richtlinien oder Initiativen zum Abbau von Barrieren im Internet. Auch haben verschiedene nationale Vereine Auszeichnungen und Tests entwickelt, für die auch Kriterien verfasst wurden. In Deutschland gibt es beispielsweise von Regierungsseite die [BITV](#)¹-Richtlinie und den BIK -Test² sowie verschiedene private Tests und Auszeichnungen wie den *BIENE*³ Award.

Diese Richtlinien und Tests lehnen sich häufig an die Dokumente der [WCAG](#) an oder beschränken sich auf das Übersetzen, Ergänzen bzw. Abändern dieser. Oft wurden Ergänzungen zu lokalen Gegebenheiten wie Symbolik und Sprache sowie kulturellen Besonderheiten eingepflegt und Änderungen in der Anordnung und Wichtigkeit der einzelnen Punkte vorgenommen.

Deutschsprachige Beispiele für solche nationalen Richtlinie wären zum Beispiel Österreich (verweist im Rahmen seines Bundes-Behindertengleichstellungsgesetz auf die [WAI](#) Richtlinien⁴), Schweiz (mit ihren *Richtlinien des Bundes für die Gestaltung von barrierefreien Internetangeboten*, die im Rahmen des seit 1.1.2004 gültigem Behinderungsgleichstellungsgesetz (BehiG) auf die [WCAG](#) 1.0. verweisen⁵) oder Deutschland (mit dem [BITV](#), siehe Punkt 4.2).

Auch andere europäische Länder haben eigene Richtlinien, die sich an den [WCAG](#) 1.0 orientieren und sich meist in der Weiterentwicklung befinden, um der zweiten Version der [WCAG](#) genüge zu tragen. Hierzu zählen England mit der vom [BIS](#)⁶ herausgegebenen PAS-78 Richtlinie und deren Nachfolger BS8878⁷, die Niederlande mit ihren *Web Guidelines* – herausgegeben vom *Ministry of Interior and Kingdom Relations*⁸ – und Schweden mit ihrer allgemeiner formulierten *Swedish National Guidelines for Public Sector Websites*⁹.

1 Barrierefreie Informationstechnik-Verordnung

2 Das BIK (barrierefrei informieren und kommunizieren) Projekt wird vom Bundesministerium für Arbeit und Soziales gefördert und ist ein Gemeinschaftsprojekt deutscher Blinden- und Sehbehindertenverbände und der DIAS GmbH. Es bietet ein Prüfverfahren auf Basis der BITV an.
<http://www.bitvtest.de>

3 BIENE (Barrierefreies Internet eröffnet neue Einsichten)-Award der Aktion Mensch und Stiftung Digitale Chancen, <http://www.biene-award.de>

4 vgl. Web-Accessibility (Internet Zugang für alle) - Bundeskanzleramt Österreich [Österreich](#)

5 vgl. Richtlinien des Bundes für die Gestaltung von barrierefreien Internetangeboten - 2.1 Prioritäten des WCAG 1.0 [Tina Kohler \[2005\]](#)

6 The British Standards Institution

7 vgl: Vortrag Dr. J. Hassell auf der EAFRA Konferenz [Hassell \[2009\]](#)

8 vgl: Vortrag Raph de Rooij auf der EAFRA Konferenz [de Rooij \[2009\]](#)

9 vgl: Swedish National Guidelines for Public Sector Websites - Adoption of Web Accessibility Content Guidelines [Agency \[2006\]](#)

Auch die EU selbst hat bereits mehrfach die [WCAG](#) als Richtlinie empfohlen und gefordert¹. Sie empfiehlt ihren Mitgliedsstaaten, Maßnahmen zur Förderung von Barrierefreiheit zu erlassen und die Anpassung an die [WCAG](#) 2.0 umzusetzen².

Außerhalb Europas wären zum Beispiel Japan mit dem [JIS](#)³ *Web content accessibility guideline* (JIS X 8341-3)⁴, Kanada mit der *Common Look and Feel Standards for the Internet*⁵ und die USA mit ihren *Section 508 Standards* (die Teil der Section 508 Verordnung sind)⁶ zu nennen, welche sich ebenfalls an den [WCAG](#) orientiert haben.

Viele dieser Richtlinien sind auch in den jeweiligen nationalen Gesetzen verankert und damit für Teile der Gesellschaft verbindlich. Weitere Informationen zu diesem Thema sind unter Punkt [2.2.2](#) zu finden.

Zusammenfassend lässt sich sagen, dass so gut wie alle nationalen Richtlinien die [WCAG](#) in ihrer ersten Version adaptiert oder gänzlich übernommen haben und viele bereits eine Angleichung an die neuere Version [WCAG](#) 2.0 vornehmen oder vorgesehen haben. Eine Übersicht über die Bezug nehmenden Richtlinien und ihrer etwaigen Verankerung in Gesetze bietet auch die [W3C](#) selbst an⁷.

Bei der nun folgenden Analyse der verschiedenen Richtlinien stellte sich heraus das die [WCAG](#) 2.0 die am besten taugliche Richtlinie für die Konzeption um Umsetzung des praktischen Teils dieser Arbeit ist. Dies zusammen mit ihrer weltweiten Bedeutung als Basis für die meisten anderen Richtlinien auf diesem Gebiet, macht sie zur grundlegenden Richtlinie für diese Arbeit. Des weiteren werden die beiden relevanten [ISO](#) Richtlinien analysiert um ihre Bedeutung für diese Arbeit zu erörtern. Die [BITV](#) Richtlinie soll zusätzlich als Beispiel einer nationalen Richtlinie untersucht werden, da die vorliegende Arbeit in einem in Deutschland angesiedeltem Unternehmen entsteht und die für Deutschland gültige [BITV](#) somit von besonderer Bedeutung ist.

1 siehe hierzu Punkt [2.2.2](#)

2 vgl. Rat der europäischen Union - Mitteilung an die Presse - 2935. Tagung des Rates - Verkehr, Telekommunikation und Energie - Telekommunikation: Punkt 7 g [der europäischen Union \[2009\]](#)

3 Japan Industrial Standard

4 vgl. JIS Web Content Accessibility Guideline (Englische Übersetzung) [Takayuki \[2004\]](#)

5 vgl. Common Look and Feel Standards for the Internet - Part 2: Standard on the Accessibility, Interoperability and Usability of Web Sites - 1. Compliance with World Wide Web Consortium Priority 1 and Priority 2 checkpoints [of Canada \[2007\]](#)

6 vgl. Section 508 Standards - Subpart B (Technical Standards) - 1194.22 Web-based intranet and internet information and applications [Administration \[1998\]](#) und Side by Side WCAG vs. 508 [Thatcher](#)

7 W3C WAI - Policies Relating to Web Accessibility <http://www.w3.org/WAI/Policy/Overview.html>

4.1 ISO

Die ISO ist eine weltweite Vereinigung von internationalen Normungsorganisationen und umfasst derzeit ca. 150 Länder. Sie erstellt Normen in verschiedenen Bereichen und bildet zusammen mit der Internationalen elektrotechnischen Kommission (IEC) und der Internationalen Fernmeldeunion (ITU) die World Standards Cooperation (WSC). Offiziell am 23. Februar 1947 gegründet und mit Hauptsitz in Genf in der Schweiz stellt die ISO eine der mächtigsten Nicht-Regierungs-Organisationen der Welt dar. Die von ihr erlassenen Regelungen haben weitreichenden Einfluss auf die Wirtschaft und nationale Gesetzgebungen. Die Richtlinien haben aber keine gesetzliche Verbindlichkeit.

Zum Thema Barrierefreiheit im Internet existieren zwei relevante, international gültige Normen der ISO. *Leitlinien von Benutzerschnittstellen für das World Wide Web* und *Leitlinien für die Zugänglichkeit von Software* sind unter der Reihe *9241 - Ergonomie der Mensch-System Interaktion*¹ erschienen.

Die Reichweite der Publikationen im Bereich der Entwicklung von Internetseiten und Applikationen im relevanten Umfeld stellt sich dem Autor als gering dar. Während der Recherche und auf Nachfragen bei verschiedenen Experten im Bereich Barrierefreiheit wurden nie Leitlinien der ISO erwähnt. Dies mag mit der Notwendigkeit zusammenhängen, zum Verständnis der Richtlinie auf mehrere Dokumente zurückgreifen zu müssen, auf die die Richtlinie Bezug nimmt. Auch die nicht unerheblichen Anschaffungskosten dieser Dokumente könnten eine Rolle spielen. Außerdem wird kaum begleitendes Material angeboten und die Leitlinien selbst beinhalten nur Anhaltspunkte, kaum detaillierte, technische Umsetzungsempfehlungen für reale Probleme. Des Weiteren ist die Struktur einem effizienten, praxistauglichen Arbeiten nicht dienlich, auch weil eine Einteilung in Stufen der Barrierefreiheit fehlt. Oftmals wäre es kaum möglich, wirklich alle geforderten Punkte zu erfüllen, um so der jeweiligen Richtlinie zu entsprechen.

Insgesamt beinhalten die Leitlinien daher nur wenig konkrete Ansätze für die praktische Umsetzung dieser Arbeit. Aufgrund der Wichtigkeit der erstellenden Organisation und ihrer guten Anwendbarkeit als Standardwerke für Nutzbarkeit und Zugänglichkeit sollen sie hier dennoch kurz skizziert werden.

4.1.1 ISO 9241-151

Teil 151: Leitlinien zur Gestaltung von Benutzerschnittstellen für das World Wide Web (ISO 9241-151:2008)

¹ ISO 9241-171 Zugänglichkeit von Software wurde dieser Reihe erst im Juli 2006 zugefügt (ehemals ISO/TS 16071)

Diese Europäische Norm wurde am 8. Mai 2008 von der ISO angenommen und ist in drei offiziellen Sprachen (Deutsch, Englisch und Französisch) erhältlich. Die folgenden Erläuterungen beziehen sich auf die deutsche Fassung.

Die Richtlinie wendet sich an Entwickler und Designer von Internetinhalten, Anbieter von Inhalten, Gutachter und Käufer von Softwareprodukten sowie Entwickler von Erstellungswerkzeugen. Sie soll bei Internet-Benutzerschnittstellen auf allen PC-Systemen einschließlich mobiler Endgeräte anwendbar sein.

Diese Leitlinien beschäftigen sich zwar mit Barrierefreiheit, aber nur, insofern sie Bestandteil einer funktionierenden, erfolgreichen Webseite ist. Es werden viele wichtige Aspekte von Barrierefreiheit angesprochen, doch das Dokument ist eher als eine Art Sammlung von Qualitätskriterien für ein Internetangebot zu sehen. Es geht auf die vielfältigen Seiten einer qualitativ hochwertigen Internetseite ein. Das Hauptaugenmerk liegt hier auf guter Nutzbarkeit der Anwendung, weniger auf guter Zugänglichkeit. Die eigentlichen Richtlinien sind in 5 Felder gegliedert, deren wichtigste Punkte hier kurz skizziert werden. Um die Tragweite und Relevanz für diese Arbeit zu analysieren, soll zusätzlich ein Überblick über die für Barrierefreiheit relevanten Regeln gegeben werden.

Grundlegende Gestaltungentscheidungen und -strategien

Hier wird auf verschiedene Aspekte einer funktionierenden, erfolgreichen Webseite eingegangen. Neben Zielgruppenanalyse, Definition von Zielen und Verwendungszweck einer Internetanwendung werden einheitliche, übergreifende Strategien für Internetseiten und Notwendigkeiten wie Kennzeichnung des Anbieters angesprochen.

Die ISO Richtlinie fordert als eines der Ziele bei der Entwicklung von Internetauftritten die möglichst breite Zugänglichkeit und schließt explizit Menschen mit Behinderung in diese Definition ein.

Auch wenn Barrierefreiheit unweigerlich mit der Benutzbarkeit von Internetinhalten und Applikationen zusammenhängt, zielt die Richtlinie nicht darauf ab, umfassend Regeln zur Konzeption barrierefreier Internetseiten aufzustellen. Es wird auf die im nachfolgenden Abschnitt beschriebene ISO Richtlinie 9241-171 und die auf Seite 41 erläuterte WCAG 2.0 verwiesen¹. Letztere befand sich bei der Erstellung der hier erwähnten Richtlinie 9241-151 noch in der Fertigstellung.

1 vgl. DIN EN ISO 9241-151 - Seite 16 für Normierung e. V. [2008a]

Gestaltung des Inhalts

Diese Rubrik befasst sich mit der Eignung des Inhalts für die Zielgruppe sowie dessen Beschaffenheit. Seine Struktur, Vollständigkeit und Granularität wird angemahnt. Des weiteren wird empfohlen, die Struktur, die Darstellung und den Inhalt einer Webseite möglichst unabhängig voneinander zu halten, um eine korrekte Auslieferung auf verschiedenen Medien und Geräten zu gewährleisten.

Barrierefreie Aspekte sind hier die Forderung nach gleichwertigen Textbeschreibungen für nicht-textuelle Objekte und nach Kontrolle des Nutzers über zeitabhängige Medienobjekte. Es wird auf die Möglichkeit von nutzerspezifischen Individualisierungen eingegangen.

Navigation und Suche

In diesem Teil der Leitlinie wurde festgelegt, wie eine Navigation beschaffen sein soll. Fast alle Regeln dieses Abschnitts sind mit Barrierefreiheit und Nutzbarkeit von Applikationen verknüpft.

Es wurde auf allgemein anerkannte Richtlinien wie eine selbstbeschreibende Gestaltung, Anzeige der Position, das Anbieten und Unterstützen von verschiedenen Navigationspfaden und das Minimieren von Navigationsaufwand eingegangen. Aspekte wie die Tiefe und Breite einer Navigationsstruktur sowie die Struktur im allgemeinen werden angesprochen. Die Leitlinie besagt außerdem dass die Navigation und ihre Komponenten in Form, Farbe und Position erkennbar und konsistent sein sollten. Das Navigationskonzept und dessen Umsetzung sollte funktional sein.

Im zweiten Teil dieser Rubrik wird auf die verschiedenen Aspekte einer Suche eingegangen. Dazu zählen die verschiedenen Arten einer Suche und deren Verfügbarkeit sowie die Darstellung und Funktion der Benutzerschnittstelle. Die Suche sollte fehlertolerant sein und die Ergebnisse nach verschiedenen Aspekten klassifizier-, sortier- und filterbar sein. Der Nutzer sollte die Suche selbst einschränken, wiederholen, ändern und verfeinern können.

Darstellung des Inhalts

Dieser Abschnitt der Richtlinien weist auf die visuelle Darstellung von Inhalten und auf deren Aufbereitung im allgemeinen hin. Das beinhaltet zum Beispiel auch deren Kennzeichnung oder die technische Umsetzung.

Es wird zunächst auf verschiedene andere ISO-Richtlinien verwiesen, die sich mit Menüs, Befehlsdialogen, Manipulation und Formularen sowie Entwurfsprinzipien beschäftigen. Viele der Aussagen innerhalb dieser Rubrik haben eine Bedeutung für Barrierefreiheit: Die Leitlinie besagt, dass auf eine konsistente Gestaltung Wert gelegt werden soll, und es wird sehr knapp die Verwendung von Farben erläutert. Hierbei wird auch auf die Notwendigkeit des Auszeichnens von Navigationselementen und Links eingegangen. Des

weiteren wird die Art, der Stil, die Qualität und die Beschaffenheit von Text beschrieben. Die Richtlinien fügen dem hinzu, dass auf aktualisierte Inhalte sowie zeitlich begrenzte Gültigkeit von Inhalten hingewiesen werden sollte. Weiterhin sollen Tastenkürzel für wichtige Verknüpfungen und Interaktionsobjekte vorhanden sein und man weist auf Probleme und Lösungen in Zusammenhang mit der Verwendung von iframes hin. Es wird ein gleichwertiger Ersatz für diese gefordert.

Allgemeine Gestaltungsaspekte

Dieser Teil der Leitlinie behandelt weitere technische Aspekte allgemeiner Natur. Für Barrierefreiheit relevant wäre hieraus die Nutzung allgemein anerkannter Technologien und Standards sowie deren robuste Gestaltung. Die Benutzerschnittstellen sollten unabhängig vom Eingabegerät bedienbar sein. Eingegebettete Medien müssen ebenfalls gebrauchstauglich und zugänglich sein.

Es wird gefordert, dem Nutzer Hilfen anzubieten¹ und Fehlerquellen zu minimieren sowie ggf. eindeutige Fehlermeldungen auszugeben. Bei Nutzung verschiedener Sprachen sollten diese korrekt ausgezeichnet werden.

4.1.2 ISO 9241-171

Teil 171: Leitlinien für die Zugänglichkeit von Software (ISO 9241-171)

Diese Europäische Norm wurde am 28. Juni 2008 von der ISO angenommen und ist in drei offiziellen Sprachen (Deutsch, Englisch und Französisch) erhältlich. Die folgenden Erläuterung bezieht sich auf die Deutsche Fassung.

Die Richtlinie wendet sich an Entwickler und Gestalter von Benutzerschnittstellen, Prüfer und Käufer von Softwareprodukten sowie Entwickler von Erstellungswerkzeugen. Anwendbar ist sie auf jede Software, die Teil von interaktiven Systemen ist.

Da sich die Leitlinie stark an stationäre Software und weniger an Applikationen im Internet richtet, werden verschiedene Aspekte dieses speziellen Mediums außer acht gelassen. Hierzu zählt, dass auf die Datenmenge aufgrund der Bandbreite Rücksicht genommen werden müsste, was Funktionalitäten, wie die oft geforderte Individualisierbarkeit einschränkt. Hier interferieren verschiedene Aspekte der Zugänglichkeit: anpassbare Benutzereinstellungen mit der Datengröße der Anwendung, der *Performance* der Anwendung und der Geräteunabhängigkeit.

Auch sind bestimmte Empfehlungen aufgrund der Beschaffenheit von Internetapplikationen nicht umsetzbar, da sie technisch noch nicht zu realisieren sind oder eine

1 hierzu wird ISO 9241-13 empfohlen

Realisierung nicht wünschenswert erscheint. Viele Punkte der geforderten Funktionen und Eigenschaften werden im Fall von Internetapplikationen auch durch den Browser bzw. das zugrundeliegende Betriebssystem bereitgestellt.

Trotzdem stellt sie eine ausgezeichnete, detaillierte Grundlage an Wissen zur barrierefreien Gestaltung von Applikationen jeglicher Art dar, nur sind diese Informationen eben sehr wenig spezifisch für Internet-Anwendungen. Außerdem fehlen auch bei dieser ISO Richtlinie eindeutige technologische Empfehlungen sowie Beispiele. Aus diesem Grund ist sie zur Sensibilisierung für die Thematik gut geeignet, bietet aber keine konkreten Ansätze für die Ausarbeitung des praktischen Teils dieser Arbeit.

Hier sei angemerkt das sich diese ISO Richtlinie auf die WCAG 2.0 beruft¹.

Die eigentlichen Richtlinien sind in 4 Felder gegliedert, deren für die Entwicklung von Internetapplikationen relevanten Punkte hier kurz skizziert werden.

Allgemeine Richtlinien und Anforderungen

In diesem Abschnitt werden unter anderem Empfehlungen bezüglich der Beschriftung von Elementen getroffen. Namen sollten innerhalb des Kontextes eindeutig, kurz und verständlich gewählt sein und der AT kenntlich gemacht werden.

Damit sich der Nutzer die Software nach seinen Bedürfnissen einrichten kann, werden Funktionen zur Individualisierung gefordert. Das umfasst die visuelle, haptische und akustische Darstellung, von denen nur einige Aspekte für eine Internetanwendung relevant sind. Anpassungen an Einstellungen sollten zu speichern, zu bearbeiten und zu transferieren sein.

Es werden grundsätzliche Anforderungen definiert, wie diejenige, das Arbeitsschritte zu optimieren und rückgängig zu machen sind. Kritische Aktionen sollten einer Bestätigung bedürfen. Kopieren und Einfügen sollten verfügbar sein.

Fehler und Benachrichtigungen müssen einheitlich, klar und verständlich erstellt werden. Der Nutzer soll beim Auffinden und Korrigieren von Fehler unterstützt werden.

Etwiger Unterstützungstechnologie soll das Kommunizieren mit der Applikation möglich gemacht werden.

Eingaben

Benutzerschnittstellen-Elemente müssen visuell erkennbar und leicht auswählbar sein. Sie sollen ihren Status anzeigen, wenn zum Beispiel eine zyklische Funktion angesteuert wird². Bei Verlassen eines Fensters soll der Fokus auf das zuletzt ausgewählte Element

1 vgl. DIN EN ISO 9241-171:2008-10 - 5. Grundsätze der Gestaltung zugänglicher Software; Literatur-nachweis [52] für Normierung e. V. [2008b]

2 Zum Beispiel bei Ein- und Ausblenden eines Elements

gesetzt werden.

Interaktion darf nur explizit durch den Benutzer ausgelöst werden. Dies darf nicht durch Fokussierung von Elementen ausgelöst werden.

Alle Funktionen müssen tastaturbedienbar sein und definierte Schnelltasten müssen kenntlich gemacht werden.

Ausgaben

Im Abschnitt Ausgaben werden in Bezug auf die Darstellung und Beschaffenheit des Inhalts getroffen.

Das umfasst visuelle Aspekte wie Anfälle verursachende Blinkraten, den Einsatz von Farben und das Bereitstellen von speziell für Menschen mit Behinderung entwickelten Farbschemata. Im Allgemeinen sollte die Farbgebung über ausreichend Kontrast verfügen und individualisierbar sein.

Informationen sollten nicht nur rein visuell gekennzeichnet werden (zum Beispiel durch Farbe oder Schriftschnitt), sondern müssen auch programmatisch auslesbar sein. Außerdem sollten Textzeichen nur für die Darstellung von Informationen und nicht zu Präsentationszwecken¹ genutzt werden.

Wenn der Inhalt eine zeitliche Abfolge hat, muss dem Nutzer eine Möglichkeit gegeben werden, diese zu navigieren und zu steuern. Multimediale Inhalte wie Video und Audio müssen mit zugänglichen Alternativen ausgestattet sein. Das können zum Beispiel Untertitel für Video sein.

Bei Audioausgabe sollte die Informationsvermittlung durch Klangmuster statt Tonhöhe erfolgen (zum Beispiel bei Fehlermeldungen) und dem Benutzer eine Lautstärkeregelung angeboten werden. Es werden genaue Angaben zu Frequenzbereichen gemacht.

Fenster innerhalb der Applikation (bei Internetapplikationen könnte man von Pop-Ups ausgehen) sollten in ihrer Größe und Position änderbar sein und Standard Aktionen wie Minimieren, Wiederherstellen, Maximieren und Schließen anbieten.

Mit Änderung der Schriftgröße sollten alle darstellenden Elemente mitskalieren.

Online-Dokumentation und Hilfen zur Verfügung stellen

Hilfen und Dokumentationen sollen verständlich sein und müssen den in anderen Abschnitten erwähnten Regeln entsprechen. Es sollte für spezielle Zugänglichkeitsmerkmale Hilfen und Dokumentationen bereitgestellt werden.

Auch andere Unterstützungsdienste wie Kundendienste müssen den Anforderungen von Menschen mit Behinderung entsprechen.

¹ zum Beispiel ein Rahmen aus Raute-Symbolen.

4.2 BITV

Das [BITV](#) stellt eine Ergänzung zum [BGG](#)¹ dar und soll behinderten Menschen gemäß §3 des [BGG](#) den Zugang zu den betroffenen Internetangeboten ermöglichen bzw. verbessern. Die erste und immer noch aktuelle Version trat am 27. April 2002 in Kraft und basiert im wesentlichen auf den [WCAG](#) 1.0 des [W3C](#) vom 5. Mai 1999².

Die [BITV](#) entstand mit Hilfe von Experten aus den Behindertenverbänden auf Referentenebene innerhalb eines nicht-öffentlichen Arbeitskreises und entsprach den damaligen Forderungen der Verbände in ihrem Eckpunktepapier³. Zuvor wurden die zugrunde liegenden Anforderungen der [WCAG](#) 1.0 von der Bundesarbeitsgemeinschaft *Hilfe für Behinderte* getestet und Zwischenergebnisse der Ausarbeitung der [BITV](#) wurden Verbänden, Wirtschafts- und Wissenschaftsvertretern vorgelegt.

Die Verordnung ist bindend für alle öffentlich zugänglichen Angebote von Behörden der Bundesverwaltung, nicht aber für Internetangebote der Bundesländer. Alle Angebote, die nach Inkrafttreten neu gestaltet oder in wesentlichen Bestandteilen oder größerem Umfang verändert oder angepasst werden, sind gemäß dieser Verordnung zu erstellen. Spätestens bis zum 31. Dezember 2005 – bzw. bis zum 31. Dezember 2003, wenn sich das Angebot speziell an behinderte Menschen richtet – müssen alle Angebote nach der Verordnung barrierefrei umgesetzt sein⁴.

Gewerbliche Anbieter sollen per Zielvereinbarungen zum Umgestalten ihrer Internetangebote bewegt werden. Durch das Inkrafttreten des Allgemeinen Gleichbehandlungsgesetzes⁵ am 18.08.2006 sollte der geforderten Umsetzung auch in der Wirtschaft zusätzlich Nachdruck verliehen werden.

Viele Bundesländer haben mittlerweile gültige, auf das [BGG](#) verweisende oder ähnlich lautende Gesetze verabschiedet, die mehr oder minder umfassend eine Gleichstellung fordern⁶. Dies betrifft aber größtenteils nicht die kommunalen Internetangebote.

Auch die Europäische Union hat eine Vereinbarung darüber getroffen, dass die Mittelvergabe künftig von der Barrierefreiheit abhängig ist⁷.

1 Gesetz zur Gleichstellung behinderter Menschen

2 vgl. BITV - Anlage [des Innern \[2002\]](#) und M. Müller auf der Eafra09, 5:18 f. Müller [\[2009\]](#)

3 vgl. K. Warnke - Entstehung der BITV [Warnke \[2006\]](#)

4 vgl. BITV - § 4 Umsetzungsfristen für die Standards [des Innern \[2002\]](#)

5 in früheren Entwürfen auch Antidiskriminierungsgesetz (ADG) oder Zivilrechtliches Antidiskriminierungsgesetz (ZAG) genannt.

6 vgl. Einfach-für-Alle Blog - Gleichstellungsgesetze im Bund und in den Ländern, folgende [Caspers \[2009a\]](#)

7 vgl. Kabinet Nachrichten Mai 06 - [kabinet nachrichten \[2006a\]](#)

4.2.1 Aufbau und Relevanz

Da die [BITV](#) in ihrer ersten Version auf der mittlerweile veralteten [WCAG](#) 1.0 beruht und aufgrund der Tatsache, dass die zweite Version der [BITV](#) auf der bereits erschienen [WCAG](#) 2.0 beruhen wird, kann an dieser Stelle von einer ausgiebigen Analyse der [BITV](#) 1 zugunsten einer detaillierten Ausarbeitung der [WCAG](#) 2.0 abgesehen werden.

Die Verordnung besteht aus 14 tabellarisch aufgeführten Anforderungen, die im Prinzip die verkürzte, übersetzte und in der Nummerierung geänderte [WCAG](#) 1.0 darstellen. Diese Anforderungen enthalten wiederum eine oder mehrere Bedingungen.

An Stelle der 3 Prioritätsstufen der [WCAG](#) kennt die [BITV](#) nur 2 Stufen: Priorität I und Priorität II.

Im folgenden werden die 14 Anforderungen kurz skizziert:

1. Audio- oder visuelle Inhalte müssen mit gleichwertigen Alternativen ausgestattet werden.
2. Informationen müssen auch ohne Farben erfassbar sein.
3. Quellcode und Stylesheets müssen gemäß der entsprechenden Spezifikation benutzt werden.
4. Sprachliche Besonderheiten müssen erkennbar gemacht werden.
5. Tabellen sind nur zur Darstellung tabellarischer Daten zu nutzen.
6. Internetangebote müssen rückwärtskompatibel sein.
7. Zeitgesteuerte Änderungen des Inhalts müssen kontrollierbar sein.
8. Eingegebettete Inhalte müssen ebenfalls zugänglich sein.
9. Internetangebote müssen unabhängig vom Ein- oder Ausgabegerät nutzbar sein.
10. Ältere [UA](#)¹ und [AT](#) müssen unterstützt werden, wenn der Aufwand vertretbar ist.
11. Verwendete Technologien sollten öffentlich zugänglich und vollständig dokumentiert sein.
12. Es müssen Informationen zum Kontext und zur Orientierung bereitgestellt werden.
13. Navigationsmechanismen sollten übersichtlich und schlüssig gestaltet werden.

¹ User Agent; Client-Programm, mit dem ein Netzwerkdienst genutzt werden kann. Es ist die Schnittstelle zum Benutzer, die die Inhalte darstellt und Befehle entgegennimmt.

14. Das allgemeine Verständnis der angebotenen Inhalte ist zu fördern.

Diese Anforderungen enthalten jeweils eine oder mehrere Bedingungen, die je nach Prioritätsstufe zu erfüllen sind.

Eine ausgezeichnete Erklärung mit Beispielen und ein Abgleich mit einem aktuellen technischen Stand findet man auf der Einfach-für-Alle Internetseite der Aktion Mensch¹.

4.2.2 Kritik an der BITV

Die Tragweite der erlassenen Gesetze darf angezweifelt werden. Nach verschiedenen Gesprächen mit Betroffenen und eigener Recherche stellt sich die Umsetzung als unzureichend dar. Die Verordnung sind zwar durch die Landesgesetze nicht nur für Angebote des Bundes verpflichtend – sie ziehen aber keine Konsequenzen bei Nichterfüllung nach sich. Auch auf europäischer Ebene scheint noch einiger Nachholbedarf zu bestehen².

Auch bei der [BITV](#) fehlen, wie bei der [ISO](#)-Richtlinie, Technologie-Empfehlungen, Best-Practices und konkrete Beispiele. Die [BITV](#) verweist hier aber auf die Dokumente der [WAI](#) und damit auf die begleitenden Materialien der [WCAG](#).

Von verschiedenen Seiten wurde den Richtlinien vorgeworfen, technisch veraltet und etwas knapp gehalten zu sein sowie nicht genug auf verschiedene Arten von Behinderung einzugehen³. All diese Probleme waren aber schon an der zugrunde liegenden [WCAG 1.0](#) bemängelt worden. Auch sollte nicht vergessen werden, dass die [BITV](#) eine Verordnung für Behörden darstellt.

„Bei aller Kritik an einzelnen Details der BITV – sie ist als politische Richtlinie unentbehrlich.“⁴

Denn trotz allem hat die [BITV](#) maßgeblich zur Verbesserung der Situation beigetragen: Behörden müssen sich früher oder später mit dem Thema auseinander setzen und auch eine gewisse Öffentlichkeit konnte so für die Thematik gewonnen werden.

4.2.3 BITV 2

Nach 3 Jahren wurde die [BITV](#) wie geplant in Zusammenarbeit mit Behindertenverbänden überprüft. Ergebnis war vor allem die Forderung nach einer Anpassung an den

1 <http://www.einfach-fuer-alle.de/artikel/bitm-reloaded/>

2 vgl. Diskussion zur Harmonisierung von europäischen Barrierefreiheitsrichtlinien auf der Eafra09 - bis 5:15 [eaf \[2009\]](#)

3 vgl. BIK Blog, f. [Zapp \[2005\]](#)

4 Zitat: Tomas Caspers von Einfach-für-alle [Caspers \[2007\]](#)

Stand der Technik und die Aufnahme von Regelungen für gehörlose sowie kognitiv beeinträchtigte Menschen. Des weiteren sollte die Aufnahme von leichter Sprache und Gebärdensprache als Bedingung überprüft werden. Seit Mai 2007 beschäftigt sich die Arbeitsgruppe mit der Aktualisierung¹ der [BITV](#).

Für die [BITV](#) 2.0 sollen die Prinzipien, Richtlinien und Erfolgskriterien aus der [WCAG](#) 2.0 übernommen und es soll auf ihre Techniken verwiesen werden². Die Kriterien sollen allerdings durch nationale Besonderheiten wie abweichende Schulformen (relevant für die Sprachvorgaben) sowie grundsätzlichere Regelungen in Bezug auf Gebärdensprache ergänzt werden³. Es soll durchgesetzt werden, dass grundlegende Informationen eines Angebotes in Gebärdensprache verfügbar sind und diese gewissen Standards entsprechen. Das nationale Gegebenheiten Anpassungen nötig machen, wird von der [WAI](#) in der [WCAG](#) auch angemerkt und erbeten. Die [W3C](#) wünscht sich allerdings, dass lokale Anpassungen auch in die [WCAG](#) selbst einfließen⁴, um sie einem möglichst breitem Publikum anzubieten, zum Beispiel ausländischen Unternehmen mit deutschem Internetangebot.

Auf der [EAFRA](#)⁵ wurde Kritik an der [BITV](#)-Verordnung generell und vor allem an deren Erarbeitungsweise geäußert⁶. Dass die Erarbeitung wieder nicht öffentlich ist, wurde schon mehrfach kritisiert, zum Beispiel von Martin Kliehm, der zugleich auch auf die Missstände bei der Umsetzung in der Wirtschaft und beim Bund aufmerksam macht⁷.

4.3 WCAG 2.0

Die erste Fassung ([WCAG](#) 1.0) dieser für den internationalen Einsatz konzipierten Sammlung von Richtlinien wurde im Mai 1999 von der Arbeitsgruppe [WAI](#) veröffentlicht. Dieser Arbeit wird die am 11. Dezember 2008 veröffentlichte, aktualisierte Version 2.0 der [WCAG](#) zu Grunde gelegt. Die [WCAG](#) 2.0 ist eine verbesserte, abwärtskompatible Version der [WCAG](#) 1.0 und enthält einige elementare Änderungen sowie Ergänzungen. Die Entwicklung der [WCAG](#) wird stetig fortgeführt, um neue Technologien aufzunehmen und Verbesserungen vorzunehmen.

Entwickelt wurden die Richtlinien in Zusammenarbeit mit verschiedenen Personen und Organisationen auf der ganzen Welt. Es ist prinzipiell jedem möglich, an der Gestal-

1 Drucksache 16/9283 des dt. Bundestag - S.33 [Bundesregierung \[2008\]](#)

2 vgl. M. Müller auf der Eafra09, 6:20 f. [Müller \[2009\]](#)

3 vgl. M. Müller auf der Eafra09, 7:10 f. [Müller \[2009\]](#)

4 vgl. Shadi Abou-Zahra auf der Eafra09 - ab 10:43 [eaf \[2009\]](#)

5 Europäisches Accessibility Forum Frankfurt am Main

6 vgl. Diskussion zur Harmonisierung von europäischen Barrierefreiheitsrichtlinien auf der Eafra09 - ab 5:28 [eaf \[2009\]](#)

7 vgl. Namics Weblog [Kliehm \[2009\]](#)

tung und Ausarbeitung der Richtlinien mitzuwirken¹ oder die bestehenden Richtlinien zu kommentieren² bzw. Techniken vorzuschlagen³.

Die [WCAG](#) richtet sich an Webentwickler, die Internetseiten erstellen, Entwickler von Software zur Erstellung von Internetseiten, Entwickler von sogenannter Hilfstechnologie⁴ und Andere, die einen technischen Standard für barrierefreies Internet benötigen. Sie hat das Ziel, Entwicklern von aktuellen und zukünftigen Internetinhalten Richtlinien für Barrierefreiheit aufzuzeigen. Internetinhalte können hier alle Arten von im Internet angebotenen Inhalten und Applikationen wie Text, Bilder, Formulare und multimediale Inhalte sein.

Die Richtlinien sollen die Verwendbarkeit von Internetinhalten für Menschen mit Behinderungen verbessern, haben aber nicht den Anspruch jede Art und jede mögliche Kombination von Behinderung abzudecken⁵.

4.3.1 Aufbau und Relevanz

Die [WCAG](#) 2.0 gliedert sich in eine Sammlung von englischsprachigen Dokumenten⁶ die sich untereinander ergänzen und verlinken. Die Wichtigsten sind die eigentliche Richtlinie (auf die im folgenden näher eingegangen wird) ⁷, eine detaillierten Referenz⁸ mit Erläuterungen und Beispielen, eine anpassbare Schnellreferenz⁹ und ein Dokument mit technischen Erläuterungen, Code-Beispielen und typischen Fehlern¹⁰. Meist enthalten die Dokumente Verweise zu Ressourcen oder Dokumenten mit weiterführenden Informationen zu den verschiedenen Spezialthemen.

Dazu kommen noch einige andere Dokumente, die zum Beispiel einen Überblick über die Zusammenhänge der einzelnen Dokumente geben oder typische Fragen zur [WCAG](#) beantworten.

Die eigentliche Richtlinie [WCAG](#) 2.0 gliedert sich in 4 Prinzipien mit verschiedenen Richtlinien, in denen dann wiederum die eigentlichen Kriterien enthalten sind. All diese

1 Participating in WAI, <http://www.w3.org/WAI/participation.html>

2 Instructions for Commenting on WCAG 2.0 Documents, <http://www.w3.org/WAI/WCAG20/comments/Overview.html>

3 Techniques for WCAG 2.0 submission form, <http://www.w3.org/WAI/GL/WCAG20/TECHS-SUBMIT/>
4 wie z. B. Braillezeile, Screenreader oder Bildschirmluppen

5 vgl: Introduction WCAG 2.0 [WAI \[2009i\]](#)

6 Verschiedene Überstzungen sind bereits in Vorbereitung; siehe [WAI \[2009h\]](#); die deutsche wird von der Aktion-Mensch erarbeitet, vgl. [Mailingliste des W3C](#)

7 Web Content Accessibility Guidelines (WCAG) 2.0, <http://www.w3.org/TR/WCAG20/>

8 <http://www.w3.org/TR/UNDERSTANDING-WCAG20/>

9 How to Meet WCAG 2.0, <http://www.w3.org/WAI/WCAG20/quickref/>

10 Techniques for WCAG 2.0, <http://www.w3.org/TR/WCAG20-TECHS/>

Richtlinien und einige Kriterien werden zunächst vorgestellt und, sofern sie für die spätere technische Umsetzung relevant sind, auch näher erläutert.

Internetseiten können verschiedenen Grade der Konformität erreichen, je nach erfüllten Kriterien:

Level A stellt den kleinsten Grad an Konformität dar und kann auch durch Anbieten einer mit gleichem Inhalt¹ versehenen alternativen Seite erreicht werden.

Level AA konforme Internetseiten sollen alle Kriterien des Level A und AA erfüllen. Dieser Grad kann ebenfalls durch eine alternative Seite erreicht werden.

Level AAA bezeichnet den höchsten möglichen Grad an Barrierefreiheit. Webseiten sind Level AAA konform wenn sie Grad A, AA und AAA erreichen. Dieser Grad kann ebenfalls durch eine alternative Seite erreicht werden.

Die Kriterien gelten immer für die ganze Internetseite² und komplette Prozesse.

Principle 1: Perceivable

Information and user interface components must be presentable to users in ways they can perceive.⁵

Inhalte müssen für den Nutzer in der für ihn wahrnehmbaren Form erfassbar sein. Das umfasst folgende Richtlinien und Kriterien:

Guideline 1.1: Text Alternatives

Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.⁶

Alle nicht in Textform vorliegenden Inhalte (wie Grafiken, Multimedia und zum Beispiel Formulare oder Bedienfelder) sollen mit Alternativen in Textform versehen werden, die gegebenenfalls in Form von großen Ausdrucken, Blindenschrift, Sprachausgabe, Symbolen oder vereinfachter Sprache darstellbar sind.

Relevant ist hier Kriterium *1.1.1 Non-text Content* (Level A), das besagt, dass alle nicht in Textform vorliegenden Elemente eine entsprechende Alternative in Textform haben sollen, die den gleichen Zweck erfüllt. Diese kann, bei vorhandener Spezifikation,

1 d.h. gleiche Sprache und Funktionalität sowie Aktualität; vgl. WCAG 2.0 - Conformance [WAI \[2009\]](#)

2 Von der [WAI](#) definiert als die nicht eingebettete Ressource, die unter einer einzelnen [URI](#)³ (per [HTTP](#)) erreichbar ist.

5 Zitat: WCAG 2.0 - Principle 1: Perceivable [WAI \[2009\]](#)

6 Zitat: WCAG 2.0 - Principle 1: Perceivable [WAI \[2009\]](#)

direkt innerhalb des Elements definiert werden. In bestimmten, sinnvollen Sonderfällen kann auf die Alternative verzichtet werden.

Guideline 1.2: Time-based Media

Provide alternatives for time-based media.¹

Diese Richtlinie beschäftigt sich mit Zeit-basierten Medien wie Video und Audio und liegt damit außerhalb der Zielsetzung dieser Arbeit. Sie besagt das für Zeit basierte Medien eine Alternative in Form von Audio, Video (zum Beispiel Gebärdensprache) oder Text (zum Beispiel Untertitel oder Erklärungen) vorhanden sein sollen.

Guideline 1.3: Adaptable

Create content that can be presented in different ways (for example simpler layout) without losing information or structure.²

Inhalte sollen in verschiedenen Formen dargestellt werden können, ohne Information oder Struktur zu verlieren. Dies bezieht sich sowohl auf die Art des Inhalts als auch auf seine visuelle Präsentation.

Kriterium 1.3.1 *Info and Relationship* (Level A) verlangt, dass Informationen, Strukturen und Beziehungen, die durch eine bestimmte Präsentation dargestellt werden, programmatisch (das heißt durch technische Maßnahmen) bestimmbar sein sollen oder als Text verfügbar sein sollen. Dies wird durch semantisch korrekte Auszeichnung erreicht.

Kriterium 1.3.2 *Meaningful Sequence* (Level A) bedeutet, dass, wenn die Struktur eines Inhalts von seiner Reihenfolge abhängt, diese programmatisch bestimbar sein soll. Dies wird durch eine sinnvolle programmatische Reihenfolge der Elemente erreicht.

Kriterium 1.3.3 *Sensory Characteristics* (Level A) soll vermeiden, dass die Bedeutung einer Information von sensorischen Faktoren wie seiner Form, Größe, visuellen Anordnung, Orientierung oder Geräusch abhängt. Farben werden gesondert in Kriterium 1.4.1 behandelt.

1 Zitat: WCAG 2.0 - Principle 1: Perceivable [WAI \[2009\]](#)

2 Zitat: WCAG 2.0 - Principle 1: Perceivable [WAI \[2009\]](#)

Guideline 1.4: Distinguishable

Make it easier for users to see and hear content including separating foreground from background.¹

Diese Richtlinie soll dafür sorgen, dass der Nutzer in der Lage ist, die Information gegenüber dem visuellen und akustischen Hintergrund wahrzunehmen.

Kriterium 1.4.1 *Use of Color* (Level A) besagt, dass Farbe nicht als einziges Mittel verwendet werden soll, um eine Information kenntlich zu machen. Relevant ist dieses Kriterium insbesondere, da oft Farbe zum Markieren von Fehlern genutzt wird.

Kriterium 1.4.4 *Resize Text* (Level AA) zeigt auf, dass alle Texte – mit Ausnahme von Untertiteln für Audio-Inhalte – um 200 Prozent zu vergrößern sein sollen, ohne Inhalt oder Funktionalität zu verlieren. Dies soll ohne [AT](#) möglich sein.

Kriterium 1.4.5 *Images of Text* (Level AA) verfügt, dass Informationen, wenn der [UA](#) die gewünschte Darstellung visualisieren kann, im Normalfall immer als Text anstelle von Grafiken dargestellt werden sollen. Ausnahmen in Spezialfällen, wenn zum Beispiel die Grafik vom Nutzer angepasst werden kann oder wenn die Darstellungsform elementar ist², sind akzeptabel.

Kriterium 1.4.5 *Images of Text (No Exception)* (Level AAA) schließt die Benutzung von Grafiken außer zu reinen Darstellung aus, es sei denn, die Darstellungsform ist elementar.

Principle 2: Operable

User interface components and navigation must be operable.³

Die Elemente der Bedienoberfläche und die Navigation müssen bedienbar sein. Das beinhaltet Tastaturbedienbarkeit, zeitliche Abfolgen, die Funktion oder Inhalt betreffen, blinkende Elemente, scrollende Elemente und automatische Aktualisierungen.

Guideline 2.1: Keyboard Accessible

Make all functionality available from a keyboard.⁴

1 Zitat: WCAG 2.0 - Principle 1: Perceivable [WAI \[2009\]](#)

2 Dies kann zum Beispiel bei einem Logo der Fall sein.

3 Zitat: WCAG 2.0 - Principle 2: Operable [WAI \[2009\]](#)

4 Zitat: WCAG 2.0 - Principle 2: Operable [WAI \[2009\]](#)

Diese Richtlinie legt fest, dass alle Funktionen mit der Tastatur nutzbar sein sollen und wie diese Vorgabe zu erfüllen ist.

Kriterium 2.1.1 *Keyboard* (Level A) verlangt, dass alle Funktionen durch eine Tastatur nutzbar sein sollen. Dies soll ohne zeitliche Beschränkungen möglich sein. Beim Erfassen von Bewegungsmustern des Nutzers ist eine Ausnahme zulässig, wenn nicht nur Start- und Endpunkt der Nutzereingabe, sondern der Weg benötigt wird.

Kriterium 2.1.2 *No Keyboard Trap* (Level A) soll dafür sorgen, dass es jederzeit möglich sein soll, mit der Tastatur den Fokus auf ein anderes Element zu legen. Insbesondere muss dies auch möglich sein, wenn sich der Fokus innerhalb eines sogenannten Applets oder Widgets befindet.

Kriterium 2.1.3 *Keyboard (No Exception)* (Level AAA) bestimmt, dass alle Funktionen durch eine Tastatur nutzbar sein sollen. Dies soll ohne zeitliche Beschränkungen möglich sein.

Guideline 2.2: Enough Time

Provide users enough time to read and use content.¹

Diese Richtlinie legt Kriterien für die Entwicklung von Zeit basierten, blinkenden, scrollenden und automatisch aktualisierten Inhalten fest. Diese sind für die vorliegende Arbeit kaum relevant, da keine auf Zeit basierenden Applikationen behandelt werden.

Kriterium 2.2.4 *Interruptions* (Level AAA) verlangt, dass alle Unterbrechungen vom Nutzer hinausgezögert oder unterdrückt werden können. Ausnahme sind Notfälle².

Guideline 2.3: Seizures

Do not design content in a way that is known to cause seizures.³

Diese Richtlinie legt Vorgaben für die Nutzung von Inhalten fest, die dafür bekannt sind, Krämpfe und epileptische Anfälle auszulösen. Diese Richtlinie ist für diese Arbeit

1 Zitat: WCAG 2.0 - Principle 2: Operable [WAI \[2009i\]](#)

2 Als Ausnahmen bezeichnet die [WAI](#) Notfälle die Gesundheit, Sicherheit oder Besitz betreffen.

3 Zitat: WCAG 2.0 - Principle 2: Operable [WAI \[2009i\]](#)

nicht relevant, da keine der betroffenen Inhalte verwendet werden.

Guideline 2.4: Navigable

Provide ways to help users navigate, find content, and determine where they are.¹

Diese Richtlinie verpflichtet den Programmierer, den Nutzer bei der Navigation und Orientierung zu unterstützen. Sie bestimmt Vorgaben zum Überspringen und Fokussieren von Seitenelementen, für den Titel der Internetseite, für Link-Beschriftungen und Überschriften.

Kriterium 2.4.3 *Focus Order* (Level A) besagt, dass die Elemente einer Internetseite der Reihenfolge nach den Fokus erhalten. Diese Reihenfolge soll semantisch korrekt sein und die Bedienbarkeit unterstützen.

Kriterium 2.4.7 *Focus Visible* (Level AA) legt darauf Wert, dass die fokussierten Bedienelemente einer Internetseite visuell kenntlich gemacht sind.

Principle 3: Understandable

Information and the operation of user interface must be understandable.²

Hier werden Vorgaben gemacht, die die verwendete Sprache sowie deren Komplexität und Eigenheit, Aktualisierungen durch Veränderungen des Status von Bedienoberflächen, Fehlermeldung und Fehlerverhütung betreffen.

Guideline 3.1: Readable

Make text content readable and understandable.³

Diese Richtlinie beschreibt Kriterien, die der Verständlichkeit und der Lesbarkeit von Texten dienen. Sie reglementiert die Komplexität der Sprache und erläutert die notwendige Kennzeichnung ungewöhnlicher Worte, Abkürzungen und der verwendeten Sprache der Internetseite oder Teilen davon. Sprachbezogene Richtlinien sind für diese technisch orientierte Arbeit nicht von Belang.

1 Zitat: WCAG 2.0 - Principle 2: Operable [WAI \[2009\]](#)

2 Zitat: WCAG 2.0 - Principle 3: Operable [WAI \[2009\]](#)

3 Zitat: WCAG 2.0 - Principle 3: Understandable [WAI \[2009\]](#)

Guideline 3.2: Predictable

Make Web pages appear and operate in predictable ways.¹

Diese Richtlinie soll dem Nutzer – unter anderem durch Kriterien für Status-Änderungen und Navigation – helfen, sich auf Internetseiten zurecht zu finden.

Kriterium 3.2.1 *On Focus* (Level A) bedeutet, dass ein Element keine Kontext Änderung herbeiführen soll, wenn fokussiert wird.

Kriterium 3.2.2 *On Input* (Level A) soll vermeide, dass ein Element eine Kontextänderung herbeiführt, wenn sein Status² verändert wird. Hier ist zu differenzieren, dass ein Verändern des Inhaltes nicht zwangsläufig ein Verändern des Kontextes bedeuten muss.³

Kriterium 3.2.5 *Change on Request* (Level AAA) soll dafür sorgen, dass eine Kontextänderung nur durch den Nutzer initiiert wird oder ein Mechanismus zur Verfügung steht, um gegenteilige Funktionen abzuschalten.

Guideline 3.3: Input Assistance

Help users avoid and correct mistakes.⁴

In dieser Richtlinie werden Kriterien zu Fehlermeldung und Fehlervermeidung sowie zur Kennzeichnung und Erklärung von Inhalten, die Nutzereingaben erfordern, beschrieben.

Kriterium 3.3.1 *Error Identification* (Level A) legt fest, dass ein automatisch erkannter Fehler durch das Kenntlichmachen des fehlerhaften Elements und durch eine Erklärung in Textform angezeigt werden soll.

Kriterium 3.3.2 *Labels or Instructions* (Level A) besagt, dass Auszeichnungen oder Erklärungen angegeben werden sollen, wenn Inhalte Nutzereingaben erfordern.

1 Zitat: WCAG 2.0 - Principle 3: Understandable [WAI \[2009\]](#)

2 Status kann hier zum Beispiel das aktivieren einer Checkbox sein oder das eingeben von Text in ein Formularfeld

3 vgl. Understanding WCAG 2.0 - Understanding Success Criterion 3.2.1, Techniques and Failures, Sufficient Techniques, Note 1 [WAI \[2009d\]](#)

4 Zitat: WCAG 2.0 - Principle 3: Understandable [WAI \[2009\]](#)

Kriterium 3.3.3 *Error Suggestion* (Level AA) zeigt auf, dass, wenn ein Fehler erkannt wurde und mögliche Lösungen bekannt sind, diese dem Nutzer angeboten werden sollen. Eine Gefährdung der Sicherheit oder des Sinns stellen eine Ausnahme dar.

Kriterium 3.3.4 *Error Prevention (Legal, Financial, Data)* (Level AA) bestimmt, dass Internetseiten die Nutzereingaben erfordern, die finanzielle Transaktionen, rechtskräftige Verpflichtungen oder Datensatz verändernde Angaben betreffen, mindestens eine der folgenden Bedingungen erfüllen sollen: Absenden kann zurückgenommen werden, eingegebene Daten werden auf Fehler überprüft und können vom Nutzer korrigiert werden, ein Mechanismus gestattet das Überprüfen und gegebenenfalls das Korrigieren der Daten vor dem Versand.

Kriterium 3.3.6 *Error Prevention (All)* (Level AAA) verlangt, dass alle Internetseiten, die Nutzereingaben erfordern, mindestens eine der folgenden Bedingungen erfüllen sollen: Absenden kann zurückgenommen werden, eingegebene Daten werden auf Fehler überprüft und können vom Nutzer korrigiert werden, ein Mechanismus gestattet das Überprüfen und gegebenenfalls Korrigieren der Daten vor dem Senden.

Principle 4: Robust

Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.¹

Das Prinzip besagt, dass Inhalte robust genug sein sollen, um auf einer breiten Palette von [UA](#) und [AT](#) nutzbar zu sein.

Guideline 4.1: Compatible

Maximize compatibility with current and future user agents, including assistive technologies.²

Diese Richtlinie soll dafür sorgen, dass die Kompatibilität zu aktuellen und zukünftigen [UA](#) sichergestellt ist. Dies betrifft sowohl die Qualität des Quelltextes als auch den Zugriff auf dessen Elemente.

Kriterium 4.1.1 *Parsing* (Level A) verfügt, dass valide Auszeichnungssprache gemäß Spezifikation verwendet werden soll.

1 Zitat: WCAG 2.0 - Principle 4: Robust [WAI \[2009\]](#)

2 Zitat: WCAG 2.0 - Principle 4: Robust [WAI \[2009\]](#)

Kriterium 4.1.2 *Name, Role, Value* (Level A) legt fest, dass der Name und die Rolle (Funktion) aller Bedienoberflächen programmatisch bestimmbar sein soll. Status, Eigenschaften und Werte sollen ebenfalls programmatisch erfassbar sein und Änderungen daran sollen durch den UA inklusive AT erfassbar sein. Standard HTML-Elemente erfüllen dieses Kriterium bereits.

4.3.2 Kritik an der WCAG 2.0

An der zweiten Version der Richtlinie, genauer an deren erstem Entwurf Anfang 2006, wurde massiv Kritik geäußert. Nachdem die W3C zu einer letzten Kontrolle vor der *Candidate Recommendation*¹ aufrief, musste die recht knapp angelegte Frist verlängert werden, um der Kritik mehr Raum für Diskussionen zu lassen². Es wurde mehrfach nachgebessert und wieder Kritik geäußert³. Bis Anfang 2008 durften dann Kommentare zu einem zweiten *Last Call Working Draft* gemacht werden. Nachdem diese letzten Änderungswünsche diskutiert und ggf. eingebracht wurden, hat man die Richtlinien im Dezember 2008 zur Empfehlung erklärt.

Im nachfolgenden sollen einige der wichtigsten Kritikpunkte aufgenommen und kurz besprochen werden.

Kritisiert wurde die Menge der zu bewältigenden Dokumente und die zu allgemeine Formulierung. Beide Punkte sind nicht von der Hand zu weisen. Die Richtlinien wurden zu Gunsten von Zukunftsfähigkeit und Anwendbarkeit sehr unspezifisch verfasst. Erst zusammen mit den zugehörigen *Understanding* und *Techniques* Dokumenten ergeben Sie anwendbare Richtlinien. Dies – zusammen mit den stellenweise langen, komplizierten und fragmentierten Aussagen – macht die Richtlinie und ihre Dokumente unübersichtlich und es stellenweise schwierig mit ihnen zu arbeiten.

Dass Tabellen, die der Darstellung dienen, nicht ausdrücklich verboten sind, hat mehrfach für Irritation gesorgt. Es untergräbt – zusammen mit dem Eindruck, dass kein valider Quellcode gefordert wird – die Bemühungen, Internetentwickler an Standards heranzuführen. Valider Quellcode wird aber in der aktuellen Version gefordert und Tabellenlayout wird ausdrücklich nicht empfohlen.

Immer noch kritisiert wird die Erlaubnis, zugängliche und unzugängliche Seiten voneinander zu trennen und Seiten auszuklammern.

1 Die dritte Stufe des für W3C-Empfehlungen festgelegten fünfstufigen Prozesses.

2 vgl. W3C Mailinglist <http://lists.w3.org/Archives/Public/w3c-wai-ig/2006AprJun/0083.html>

3 vgl. BIK Webseite - Teil 1: Vorstellung der WCAG 2.0 BIK [2008]

Unverständlich auch für den Autor ist, dass die Funktionsfähigkeit einer Applikation ohne Java-Script nicht in die [WCAG](#) 2.0 übernommen wurde. In der ersten Version war diese sinnvolle Anforderung noch enthalten¹.

Ein weiterer Kritikpunkt aus Sicht des Autors ist die unzureichende Aufnahme der [WAI ARIA](#) Spezifikation in die Richtlinie. Bisher wird zwar das Ergebnis von korrekter Implementation von [ARIA](#) gefordert, jedoch nicht der Standard selbst. Dies erklärt sich vermutlich aus dem frühen Entwurfsstadium der Spezifikation während der abschließenden Arbeiten an der [WCAG](#). Es ist anzunehmen, dass dies in naher Zukunft geändert wird: Eine Version 2.1 ist bereits in Arbeit².

Trotz einiger Kritik ist die [WCAG](#) die verlässlichste und am weitesten angenommene Richtlinie zur Barrierefreiheit im Netz. Sie bietet umfangreiches Wissen und konkrete Umsetzungsempfehlungen für Webentwickler. Wichtig ist in diesem Zusammenhang die stetige Weiterentwicklung und der relativ transparente Entwicklungsprozess, bei dem eine Einflussnahme bewiesenermaßen gewährleistet ist. Abschließend kann man sagen, dass die [WCAG](#) auf absehbare Zeit der Standard für Barrierefreiheit im Internet ist und bleiben wird, an dem sich auch weiterhin die meisten nationalen Regelungen orientieren werden.

4.4 WAI ARIA

Die [WAI ARIA](#) ist eine Spezifikation für Internetinhalte, speziell Applikationen, um diese barrierefrei zu erstellen. Sie soll beim Programmieren von komplexen Benutzeroberflächen mit dynamischen Inhalten auf Basis von HTML, CSS, Java-Script (insbesondere [AJAX](#)) und ähnlich gelagerten Technologien helfen.

Obwohl die Spezifikation noch nicht in einer finalen Version vorliegt ist seine Implementation in [AT](#) bereits fortgeschritten und hat einen nachweisbaren Nutzen für Barrierefreiheit bei Internetanwendungen³. Auch die [WCAG](#) 2.0 empfiehlt ihre Verwendung⁴ und sie gilt gemeinhin als anerkannter Standard. Aus diesen Gründen soll sie bei der praktischen Umsetzung Verwendung finden.

Viele der mittlerweile oft genutzten Benutzeroberflächen – wie Schieberegler, komplexe Dialoge, während der Laufzeit ausklappbare Listen (Baumansicht) und sortierbare Tabellen – sind mit den Standard-Elementen nicht abzubilden und werden daher aus den verfügbaren [HTML](#)-Elementen nachgebaut. Diese Konstrukte sind meist für die [AT](#) kaum verständlich und somit von Menschen mit Behinderung nur partiell oder gar nicht

1 vgl. WCAG 1.0 - Checkpoint 6.3 [WAI](#) [1999]

2 vgl. Shadi Abou-Zahra auf der Eafra09 - ab 16:16 [eaf](#) [2009]

3 vgl. mit Punkt 3.1: Unterstützungstechnologie

4 vgl. Techniques for WCAG 2.0 - ARIA1, ARIA2, ARIA3, ARIA4 [WAI](#) [2009c]

nutzbar.

WAI ARIA soll hier Abhilfe schaffen, indem es Semantik in die verwendeten HTML-Elemente bringt. Mithilfe von verschiedenen zusätzlichen Eigenschaften, Attributen und Zuständen sollen die Elemente einer Webseite genauer als bisher beschrieben werden können. Der UA bzw. die AT kann diese Informationen dann zur besseren Nutzbarkeit verarbeiten und darauf reagieren sowie die Informationen an den Nutzer weitergeben.

Standard-HTML-Elemente wie Listen, Formularfelder oder Überschriften reichen oft für moderne Applikationen und deren Benutzeroberflächen nicht mehr aus. Abhilfe kommt vielleicht mit der neuen HTML 5 Spezifikation, in der neue Elemente eingeführt werden sollen¹. Bis diese aber eingeführt werden und vor allem von den weit verbreiteten UA unterstützt werden, kann noch einige Zeit vergehen. Auch gibt es Überschneidungen, die bereits diskutiert werden², aber nicht Thema dieser Arbeit sind. Die zusätzlichen Eigenschaften, Attribute und deren Zustände gehören zur Zeit nicht zum XHTML-Standard, es sind aber bereits verschiedene DTD³ geplant, um Quellcodes mit WAI ARIA-Unterstützung validierbar zu machen⁴. Die hinzugefügten Attribute und Eigenschaften werden von alten Browsern ignoriert⁵, so dass keine negativen Seiteneffekte auftreten.

Neben der eigentlichen Richtlinie gibt es noch einige weitere Dokumente wie die Grundlagensammlung⁶, die Best-Practice-Dokumente⁷ und einige weitere den Aufbau und die Struktur erläuternde Dokumente.

Auf die wichtigsten Techniken der zur Zeit noch als Arbeitsversion vom 24. Februar 2009⁸ vorliegenden Spezifikation wird im folgenden eingegangen. Dabei orientiert sich die Reihenfolge der Punkte an der oben genannten Best-Practice und geht auf die relevanten Zustände und Eigenschaften der Spezifikation ein.

4.4.1 Tastatur-Navigation und Fokus

In HTML 4.01⁹ wurde festgelegt, dass Formularelemente und Links vom Nutzer mit der Tabulator Taste in ihrer programmatischen Reihenfolge durchgegangen werden können.

1 vgl. HTML 5 W3C [2009]

2 vgl. W3C Mailing List <http://lists.w3.org/Archives/Public/public-html/2007Jul/0903.html>

3 Dokumenttypdefinition; ein Satz an Regeln, der benutzt wird, um Dokumente eines bestimmten Typs zu deklarieren.

4 vgl. WAI-ARIA Best Practices - Introduction WAI [2009f]; vgl. States and Adaptable Properties Module WAI [2009b]

5 vgl. WAI-ARIA FAQ - What happens in older browsers when WAI-ARIA is implemented? WAI [2009g]

6 WAI-ARIA Primer <http://www.w3.org/TR/wai-aria-primer/>

7 WAI-ARIA Best Practices <http://www.w3.org/TR/wai-aria-practices/>

8 vgl. Accessible Rich Internet Applications (WAI-ARIA) 1.0 WAI [2009a]

9 HTML 4.01 Specification <http://www.w3.org/TR/html401/>

Bei vielen Elementen, die zur Entwicklung eigener Applikationen (wie bei Listen, Bildern oder den obligatorischen divs) benötigt werden, ist dies nicht möglich, kann aber mit dem Attribut `tabindex` manuell festgelegt werden.

Mit den [ARIA](#) Vorgaben wird das Attribut `tabindex` für alle [HTML](#) Elemente mit folgenden Auswirkungen möglich¹:

- `tabindex = 0` wird zur Tab-Reihenfolge des Dokuments hinzugefügt
- `tabindex > 0` wird je nach Wert zur Tab-Reihenfolge hinzugefügt
- `tabindex < 0` wird nicht zur Tab-Reihenfolge hinzugefügt, kann aber per Skript fokussiert werden

Ein Hinzufügen zum Tabindex² ist ausreichend, wenn der Quelltext eine logische Struktur hat.

Innerhalb von Applikationen soll mit Pfeiltasten, Enter, Leertaste und Escape gesteuert werden können. Das bedeutet, dass das Elternelement ein `tabindex = 0` sein muss, damit es innerhalb der Dokumentstruktur anwählbar ist; die Kindelemente einen `tabindex = -1`, damit sie nicht aufgenommen werden, aber fokusierbar sind.

Um den Status (das letzte angewählte Element) zu speichern, kann nun entweder der Tabindex der Kinder geändert werden (alle erhalten `tabindex = -1`, außer dem aktiven, der erhält `tabindex = 0`) oder es kann das [ARIA](#) Attribut `aria-activedescendant` verwendet werden. In diesem dem Elternelement hinzugefügten Attribut wird bei jeder Fokusänderung die [ID](#)³ des aktiven Elements gespeichert. Der [UA](#) sollte sich dann um das Setzen des Fokus' bei der nächsten Anwahl der Applikation⁴ kümmern. Gerade diese Eigenschaft bisher nur wenig unterstützt.

4.4.2 Rollenvergabe

Um die Funktion – also die Rolle im Kontext – eines Elements oder einer Elementgruppe genauer zu spezifizieren, führt die [WAI ARIA](#) das `role` Attribut ein. Es gibt verschiedene Werte für das Attribut `role`, die auf verschiedene Situationen und Anwendungen zugeschnitten sind.

Die folgenden Auflistungen, die nur einen groben Überblick über die Möglichkeiten geben sollen, beziehen sich auf die WAI-ARIA 1.0⁵. Eine genauere Erläuterung zu den einzelnen Rollen wird bei den jeweiligen umzusetzenden Applikationen vorgenommen.

1 vgl. WAI-ARIA Best Practices - 3.1.2. Keyboard Navigation between Widgets [WAI \[2009f\]](#)

2 vgl. Introduction to WAI ARIA - Keyboard Navigation [Lemon \[2009\]](#)

3 Identifikator; in Bezug auf HTML und CSS das eindeutig identifizierende Attribut

4 vgl. WAI-ARIA Best Practices - 3.1.3 [WAI \[2009f\]](#)

5 WAI-ARIA 1.0 [WAI \[2009a\]](#)

User Input Widgets

Die Rollen `checkbox`, `combobox`, `input (abstract role)`, `listbox`, `option`, `radio`, `radiogroup`, `range` (Abstrakte Rolle), `select` (Abstrakte Rolle), `slider`, `spinbutton` und `textbox` dienen zur Beschreibung von Benutzeroberflächen für Nutzereingaben.

User Interface Element roles

Die Rollen `button`, `link`, `menu`, `menubar`, `menuitem`, `menuitemcheckbox`, `menuitemradio`, `tablist`, `tabpanel`, `tab`, `toolbar`, `tooltip`, `tree`, `treegrid` und `treeitem` dienen zur Beschreibung von graphischen Benutzeroberflächen.

Document Structure roles

Die verschiedenen Rollen `article`, `columnheader`, `definition`, `directory`, `document`, `grid`, `gridcell`, `group`, `heading`, `img`, `list`, `listitem`, `math`, `note`, `presentation`, `region`, `row`, `rowheader`, `section` (Abstrakte Rolle), `sectionhead` (Abstrakte Rolle) und `separator` helfen bei der Beschreibung der Struktur einer Webseite. Sie sind im Allgemeinen nicht interaktiv.

Application Structure

Die Rollen `alert`, `alertdialog`, `dialog`, `log`, `marquee`, `progressbar`, `status` und `time` sind für spezielle, eigenständige Applikationen gedacht.

Landmark roles

Die Rollen `application`, `banner`, `complementary`, `contentinfo`, `main`, `navigation` und `search` dienen zum Einteilen der Webseite in verschiedene Bereiche. Sie sollen dem Nutzer helfen, die gesuchten Inhalte zu finden.

Mit den Zuständen und Eigenschaften der einzelnen Rollen der `ARIA`-Spezifikation können der `AT` zusätzliche Informationen über die Applikation mitgeteilt werden. Aufgrund der Menge der Zustände und Eigenschaften erfolgt eine Erklärung dieser erst bei den jeweiligen umzusetzenden Applikationen. Hier sei aber noch angemerkt, dass es Eigenschaften gibt, die auf alle Elemente anwendbar sind (globale), und solche, die nur bei bestimmten Rollen eingesetzt werden dürfen (`widget`, `live region`, `drag-and-drop` und `relationship` Attribute).

Die vollständige Liste ist in der WAI-ARIA Spezifikation einzusehen¹.

¹ siehe WAI-ARIA - 2.2. WAI-ARIA States and Properties <http://www.w3.org/TR/wai-aria/>

4.4.3 Beziehungen von Elementen

Mit den zusätzlichen Attributen der [ARIA](#) können Beziehungen zwischen Elementen verdeutlicht werden.

Mit dem Attribut `aria-labelledby` kann ein Element eindeutig von einem anderen beschriftet werden. Es wird die [ID](#) des beschriftenden Elements eingetragen. Es erweitert so die Funktionalität, die zum Beispiel mit dem `label` Tag für Formularfelder schon möglich war.

Mit dem Attribut `aria-describedby` kann ein Element eindeutig von einem anderen beschrieben werden. Es wird die [ID](#) des beschreibenden Elements eingetragen. So kann zum Beispiel die Erklärung einer Grafik zum Schließen eines Dialogs mit der Grafik selbst verknüpft werden.

Ein weiteres wichtiges Attribut ist `aria-controls`, mit dem ein Element definiert werden kann, das ein anderes steuert. Es wird dem steuernden Element angefügt und erhält als Wert ebenfalls eine Liste mit den per Leerzeichen getrennten [IDs](#) der gesteuerten Elemente.

Um nicht durch das [DOM](#) ersichtliche hierarchische Beziehungen abzubilden, kann das Attribut `aria-owns` genutzt werden. Diesem sollten die [IDs](#) der verknüpften Elemente übergeben werden. Mit den dazugehörigen Attributen `aria-posinset` und `aria-setsize` kann die Position bzw. die Menge der Elemente festgelegt werden. Diese Technik sollte jedoch nur wenig genutzt werden, da es unter anderem möglich ist, die Eltern-Kind-Beziehung durch den [DOM](#), Standard [HTML](#)-Elemente und das `role` Attribut abzubilden.

Mit dem Attribut `aria-flowto` kann die Lesereihenfolge beeinflusst werden. Es soll als Wert die [ID](#) des jeweils nächsten Elements oder der nächsten Elemente eingetragen werden.

4.4.4 Live-Regionen

In modernen Applikationen werden oft [HTML](#)-Konstrukte zur Laufzeit in den bestehenden [DOM](#) eingebracht. Diese Veränderungen sind für die [AT](#) nicht ohne weiteres erkennbar, ohne den Fokus zu versetzen, was wiederum den Nutzer in seinem Arbeitsablauf stören würde. Dieser Problematik wird mit den den [ARIA Live-Regions](#) entgegengewirkt.

Das `aria-live` Attribut kann folgende Zustände haben, die die Priorität der Aktualisierung abbilden.

- `off` - der Standardwert; keine Änderung

- **polite** - keine Reaktion nötig, solange der derzeitige Vorgang nicht abgeschlossen ist
- **assertive** - höhere Priorität als **polite**, aber ebenfalls keine Reaktion nötig, solange der derzeitige Vorgang nicht abgeschlossen ist

Wenn eine *Live-Region* mehrere Aktualisierungen erhält, die einige Zeit in Anspruch nehmen, kann mit der Eigenschaft `aria-busy="true"` eine Verzögerung der Benachrichtigung durch die [AT](#) erwirkt werden. Nach Abschluß der Aktualisierung sollte die Eigenschaft wieder auf `false` gesetzt werden.

Um genauer spezifizieren zu können, ob alle Inhalte einer *Live-Region* aktualisiert wurden, kann man die Eigenschaft `aria-atomic` mit ihren Zuständen `true` oder `false` (Standard) angeben. Bei `true` werden alle Kindelemente als geändert an die [AT](#) gemeldet, es sei denn, ein Kindelement hebt die Einstellung mit `false` wieder auf.

Mit `aria-relevant` kann exakter angegeben werden, was genau in der *Live-Region* verändert wurde. Die Eigenschaft kann die Status `additions` für das Hinzufügen von Elementen, `removals` für das Entfernen von Elementen, `text` für das Ändern von Texinhalt (dazu zählen auch Äquivalente wie das `alt` Tag) und `all` für alle 3 Status zur gleichen Zeit haben. Die Information über das Entfernen oder das Hinzufügen eines oder mehrerer Kindelemente sollte eine informative Bedeutung für das Element haben.

In bestimmten Fällen (zum Beispiel bei ständiger Aktualisierung) kann es sinnvoll sein, auf die Verwendung von *Live-Regions* zu verzichten und stattdessen auf eine der sogenannten *Special Case Live Regions* zurückzugreifen¹.

4.4.5 Formular Eigenschaften

Mit einigen Eigenschaften und Zuständen können Benutzereingaben vereinfacht werden, indem ihr Status programmatisch auslesbar gemacht wird.

Zum Beispiel kann man mit dem Attribut `aria-valuemax` einen maximalen Wert bzw. per `aria-valuemin` einen minimalen Wert festlegen. Mit dem Attribut `aria-required` (mögliche Werte `true` oder `false`) lässt sich ein Pflichtfeld definieren und mit dem Attribut `aria-invalid` ein fehlerhaft oder nicht ausgefülltes Formularfeld markieren. Das Attribut `aria-invalid` kann den Wert `grammar` für einen grammatischen Fehler, `spelling` für einen Rechtschreibfehler oder den meistens verwendeten Wert `true` für einen allgemeinen Fehler annehmen. Der Wert `false` sollte als Standard-Wert nach erfolgter Korrektur gesetzt werden.

¹ angezeigt durch bestimmte Rollen wie alert, alertDialog, status, timer, marquee oder log; vgl. WAI-ARIA Best Practices - 5.3 [WAI \[2009f\]](#)

4.4.6 Drag und Drop

Um eine sogenannte Drag-and-Drop-Funktionalität barrierearm umzusetzen, bietet die [WAI ARIA](#) eigene Techniken an, auf die im folgenden nur kurz eingegangen wird, da sie für die vorliegende Arbeit wenig relevant sind.

Mit den Möglichkeiten von `aria-grabbed` und `aria-dropeffect` können die verschiedenen Status eines Drag-and-Drop-Prozess abgebildet und programmatisch auslesbar gemacht werden. Aufnehmbare Elemente müssen als solche mit `aria-grabbed` erkennbar gemacht werden und sollen auch mit der Tastatur bedienbar sein. Elemente, auf denen die aufgenommenen Elemente oder das aufgenommene Element fallengelassen werden können, sollen mit `aria-dropeffect` kenntlich gemacht werden. Mit den verschiedenen Werten des `aria-dropeffect` kann festgelegt werden, welche Aktion – wie Kopieren oder Verschieben – ausgeführt wird. Nach erfolgter Aktion sollten die Attribute wieder entfernt oder inaktiv gesetzt werden.

KAPITEL 5

Praktische Umsetzung

In diesem Kapitel werden zunächst allgemeine Vorgaben zur Programmierung getroffen. Das beinhaltet das Definieren von selbstaufgerlegten Programmierrichtlinien betreffend [HTML](#), [CSS](#) und [JS](#), aber auch grundsätzliche Vorgaben wie Aufbau und Struktur. Dies soll helfen eine einheitliche Beschaffenheit und einfache Verwendbarkeit zu schaffen.

Anschließend sollen die vier programmierten Applikationen (im folgenden Widgets genannt) ausführlich erläutert werden. Es handelt sich hierbei um ein Formular (bzw. dessen Live-Validierung), eine sog. Lightbox-Anwendung, ein Tabs-Applikation (Reiternavigation) und sortierbare Tabellen. Letzteres Widget wurde als Ersatz für ein Accordion-Widget gewählt, da dieses bereits von anderer Seite umgesetzt wurde (siehe auch Punkt [3.2](#)). Diese Widgets wurden deshalb gewählt, da ihre Design-Pattern mittlerweile auf den meisten modernen Internetseiten zu finden sind.

Zunächst soll im Punkt Definition jedes Widget und seine Funktionen genau definiert werden, um sie dann im Punkt Umsetzung zu spezifizieren. Das heißt, es werden mit Hilfe der Kriterien aus den [WAI WCAG](#) 2.0 Richtlinien und der [WAI ARIA](#) Spezifikation die genauen Eigenschaften der zu programmierenden Applikation festgelegt. Abschließend wird das entwickelte Widget dokumentiert. Innerhalb des Punktes Dokumentation wird der Aufruf, Ablauf und Aufbau erklärt und Übersichten zu Events, Methoden und Optionen bereitgestellt. Auch auf eventuell nötige Anpassungen für Kundenprojekte wird eingegangen.

5.1 Programmierrichtlinien

5.1.1 Grundsätzliches

Die in dieser Arbeit entwickelten Widgets sollen später für kundenspezifische Anforderungen genutzt werden. Sie sollen funktional den Mindestanforderungen der Design-Pattern entsprechen und möglichst modular aufgebaut sein. Das heißt, sie müssen alle notwendigen Funktionen abdecken und einfach erweiterbar und anpassbar sein. Gleichzeitig sollen

sie keine nicht zur Grundfunktionalität benötigten Funktionen haben, um eine kleine Dateigröße und Übersichtlichkeit des Quellcodes zu gewährleisten. Es soll verhindert werden, dass die Applikationen aufgrund zu vieler Optionen nicht eingesetzt werden, da die Einarbeitung für Kundenprojekte zu lange dauert.

Die Dokumentation erfolgt in deutscher Sprache. Der Quellcode wird nach ausführlich in englisch kommentiert und mit einfachem Tabulator-Zeichen eingerückt sein. Dies stellt die Dateigröße betreffend kein Problem dar, da der Quellcode komprimiert wird und so die in einer Produktivumgebung überflüssigen Kommentare und Abstände entfernt werden. Eine solche, sogenannte minifizierte Datei, wird erstellt.

Um eine bessere Lesbarkeit zu erreichen, werden nicht die deutschen Übersetzungen für [HTML](#)-Elemente, sondern an die Namensgebung bei SelfHTML¹ angelehnte Namen genutzt.

Damit die Widgets in Kundenprojekten sinnvoll einsetzbar sind, sollen diese mehrsprachig konfigurierbar sein.

Die Arbeit bezieht sich auf die jeweiligen *stable* Versionen vom 12. Mai 2009:

[jQuery v.1.3.2](#)

[jQuery UI v1.7.1](#)

Die Arbeit soll konform zur [WCAG](#) 2.0 (siehe Punkt [4.3](#)) und [ARIA](#) (siehe Punkt [4.4](#)) umgesetzt werden.

5.1.2 HTML

Es wird valides [XHTML](#) 1.1² verwendet. Um möglichst nah an den Standard-[HTML](#)-Elementen zu arbeiten, werden nach Möglichkeit diese verwendet und per Java-Script modifiziert und erweitert. Klassennamen werden wie unter Punkt [5.1.4](#) erläutert eingesetzt.

Die Elemente sollen in einer semantisch korrekten Reihenfolge angeordnet sein, damit derTabIndex und der Fokus korrekt gesetzt werden³. Dies wird durch korrekt sortierte Elemente im [DOM](#) erreicht.

1 Das Standard Nachschlagewerk für HTML; <http://de.selfhtml.org/>

2 <http://www.w3.org/TR/xhtml11/>

3 vgl. Techniques for WCAG 2.0 - G59, H4, C27 WAI [2009c]

5.1.3 Java-Script

Die Namenskonvention wird angelehnt an die ungarische Notation¹ erfolgen. Das heißt, dass die Variablen nach dem Muster `preFixVariablenName` vergeben werden. Die Variablennamen sollen selbsterklärend benannt werden, dabei aber von der Länge her lesbar bleiben. Dies stellt kein Problem in Bezug auf die Dateigröße dar, da die Script Dateien zum Einsatz in einer Produktivumgebung komprimiert werden und somit auch die Variablennamen gekürzt werden.

Des weiteren werden die Konventionen des jQuery Developer Guides² beachtet.

Die Programmierung der Widgets erfolgt in der für moderne Widgetentwicklung gebräuchlichen Art unter Verwendung der *widget factory*³.

Hierbei werden wenn möglich die Standard Methoden zur Verwendung kommen:

- `destroy()` - zum Entfernen der erzeugten Instanz, inklusive aller gesetzten Attribute, Klassen und `HTML`-Elemente
- `getData/setData/data` - zum Hinzufügen bzw. Entfernen von Daten für die Instanz
- `enable()` - aktiviert die Instanz; setzt `disable()` auf *false*
- `disable()` - deaktiviert die Instanz; setzt `disable()` auf *true*

Diese Methoden sollen teilweise ergänzt werden.

Um eine einfache Erweiterung möglich zu machen, werden an sinnvollen Punkten Rückruffunktionen, sogenannte *Callbacks*, implementiert.

5.1.4 Cascading Stylesheets

Die `CSS`-Klassen des jQuery UI CSS Frameworks⁴ sind zu nutzen, um eine größtmögliche Wiederverwendbarkeit zu gewährleisten. Neue Klassennamen werden anlehnd an die Konventionen⁵ erstellt. Dass bedeutet das alle Klassen mit `.ui-` beginnen und darauf der Widget Name und Verwendungszweck folgen: `.ui-widgetname-content`. Es werden keine globalen Angaben gemacht und keine IDs zur Gestaltung verwendet. Des weiteren soll zur Worttrennung die Verwendung von Bindestrichen der von Unterstrichen vorgezogen werden. Alle Klassennamen sind in Kleinschreibung zu verfassen.

Das Aussehen der Widgets soll möglichst einfach gehalten werden. Hierfür wird der

1 http://de.wikipedia.org/wiki/Ungarische_Notation

2 vgl: UI/Developer Guide Resig u. the jQuery Team [2009]

3 vgl: UI/Developer Guide Resig u. the jQuery Team [2009]

4 vgl: UI/Theming/API Bakaus u. the jQuery UI Team. [2009a]

5 vgl: UI/Theming/CustomThemes Bakaus u. the jQuery UI Team. [2009b]

Standard-Theme *UI Lightness* von jQuery UI verwendet.

Um die Skalierbarkeit der Widgets auch innerhalb älterer Browser zu gewährleisten, sollen so weit wie möglich relative Werte in `em` für Größenangaben genutzt werden¹.

5.2 Formular

5.2.1 Definition

Die sogenannte Live-Validierung, also die Formularfeld-Validierung während der Eingabe, soll hier umgesetzt werden. Es wird eine clientseitige, dass heißt im Browser per `JS` stattfindende Validierung von Formularfeldern umgesetzt. Es werden keine Abhängigkeiten der Formularelemente voneinander programmiert. Ausnahmen stellen hier Passwortfelder dar, die eine Überprüfung auf gleichen Inhalt benötigen.

Folgende Formularelemente sollen mit den aufgeführten Funktionen umgesetzt werden:

Einzelige Eingabefelder

Diese sollen sich als Pflichtfeld definieren lassen. Des weiteren soll es möglich sein, eine minimale und/oder maximale Anzahl an Zeichen zu definieren. Um eine Überprüfung nach bestimmten Mustern möglich zu machen, soll außerdem ein frei definierbares `RegEx`²-Pattern angegeben werden können oder folgende vordefinierte Prüfmuster:

- natürliche Zahl
- Dezimalzahl (deutsch mit Komma)
- Dezimalzahl (ISO mit Punkt)
- [URL](#)
- Email-Adresse
- Postleitzahl (deutsch)
- Datum (deutsch DD.MM.YYYY bzw. DD.MM.YY)
- Datum (ISO YYYY-MM-DD)
- Captcha (siehe Punkt [5.2.2](#))

¹ vgl: WCAG 2.0 - 1.4.4 Resize text [WAI \[2009\]](#)

² Regular Expression; Regulärer Ausdruck

Mehrzeilige Eingabebereiche

Diese sollen sich ebenfalls als Pflichtfeld definieren lassen. Eine minimale und/oder maximale Anzahl an Zeichen soll definier- und überprüfbar sein.

Passwortfelder

Auch diese sollen sich als Pflichtfeld definieren lassen. Hier soll ein Abgleich mit einem anderen Passwortfeld verfügbar sein, um zu Überprüfen ob die eingegebene Zeichenfolge die gleiche ist. Eine minimale und/oder maximale Anzahl an Zeichen soll definier- und überprüfbar sein.

Checkboxen

Da es sich hierbei um ein Element handelt, dass nur zwei Status kennt, soll es sich lediglich als Pflichtfeld definieren lassen. Bei mehreren Checkboxen soll ebenfalls eine minimale und/oder maximale Anzahl an aktivierten Checkboxen definiert werden können.

Radio-Buttons

Diese sollen wiederum als Pflichtfeld definierbar sein.

Auswahllisten

Auch hier soll eine Definition als Pflichtfeld erfolgen können. Bei Auswahllisten mit Mehrfachauswahl soll ebenfalls eine minimale und/oder maximale Anzahl an aktivierte Einträgen definiert werden können.

Felder für Datei-Upload

Diese sollen ebenfalls die Option Pflichtfeld aufweisen. Eine Überprüfung des Dateityps soll anhand eines [RegEx](#)-Pattern möglich sein.

5.2.2 Umsetzung

An dieser Stelle sei angemerkt, dass eine serverseitige Validierung und Fehlerausgabe aus Gründen der Sicherheit und der Kompatibilität mit nicht [JS](#) unterstützenden Ausgabegeräten in jedem Fall nötig ist.

Um von Anfang an eine hohe Barrierefreit zu erreichen, werden die Formularfelder mit

Standard-[HTML](#)-Elementen mit entsprechend (auch visuell) verknüpften Beschriftungen erstellt¹, korrekt mit `fieldset` Tag gruppiert und per `legend` beschriftet². Diese Standard-[HTML](#)-Formular-Elemente sind bereits tastaturbedienbar³.

Alle Elemente sollen mit der Textgröße skalierbar sein⁴.

Damit Pflichtfelder, die der Konvention nach mit einem Asterisk (*) gekennzeichnet werden, auch für [AT](#) erkennbar sind, werden diese semantisch hervorgehoben⁵ und zusätzlich mit einem `title` Tag ausgestattet⁶. Um gute Nutzbarkeit zu erreichen, wird ein Element, das Fokus erhält, besser sichtbar markiert. Der Standard-Fokus, für gewöhnlich eine dünne gepunktete Linie, soll also ersetzt werden⁷. Dies stellt hauptsächlich im [IE](#)⁸ 6 und 7 ein Problem dar, da diese keine [CSS](#)-Pseudo-Klassen beherrschen⁹. Diese Hervorhebung wird für alle Browser mit Java-Script-Ereignissen umgesetzt.

Um die Nutzung barrierefrei zu halten, soll eine Änderung des Kontextes nur durch Nutzerinteraktion hervorgerufen¹⁰ werden. Eine Änderung des Kontextes schließt die Anzeige von Fehlern nicht ein.

Vor dem endgültigen Absenden sollte die Möglichkeit bestehen, die Daten zu kontrollieren und gegebenenfalls zu ändern oder den Vorgang abzubrechen¹¹. Dies wird hier durch das Hinzufügen einer Checkbox zur Bestätigung erreicht.¹²

Fehlermeldungen

Bei der sogenannten Live-Validierung werden Informationen über falsch oder nicht ausgefüllte Felder während bzw. kurz nach der Eingabe durch Manipulationen am [DOM](#) angezeigt¹³. Diese Live-Validierung muss abschaltbar sein¹⁴.

Fehlermeldungen und daraus resultierende Hervorhebungen werden in Textform, semantisch und als visueller Hinweis kenntlich gemacht¹⁵. Mit Hilfe des Attributs `aria-required`

1 vgl. Techniques for WCAG 2.0 - H44, G162, H44 [WAI \[2009c\]](#)

2 vgl. Techniques for WCAG 2.0 - H71 [WAI \[2009c\]](#)

3 vgl. Techniques for WCAG 2.0 - H91, SCR35, [WAI \[2009c\]](#)

4 vgl. Techniques for WCAG 2.0 - C14, SCR34, G146, G179, C28, G142 [WAI \[2009c\]](#)

5 vgl. Techniques for WCAG 2.0 - H49 [WAI \[2009c\]](#)

6 vgl. Einfach für Alle - Formular Design - Konventionen für Pflichtfelder [Caspers \[2009b\]](#)

7 vgl. Techniques for WCAG 2.0 - C15, G149, G195, G165, SCR31 [WAI \[2009c\]](#)

8 Microsoft Internet Explorer

9 vgl. MSDN Library - CSS Improvements in Internet Explorer 8 [http://msdn.microsoft.com/en-us/library/cc304082\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/cc304082(VS.85).aspx) Corporation [2009b]

10 vgl. Techniques for WCAG 2.0 - G80 [WAI \[2009c\]](#)

11 vgl. Techniques for WCAG 2.0 - G98 [WAI \[2009c\]](#)

12 vgl. Techniques for WCAG 2.0 - G155 [WAI \[2009c\]](#)

13 vgl. Techniques for WCAG 2.0 - SCR18, SCR32 [WAI \[2009c\]](#)

14 vgl. Techniques for WCAG 2.0 - SCR14, G75 [WAI \[2009c\]](#)

15 vgl. Techniques for WCAG 2.0 - G83, G138, G122, G182, G14 [WAI \[2009c\]](#)

soll ein Pflichtfeld definiert und mit `aria-invalid` ein fehlerhaft oder nicht ausgefülltes Formularfeld markiert werden¹.

Die Fehlermeldung sollen Sprungmarken anbieten, um zum betroffenen Feld zu gelangen².

Fehlermeldungen sollen möglichst genau spezifizieren, was falsch eingegeben wurde, wenn möglich eine Lösung anbieten und dem Nutzer die Möglichkeit bieten, direkt zum betroffenen Element zu springen³. Das Anbieten einer möglichen Fehlerkorrektur durch einen Algorithmus würde den Rahmen dieser Arbeit sprengen.

Da bei der Ausgabe der Fehlermeldungen eine Manipulation am `DOM` vorgenommen wird, soll diese mit Techniken der [WAI ARIA](#) kenntlich gemacht werden. In diesem Fall wird auf die Verwendung von sogenannter *Live-Regions* zurückgegriffen. Weitere Informationen hierzu finden sich in Punkt [4.4.4](#) dieser Arbeit.

Die Fehlermeldung bzw. Erfolgsmeldung soll nach dem Absenden fokussiert werden, um ältere Screenreader darauf aufmerksam zu machen, dass sich etwas im `DOM` geändert hat.

Captcha

Da ein [Captcha](#)⁴ oftmals die einzige Möglichkeit darstellt, eine unerwünschte Nutzung eines Formulars zu verhindern, muss es bei dem Widget möglich sein, ein solches zu verwenden. Gerade hier muss die barrierefreie Gestaltung beachtet werden, da [Captchas](#) oft Barrieren für Menschen mit Behinderung darstellen⁵, auch wenn es mittlerweile teilweise Lösungen wie [WebVisum](#)⁶ gibt.

Da eine Vielzahl von Lösungen existiert und oftmals in einem Kundenprojekt bereits eine bestimmte Technik verwendet wird, soll an dieser Stelle auf eine spezielle Lösung verzichtet und stattdessen eine Schnittstelle angeboten werden. Diese wird als ein Callback angeboten, mit dessen Hilfe zum Beispiel ein [AJAX](#)-Aufruf genutzt werden kann, um die Eingabe serverseitig zu kontrollieren.

In jedem Fall muss das [Captcha](#) barrierefrei konzipiert sein. Das kann je nach Art des [Captcha](#) ein Problem darstellen, das sich aber zum Beispiel durch Anbieten einer akustischen Alternative entschärfen lässt. Die Erstellung einer Lösung für ein barrierefreies [Captcha](#) gehört nicht zum Umfang dieser Arbeit.

1 vgl. Techniques for WCAG 2.0 - ARIA2, ARIA3, ARIA4 [WAI \[2009c\]](#)

2 vgl. Techniques for WCAG 2.0 - G139 [WAI \[2009c\]](#)

3 vgl. Techniques for WCAG 2.0 - G85, G84, G177 [WAI \[2009c\]](#)

4 Completely Automated Public Turing test to tell Computers and Humans Apart

5 vgl. Inaccessibility of CAPTCHA [May \[2009\]](#)

6 ein Firefox-Addon, das verschiedene Hilfestellungen für Menschen mit Behinderung anbietet, darunter auch das Lösen von Captchas; <http://www.webvisum.com/>; siehe auch <http://www.marcozehe.de/2008/07/03/review-of-the-webvisum-firefox-extension/>

Datei-Upload

Da mit einem [AJAX](#)-Aufruf keine Dateien übermittelt werden können, sollen verschiedene Arten der Formularübertragung möglich sein.

Per Parameter soll konfigurierbar sein, ob das native Absenden oder ein [AJAX](#)-Aufruf zur Formularübermittlung genutzt wird. Beim nativen Absenden wird das Formular an die im `form` Tag angegebene Adresse gesendet und der Nutzer an diese weitergeleitet. Bei einem [AJAX](#)-Aufruf wird das Formular abgesendet und als Ergebnis vom Server ein `true` für das erfolgreiche Absenden bzw. eine Fehlermeldung zurückgegeben.

Wenn der Nutzer nicht weitergeleitet werden und trotzdem ein Datei-Upload möglich sein soll, muss ein Workaround genutzt werden. Bei diesem wird in den bestehenden [DOM](#) ein `iframe` injiziert, um das Formular in dieses zu senden.

5.2.3 Dokumentation

Der gesamte Quellcode ist im Anhang einzusehen:

[B.1.1](#) HTML

[B.1.2](#) Java-Script

HTML und CSS

Das Formular benötigt in seinem `form` Tag eine [ID](#), welche dem Widget übergeben wird. Allen Formular-Elementen muss eine eindeutige ID zugewiesen werden. Ausnahme bilden hier Checkboxen (auch einzelne) und Radio-Buttons, welche zur Funktion des Widgets keine [ID](#), aber ein `name` Attribut benötigen. Dieses muss für Gruppen gleich sein.

Einige im [DOM](#) definierte [IDs](#) werden vom Widget genutzt, um Manipulationen vorzunehmen. Zu diesen gehören `ui-formular-info` zum Anzeigen des Links, der die Live-Validierung optional macht, und das `div` mit der [ID](#) `ui-formular-error`, in das die Fehlermeldungen geschrieben werden.

Im Beispiel-Quellcode wurden die `label` Elemente im [DOM](#) jeweils vor das dazugehörige Formularelement gesetzt. Wenn eine andere Anordnung nötig ist, müssen im Quellcode-Änderungen vorgenommen werden (weitere Informationen im nachfolgenden Teil [5.2.3](#)). Obwohl laut Spezifikation zulässig, sollte die implizierte Angabe, also die Verschachtelung von `input` Elementen in den dazugehörigen `label` Elementen, vermieden werden, da einige [AT](#) damit Probleme haben¹. Das Widget ist für eine solche Verschachtelung nicht vorgesehen, lässt sich jedoch anpassen.

¹ vgl. Techniques for WCAG 2.0 - H44, G162, H44 [WAI \[2009c\]](#) und Einfach für Alle - HTML Elemente - Labels in HTML [Caspers \[2009b\]](#)

Für die verschiedenen Status der Formularelemente und ihrer Beschriftungen wurden die `ui-state` CSS Klassen der jQuery UI Themes verwendet. Nähere Informationen hierzu unter Punkt 5.1.4.

Aufruf

Das erstellte Plugin wird wie alle normalen jQuery UI Widgets aufgerufen. Die Besonderheit liegt in der Option `forms`, welche zur Funktion des Widgets ausgefüllt werden muss. Diese Option ist ein Array und enthält wiederum ein Array mit der ID des zugehörigen Formularelements. Innerhalb dieses Arrays müssen zwei weitere Arrays definiert werden: `rules`, das die zu kontrollierenden Regeln enthält, und `msg`, welches die dazu passenden Fehlertexte enthält.

Das `forms` Array muss mit seinen Arrays `rules` und `msg` vollständig definiert sein, damit das Widget korrekt funktioniert.

Es ist nicht nötig, jedes Formularelement zu definieren. Nicht definierte Elemente werden ebenfalls versendet.

```
1 <script type="text/javascript" src="js/jquery.js"></script>
2 <script type="text/javascript" src="js/ui.core.js"></script>
3 <script type="text/javascript" src="js/ui.formValidator.js"></script>
4 <script type="text/javascript">
5     $(function() {
6         var formular = $("#form").formValidator({
7             forms: {
8                 inputtext: {
9                     rules: {
10                         required: true,
11                         regEx: "email"
12                     },
13                     msg: {
14                         required: "Der einzeilige Eingabebereich muss
15                             ausgefüllt werden.",
16                         regEx: "Das einzeilige Eingabefeld muss eine Email
17                             Adresse enthalten."
18                     }
19                 },
20                 txtarea: {
21                     rules: {
22                         required: true,
23                         lengthMin: 2,
24                         lengthMax: 10
25                     },
26                     msg: {
27                         required: "Der mehrzeilige Eingabebereich muss
28                             ausgefüllt werden."
29                     }
30                 }
31             }
32         })
33     })
34 
```

Listing 5.1: formValidator: 113-137: Aufruf des Widgets

Im Listing steht `inputtext` für die ID des einzeiligen Eingabefeldes. Die Regel und ihre Fehlernachricht müssen gleich benannt werden. Für `lengthMax` und `lengthMin` gibt es nur eine Fehlernachricht.

Folgende Regeln sind für die verschiedenen Formularelemente zulässig:

Regel	Werte	mögliche Formularelemente
required	Boolean	text, textarea, checkbox, radio, select, file
lengthMin	Number	text, textarea, checkbox, radio, select
lengthMax	Number	text, textarea, checkbox, radio, select
equalTo	String	text, textarea
regEx	String	text, textarea, file

Tabelle 5.1: formValidator: Übersicht Regeln

Weitere Informationen sind auch in der Definition dieses Widgets unter Punkt [5.2.1](#) zu finden.

Bei Gruppen von Radio-Buttons und Checkboxen sowie Selectfeldern bezieht sich `lengthMin` und `lengthMax` auf die Menge der selektierten Einträge.

Die Regel `regEx` ermöglicht das Kontrollieren von Formularfeldern per regulärem Ausdruck. Hierfür sind folgende vordefinierte Ausdrücke hinterlegt:

Wert	Validierung auf	Beispiel
number	natürliche Zahlen	12345
numberDE	Zahlen bzw. Kommazahlen	1,23 oder 123
numberISO	Zahlen bzw. Zahlen mit Punkt	1.23 oder 123
email	valide Email Adresse	test@domain.de
url	valide URL	http://www.domain.de
plz	valide deutsche Postleitzahl	20457
dateDE	deutsches Datum	01.02.09 oder 1.2.2009
dateISO	internationales Datum	2009-02-01 oder 2009-2-1
captcha	String	nicht definiert; ruft Callback auf

Tabelle 5.2: formValidator: Übersicht regulärer Ausdrücke

Bei nicht bekanntem Schlüsselwort wird der String als regulärer Ausdruck genutzt. Beim Schlüsselwort `captcha` wird ein Callback aufgerufen, der serverseitig die Validität kontrolliert.

Optionen

Um das Widget zu konfigurieren können folgende Optionen gesetzt werden:

Name	Typ	Standard
validateLive	Boolean	true
validateTimeout	Number	500
validateOff	String	Bitte klicken Sie hier, um die Live-Validierung zu deaktivieren.
validateOn	String	Bitte klicken Sie hier, um die Live-Validierung zu aktivieren.
errorsTitle	String	Bitte korrigieren Sie folgende Fehler:
submitHowTo	String	ajax
submitUrl	String	(wenn keine URL angegeben ist, wird die im Formular definierte action genutzt.)
submitError	String	Bei der Datenübertragung ist ein Fehler aufgetreten. Entschuldigen Sie bitte und versuchen Sie es noch einmal.
submitSuccess	String	Die Daten wurden erfolgreich übermittelt. Vielen Dank!
disabled	Boolean	false
Pflichtangaben		
forms	Array	[]

Tabelle 5.3: formValidator: Übersicht Optionen

Ob eine Validierung in Echtzeit passieren soll und wie schnell diese nach Beenden des Eintipps angestoßen wird, kann per validateLive bzw. validateTimeout (in Millisekunden) eingestellt werden. Die Optionen validateOff und validateOn definieren die verlinkten Texte, mit denen der Nutzer die Live-Validierung an- bzw. abschalten kann.

Um die Absende-Methode zu konfigurieren kann die Option submitHowTo entweder auf post (lässt das native Absenden des Formulars aus), ajax (startet einen **AJAX**-Request) oder auf iframe (wie ajax ohne Weiterleitung, aber mit Möglichkeit des Datei-Uploads) gesetzt werden¹. submitError und submitSuccess definieren die Fehler- oder Erfolgsmeldung nach dem Absenden.

Mit der Option disabled kann das Widget deaktiviert werden.

Events

Events oder auch Callbacks sind vordefinierte Aufrufe zu Funktionen, mit denen sich eigene Programmteile in den Ablauf des Widgets integrieren lassen. Folgende Events

1 siehe auch Punkt 5.2.2

sind definiert:

Name	Erklärung
onInit	wird ausgelöst, wenn Initialisierung abgeschlossen
onFormSubmitted	wird vor der Validierung, aber nach dem Absenden des Formulars ausgelöst
onShowErrors	Wird ausgelöst, nachdem die Fehler angezeigt bzw. aktualisiert wurden
onShowSuccess	wird ausgelöst, nachdem die Erfolgs- oder Fehlermeldung angezeigt wurde
checkCaptcha	wird aufgerufen, wenn als <code>regEx: "captcha"</code> angegeben wurde; muss Boolean Wert zurückliefern.

Tabelle 5.4: formValidator: Übersicht Events

Die Callback-Funktion `checkCaptcha` stellt hierbei etwas besonderes dar: Sie wird aufgerufen, wenn einem Formularelement die Regel `regEx` mit dem Wert `captcha` zugewiesen wurde. Der Callback wird mit dem eingegebenen Wert des Formularelements aufgerufen. So kann zum Beispiel ein [AJAX](#) Request an den Server erfolgen, um die Korrektheit zu überprüfen. Die Rückruffunktion muss einen Boolean Wert zurückliefern.

```

1      msg: {
2          required: "Bitte kontrollieren Sie die Daten."
3      },
4      captcha: {
5          rules: {
6              required: true,
7              regEx: "captcha"
8          },
9          msg: {
10

```

Listing 5.2: formValidator: 205-214: checkCaptcha Callback-Funktion

Das Listing zeigt die vereinfachte Verwendung im Beispiel Quelltext.

Methoden

Um das Widget von außen anzusteuern, können verschiedene Methoden genutzt werden, die in der folgenden Tabelle aufgelistet sind.

Name	Aufruf	Erklärung
disable	.formValidator('disable')	deaktiviert das Widget
destroy	.formValidator('destroy')	entfernt komplett die Funktionalität und setzt das HTML zurück in den Ausgangszustand
enable	.formValidator('enable')	aktiviert das Widget
formSubmitted	.formValidator('formSubmitted')	sendet das Formular ab
option	.formValidator('option' , optionName , [value])	fungiert als Setter bzw. Getter, wenn kein Wert angegeben wird

Tabelle 5.5: formValidator: Übersicht Methoden

Anhand der Standard Methode `destroy` wird die Funktionsweise klar:

```

1      return ( value == "123456" ) ? true : false ;
2
3  };
```

Listing 5.3: formValidator: 231-233: destroy Methode

Das Element mit der ID `#destroy` steuert in diesem Fall einen einfachen Textlink an. Auf die Variable `formular` wurde zuvor das initialisierte Widget gelegt.

Funktionsweise

Im nachfolgenden werden die Funktionsweise und der Ablauf des Widgets erläutert. Das Schaubild auf Seite 72 soll verdeutlichen, welche Funktionen beim Absenden des Formulares von wo aus aufgerufen werden.

Die `_init` Funktion wird automatisch zu Beginn aufgerufen und initialisiert das Widget. Hier wird das Standard-Event des Absende-Buttons abgefangen und bei aktiverter Live-Validierung auch das Aktivieren bzw. Tippen in die Formularelemente mit der entsprechenden Funktion (`_validator`) verbunden.

Das Widget durchläuft hier alle definierten Elemente, um diese nach Einzelement oder Gruppe einzurichten und diese Informationen abzuspeichern. Des weiteren wird für jedes definierte Element ein assoziatives Array im `errorArray` initialisiert (in dem später der Status des Feldes abgespeichert wird) und das Element selbst unter `options.forms[id].element` abgespeichert. Beim Durchlaufen der Elemente wird, falls als Regel `required` definiert wurde, das `aria-required` Attribut gesetzt.

Die `hover` und `focus` Events auf die Elemente werden ebenfalls hier gesetzt. Diese Events müssen bei abweichender Anordnung der `label` Elemente verändert werden.

Wenn der Nutzer den Absende-Button anklickt, wird die Funktion `formSubmitted`

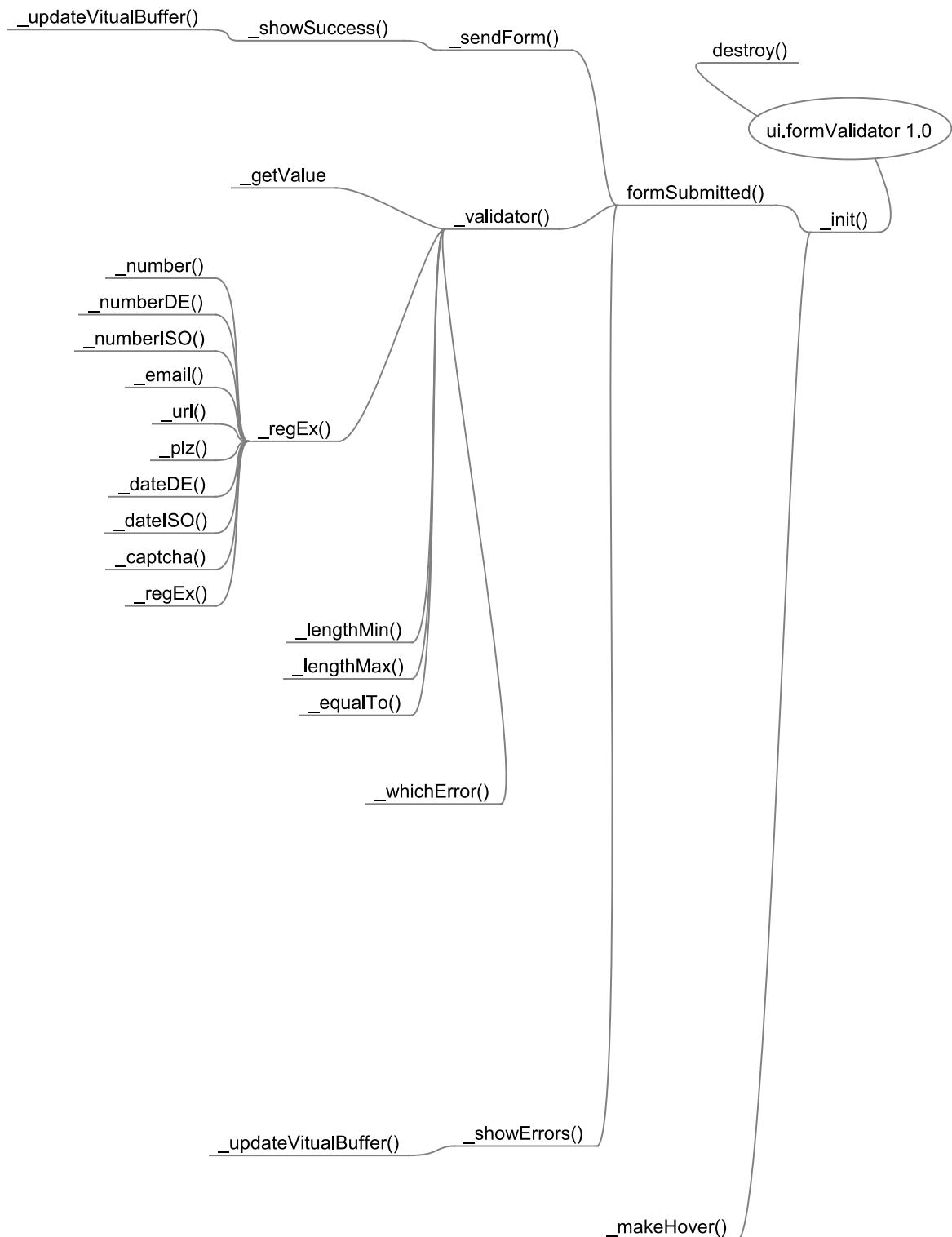


Abbildung 5.1: formValidator: Funktionsübersicht

aufgerufen, die wiederum für jedes im Array `forms` definierte Element die Funktion `_validator` aufruft. Danach wird zuerst mit Hilfe der Funktion `_getValue` der Inhalt des Elements ausgelesen. Bei einzeiligen und mehrzeiligen Textfeldern sowie Upload-Feldern ist dies der eingegebene Inhalt, wohingegen bei Checkboxen und Radio-Buttons die Menge der aktivierten Elemente zurückgegeben wird. Dann geht die Funktion alle verfügbaren Regeln für das übergebene Element durch und validiert diese. Dies geschieht mit dem Aufruf der passenden Validierungsfunktion wie zum Beispiel `_email`. Die Validierungsfunktionen liefern jeweils einen Boolean Wert zurück, der dann mit Hilfe der Funktion `_whichError` genauer klassifiziert wird. Ein Element kann drei Status besitzen: `new`, `old` und `corrected`. Diese sind später zur korrekten ARIA-Verwendung nötig und werden in das `errorArray` abgespeichert.

Wenn die Validierung der Elemente abgeschlossen ist, wird die Funktion `_showErrors` aufgerufen, um die registrierten Fehler anzuzeigen. Bei der Live-Validierung wird durch Tippen oder Anklicken eines Elementes zuerst die `_validator` Funktion und dann die `_showErrors` Funktion aufgerufen.

Die `_showErrors` Funktion verbindet die im `errorArray` abgespeicherten Fehler für jedes Element mit den entsprechenden Fehlermeldungen, kontrolliert, welche Fehler neu oder alt sind oder korrigiert wurden, und zeigt daraufhin die Fehler per DOM-Manipulation an. Je nach Fehler-Status werden die *Live-Regions* und `aria-invalid` Attribute gesetzt.

Nachdem die Fehlermeldung injiziert wurde, werden die Links mit den Formularelementen verknüpft.

Wenn keine Fehler gefunden wurden, wird die Funktion `_sendForm` zum Absenden der Daten aufgerufen.

Je nach definierter `submitHowTo` Option wird eine andere Möglichkeit der Datenübermittlung aufgerufen. Die native Sendeart und der herkömmliche `AJAX`-Request sollen hier nicht näher beschrieben werden. Um aber Datei-Upserts ohne eine Weiterleitung zu realisieren, muss ein `iframe` injiziert werden, in das die Daten gesendet werden. Nach dem Absenden der Daten wird die Rückmeldung des Servers an die `_showSuccess` Funktion weitergegeben.

Die `_showSuccess` Funktion nimmt drei verschiedene Status entgegen: `true` bzw. `false`, bei denen eine vorher definierte Erfolgsmeldung bzw. Fehlermeldung angezeigt wird, oder sie zeigt die vom Server zurückgegebene Fehlermeldung an. Je nachdem, ob die Übermittlung erfolgreich war oder nicht, wird ein definiertes Icon für die Meldung genutzt.

An diesem Punkt ist der Ablauf des Widgets abgeschlossen.

An verschiedenen Stellen wird die `_updateVirtualBuffer` Funktion aufgerufen. Diese sorgt durch Injizierung eines versteckten Formularfeldes und dessen Inhaltsänderung

für eine Aktualisierung des Virtuellen Buffers bei älteren Screenreadern. Dies ist immer dann nötig, wenn Änderungen am DOM vorgenommen wurden. Nähere Informationen hierzu in Punkt 3.1.1.

Anpassungen

An verschiedenen Stellen im Widget können Anpassungen vorgenommen werden, um evtl. Darstellungsänderungen auch im Script abzubilden. Im nachfolgenden werden Codepassagen gezeigt, an denen Änderungen vorgenommen werden können.

```

1   // add info Text and provide link to prevent live validating
2   if(options.validateLive && !options.disabled) {
3     // add the deactivate live validation message
4     self.element.find("#ui-formular-info").append("\t<p><a id=\"ui-
5       formular-live\" href=\"#nogo\">" + options.validateOff +"</a
6       ></p>\n\t");
7
8     // toggle live validating and text of the link
9     self.element.find("#ui-formular-live").toggle(
10       function () {
11         options.validateLive = false;
12         $(this).attr("aria-live","polite")
13           .attr("aria-relevant","text")
14           .html(options.validateOn);
15         self._updateVirtualBuffer();
16       },
17       function () {
18         options.validateLive = true;
19         $(this).attr("aria-live","polite")
20           .attr("aria-relevant","text")
21           .html(options.validateOff);
22         self._updateVirtualBuffer();
23       }
24     );
25   }

```

Listing 5.4: formValidator.js 82-106: Anpassungen Schalter Live-Validierung

Innerhalb der _init Funktion wird hier zuerst in das Element mit der ID ui-formular-info ein p Tag eingefügt, das einen Link zum Deaktivieren der Live-Validierungsfunktionalität enthält. Im zweiten Schritt wird auf diesen Link ein *toggle* Event gesetzt, mit dessen Hilfe die Option validateLive, der Text des Links und das aria-live Attribut geändert werden.

```

1   // go through every given form element
2   $.each(options.forms, function(id){
3     // instance the associative array with index = id of the form
4     // element
5     errors[id] = [];
6
7     // save element and which form type / add event handler / ARIA
8     var element = self.element.find("#"+id);
9     //check if radio group or checkbox group or single checkbox
10    if (!element.length) {
11      // get all group elements
12      element = self.element.find("input[name='"+id+"']");
13      // no element found? Only developers should see this
14      if (!element.length) {
15        alert("Error: Configuration corrupted!\n\nCan't find element
16        with id or name = "+id);
17      } else {
18        value = "group";
19        // change label class when hover the label
20        self._makeHover(element.next());
21        // change label class when hover the form element
22        element.bind("mouseenter", function(){ $(this).next().
23          addClass('ui-state-hover'); })
24          .bind("mouseleave", function(){ $(this).next().
25            removeClass('ui-state-hover'); })
26          .bind("focus", function(){ $(this).next().addClass('ui-
27            state-focus'); })
28          .bind("blur", function(){ $(this).next().removeClass('ui-
29            state-focus'); });
30      }
31    } else {
32      // form element hover
33      self._makeHover(element);
34    }
35  }
36}

```

Listing 5.5: formValidator.js 113-139: Anpassungen Hover/Focus Events

In diesem Listing wird ein Teil der `_init` Funktion gezeigt, in dem jedes im Array `forms` definierte Formularelement durchlaufen wird. Das Element mit der ID wird gesucht und dann per `self._makeHover(element);` mit den entsprechenden Events versehen. Wenn das Element mit der ID nicht gefunden werden kann, wird davon ausgegangen, dass es eine Gruppe von Radio-Button bzw. Checkboxen ist. Diese wird dann ebenfalls gesucht und mit `self._makeHover(element.next());` werden dann zunächst die `label` Elemente, die im DOM nach den Formularelementen angelegt sind, mit Events belegt. Dann werden auch die eigentlichen Formularelemente mit Events ausgestattet. Diese werden allerdings so übergeben, dass beim Überfahren einer Checkbox dessen `label` Element hervorgehoben wird.

```

1   for (var rule in errors[id]){
2     // set error as corrected
3     if (errors[id][rule] == "corrected") {
4       var target = options.forms[id].element;
5       // ARIA
6       target.attr("aria-invalid", false);
7       // check for radio group or checkbox group
8       if (options.forms[id].type == "group") {
9         target = target.next();
10    }
11    // unhighlight error field
12    target.removeClass("ui-state-error");
13    // ARIA: old error deleted
14    removeError = true;
15  }

```

Listing 5.6: formValidator.js 308-322: Anpassungen Fehlerklasse entfernen

Dieses Listing zeigt den Teil der `_showErrors` Funktion, der ausgelöst wird, wenn ein Fehler korrigiert wurde. Hier wird abgefragt, ob das Element eine Gruppe oder ein einzelnes Element ist, um je nach Typ die CSS-Klasse beim `label` Element oder beim eigentlichen Formularfeld zu entfernen. Da das Label im Beispiel nach dem Formular-element angeordnet ist, wird hier `target = target.next();` genutzt, um das Label anzusteuern.

```

1   if(errors[id][rule] == "new" || errors[id][rule] == "old") {
2     switch (rule) {
3       case "required":
4         msg = options.forms[id].msg.required;
5         break;
6       case "regEx":
7         msg = options.forms[id].msg.regEx;
8         break;
9       case "lengthMin":
10      msg = options.forms[id].msg.length;
11      break;
12      case "lengthMax":
13      msg = options.forms[id].msg.length;
14      break;
15      case "equalTo":
16      msg = options.forms[id].msg.equalTo;
17      break;
18    }
19    msgs += '          <li><a href="#'+id+'">'+msg+'</a></li>\n';
20    ;
21    // there are errors to show
22    isError = failure = true;

```

22 }

Listing 5.7: formValidator.js 323-344: Anpassungen Fehler sammeln

Im diesem Listing aus der `_showErrors` Funktion wird für jeden Fehler ein Listenelement mit Link erstellt und in der Variable `msgs` gespeichert. Die Fehlermeldungen werden in der Weiche je nach Fehler aus dem entsprechenden Array gezogen.

```

1   // check at last if there is an error so error class wont be
2   // removed
3   if (failure) {
4       var target = options.forms[id].element;
5       target.attr("aria-invalid", true);
6       // check for radio group or checkbox group
7       if (options.forms[id].type == "group") {
8           target = target.next();
9       }
10      // unhighlight error field
11      target.addClass("ui-state-error");
12  }
```

Listing 5.8: formValidator.js 350-360: Anpassungen Fehlerklasse hinzufügen

Dieses Listing zeigt einen Teil der `_showErrors` Funktion. Hier wird ebenfalls kontrolliert, ob ein Einzelelement oder eine Gruppe angesteuert wird. Danach wird entschieden, ob die Klasse beim `label` Element oder beim eigentlichen Element hinzugefügt wird. Auch hier wird ein `label` Element mit `target = target.next();` angesteuert.

```

1   // build up HTML / no content if no error is found
2   var html = "\n"
3   if (isError) {
4       html += '      <div '+aria+' class="info ui-state-highlight ui-
5           state ui-corner-all">+'\n";
6       html += '          <p id="ui-error-title">+'\n';
7       html += '              <span class="ui-icon ui-icon-alert" style="
8                   float: left; margin-right: 0.3em;"></span>+'\n";
9       html += '                  '+options.errorsTitle+'\n";
10      html += '          </p>+'\n';
11      html += '          <ul aria-labelledby="ui-error-title">+'\n';
12      html += msgs;
13      html += '              </ul>+'\n';
14      html += '      </div>+'\n\t\t';
15  }
// inject error HTML and make onclick event for direct error
// correction
errorElement = self.element.find("#ui-formular-error");
```

```
16     errorElement.html(html);
```

Listing 5.9: formValidator.js 369-384: Anpassungen Fehlermeldung injizieren

Im obigen Listing aus der `_showErrors` Funktion wird das **HTML**-Konstrukt zur späteren Injektion vorbereitet. Die Variable `aria` enthält die kompletten, zuvor zusammengefügten **ARIA**-Attribute. Die eigentlichen Fehlermeldungen werden hier in Form von `options.errorsTitle` als Überschrift und der Variablen `msgs` integriert. Abschließend wird das **HTML**-Konstrukt in das `div` mit der **ID ui-formular-error** injiziert. Deshalb sollte die **ID** beibehalten werden.

Wichtig ist in diesem Abschnitt, dass die Fehlermeldungen immer als Links definiert sind, da diese später mit Events versehen werden.

```
1 //build up HTML
2 var html = "\n"
3 html += '    <div id="ui-formular-success">'+"\n";
4 html += '        <div aria-live="assertive" class="info ui-state-
5         highlight ui-state ui-corner-all">'+"\n";
6 html += '            <p>'+"\n";
7 html += '                <span class="ui-icon ui-icon-"+icon+" style="
8                 float: left; margin-right: 0.3em;"></span>'+"\n";
9 html += '                    '+msg+"\n";
10 html += '                </p>'+"\n";
11 html += '            </div>'+"\n\t";
12 html += '        </div>'+"\n\t";
13 self.element.prepend(html);
14 self.element.find("#ui-formular-success").attr("tabindex",-1).
15     focus();
16 self._updateVirtualBuffer();
// Callback
17 self._trigger("onShowSuccess", 0);
```

Listing 5.10: formValidator.js 489-503: Anpassungen Erfolgsmeldung

Im diesem Listing aus der `_showSuccess` Funktion wird das **HTML**-Konstrukt mit der Variablen `msg` (enthält die Fehler bzw. Erfolgsmeldung) zur späteren Injektion vorbereitet. Die Variable `icon` dient nur zu Darstellungszwecken – sie steuert, welches Icon angezeigt wird.

Hier kann konfiguriert werden, wo die Meldung platziert werden soll und ob zum Beispiel bei erfolgreichem Datenversand das Formular ausgeblendet bzw. entfernt werden soll. Wichtig ist, dass **ui-formular-success** als **ID** beibehalten wird.

5.3 Lightbox

5.3.1 Definition

Hier soll eine sogenannte Lightbox erstellt werden. Bei Klick auf ein Vorschaubild oder eine Serie von diesen erscheint eine Art Pop-Up mit einer Großansicht des angeklickten Bildes, während der Hintergrund abgedunkelt wird. Dieses Bild, zusammen mit dem es haltenden [HTML](#)-Konstrukt, wird per [DOM](#)-Manipulation in das bestehende Seitenkonstrukt eingefügt und per [CSS](#) visuell formatiert. Die Positionierung kann horizontal mittig oder in einem definierten Abstand zur Mausposition festgelegt werden.

Innerhalb des eingespeisten Konstrukts kann der Nutzer die Bilder einer Serie durchblättern und wird über die Position in der Serie informiert.

Das Widget soll je nach übergebenem Parameter entscheiden, ob Einzelbilder oder eine Serie von Bildern angesteuert werden soll. Eine Bilderserie soll mit einem einfachen Textlink aufrufbar sein.

Per Parameter sollen die Texte (Überschrift, Button-Beschriftungen, „Bild x von x“) definiert werden. Aus welchem Attribut der Beschreibungstext und der Alternativtext für das Bild ausgelesen werden, lässt sich ebenfalls konfigurieren.

Alle Elemente sollen bei Erhöhung der Schriftgröße skalieren¹.

Der Aufbau soll möglichst modular und einfach erweiterbar sein, um eine größtmögliche Wiederverwendbarkeit zu gewährleisten.

Außerdem sollen öffentliche Funktionen implementiert werden, mit denen es möglich ist, ein bestimmtes Bild, das nächste Bild oder das vorherige Bild aufzurufen.

Ein optisches Abdunkeln (Abdimmen, Abschaffen) des eigentlichen Internetseiteninhalts wird optional eingebaut. Hierbei soll die Farbe, Transparenz und die Geschwindigkeit konfigurierbar sein.

5.3.2 Umsetzung

Die Vorschaubilder werden direkt mit den Großansicht-Bildern verlinkt, damit sie auch ohne Java-Script anwählbar sind.

Eine Serie von Bildern wird als [HTML](#) Liste angelegt².

1 vgl. WCAG 2.0 - 1.4.4 Resize Text [WAI \[2009i\]](#)

2 vgl. Techniques for WCAG 2.0 - H48 [WAI \[2009c\]](#)

Damit sich die Lightbox korrekt über alle anderen Seitenelemente legen kann, muss sie als letztes Element des [DOM](#) injiziert werden. Dies ist nicht konform mit Regel *SCR26: Inserting dynamic content into the Document Object Model immediately following its trigger element* der [WCAG 2.0](#)¹.

Alle Grafiken werden mit entsprechenden `alt` und `title` Tags ausgestattet². Damit die Grafiken und das Lightbox Konstrukt skalieren können, müssen alle Werte relativ gesetzt werden³.

Alle relevanten Elemente sollen in die Tabreihenfolge aufgenommen werden. Diese sollen sowohl im [DOM](#) als auch visuell passend angeordnet sein⁴. Hier wird das Elternelement fokussiert, damit auch die Überschrift vom Screenreader erkannt werden kann. Danach soll der *Schließen*-Link, dann der *nächstes-Bild*-Button und abschließend der *vorheriges-Bild*-Button angesteuert werden. Diese Steuerelemente sind per Standard bereits in den Tabindex aufgenommen. Wenn bei einer Bilderserie der Anfang bzw. das Ende erreicht werden, soll der entsprechende Button deaktiviert werden und der Fokus auf den anderen Button gesetzt werden.

Um eine gute Nutzbarkeit zu erreichen, soll ein Element, wenn es Fokus erhält, deutlich sichtbar sein und der Standard-Fokus, für gewöhnlich eine dünne gepunktete Linie, soll ersetzt werden⁵.

Innerhalb der Applikation soll mit den Pfeiltasten navigiert und mit der *ESC*-Taste die Lightbox geschlossen werden können⁶. Zusätzlich kann mit der *Pos1*- bzw. *Home*-Taste das erste Bild einer Serie und mit der *Ende*-Taste das letzte Bild aufgerufen werden.

Das Element, das für das Abdimmnen zuständig ist, muss bei einer Änderung des sichtbaren Bereichs (der Webseite) an die neue Größe angepasst werden.

ARIA

Damit [AT](#) Manipulationen am [DOM](#) erkennen können, sollen [ARIA Live Regions](#) eingesetzt werden. Nötig ist dies bei der [HTML](#)-Injektion in den [DOM](#), beim Ändern des angezeigten Großansicht-Bildes und der Änderungen am Beschreibungstext. Hier soll auch das Attribut `aria-busy` eingesetzt werden, da bei langsamem Verbindungen bzw. sehr großen Bildern eine längere Änderungsphase zu erwarten ist.

1 vgl. Techniques for WCAG 2.0 - SCR26 [WAI \[2009c\]](#)

2 vgl. Techniques for WCAG 2.0 - G94, G95, G92, H37 [WAI \[2009c\]](#)

3 vgl. Techniques for WCAG 2.0 - C14, SCR34, G146, G179, C28, G142 [WAI \[2009c\]](#)

4 vgl. Techniques for WCAG 2.0 - G95, C27 [WAI \[2009c\]](#)

5 vgl. Techniques for WCAG 2.0 - C15, G149, G195, G165, SCR31 [WAI \[2009c\]](#)

6 vgl. Techniques for WCAG 2.0 - G90, G21 [WAI \[2009c\]](#) und WAI-ARIA Best Practices - 3.2.5 [WAI \[2009f\]](#)

Nach dem Einfügen des [HTML](#)-Konstrukts müssen die Elemente fokussierbar gemacht werden und das erste Element fokussiert werden. Nach Beenden der Lightbox soll der Fokus zurück auf das zuletzt ausgewählte Element gesetzt werden.

Das eingespeiste [HTML](#)-Konstrukt wird mit den Attributen `aria-role="alertdialog"` und `aria-live="assertive"` gekennzeichnet und mit `aria-labelledby` von einem zu definierenden Text betitelt. Das Element, mit dem man die Lightbox schließen kann, soll mit dem Attribut `aria-role="button"` und den entsprechenden `title` Tags¹ ausgestattet werden. Dieses Element wird als Link definiert².

Der Beschreibungstext soll mit `aria-labelledby` dem Bild zugeordnet werden.

Für die Positionsanzeige (innerhalb einer Bilderserie) könnten evtl. die [ARIA](#) Attribute `aria-valuemin`, `aria-valuemax`, `aria-valuenow` genutzt werden, da diese nicht eindeutig definiert sind. Laut [WCAG](#) 2.0 ist jeder Wert erlaubt³, was für diesen Zweck nützlich wäre. Die [ARIA](#) Spezifikation besagt aber, dass nur ein Dezimalwert angegeben werden darf⁴. Eine entsprechende Anfrage an die [WAI](#) blieb bisher noch ohne Antwort⁵.

5.3.3 Dokumentation

Der gesamte Quellcode ist im Anhang einzusehen:

[B.2.1 HTML](#)

[B.2.2 Java-Script](#)

HTML

Für jedes Element, das in eine Widget-Instanz aufgenommen werden soll, muss eine Klasse für das `a` Tag, welches das aufzurufende Bild umschließt, vergeben werden. Dieser Link muss auf die Großansicht des Bildes verweisen. Der *A/t*-Text und Beschreibung können in einem beliebigen Attribut definiert werden.

Wenn die Vorschaubilder mit der Textgröße skalieren sollen, müssen diese mit relativen Größenangaben versehen werden.

1 vgl. Techniques for WCAG 2.0 - G82; und in sinngemäßer Funktion auch H65 [WAI \[2009c\]](#)

2 vgl. Techniques for WCAG 2.0 - F42 [WAI \[2009c\]](#)

3 vgl. <http://www.w3.org/TR/WCAG20-TECHS/ARIA3.html>

4 <http://www.w3.org/WAI/PF/aria/#aria-valuemax>

5 siehe <http://lists.w3.org/Archives/Public/public-comments-wcag20/2009Jun/0001.html>

Um die eigentliche Lightbox darzustellen, wird nach dem Aufruf folgendes HTML-Konstrukt injiziert:

```

1 <div id="ui-lightbox-wrapper" class="ui-dialog ui-widget ui-widget-
   content ui-corner-all" tabindex="-1" role="dialog" aria-labelledby=
   "ui-dialog-title-dialog">
2   <div class="ui-dialog-titlebar ui-widget-header ui-corner-all ui-
      helper-clearfix">
3     <span class="ui-dialog-title" id="ui-dialog-title-dialog">
        Bildergallerie</span>
4     <a href="#nogo" id="ui-lightbox-close" class="ui-dialog-titlebar-
       close ui-corner-all" title="Schließen [ESC]" role="button">
5       <span class="ui-icon ui-icon-closethick">Schließen [ESC]</span>
6     </a>
7   </div>
8   <div id="ui-lightbox-content" aria-busy="false" aria-relevant="
      additions removals text" aria-live="assertive">
9     <div id="ui-lightbox-image"></div>
10    <p id="ui-lightbox-description">Bildbeschreibung</p>
11    <p id="ui-lightbox-pager">Bild 2 von 3</p>
12
13   <div id="ui-dialog-buttonpane" class="ui-dialog-buttonpane ui-
      widget-content ui-helper-clearfix">
14     <button id="ui-lightbox-next" type="button" class="ui-state-
       default ui-corner-all">nächstes Bild</button>
15     <button id="ui-lightbox-prev" type="button" class="ui-state-
       default ui-corner-all">vorheriges Bild</button>
16   </div>
17 </div>
18 </div>
19 <div id="ui-lightbox-screendimmer" ></div>
20 <form><input id="virtualBufferForm" value="1" type="hidden"></form>
```

Listing 5.11: ariaLightbox: injiziertes HTML-Konstrukt

Die Elemente mit **ID**-Attribut werden vom Widget direkt angesteuert, die **IDs** müssen also auf jeden Fall verbleiben. Es muss bei der Umarbeitung des Konstrukts bedacht werden, dass die Elemente mit der **ID** `ui-lightbox-wrapper`, `ui-lightbox-content` und `ui-lightbox-image` in ihrer Größe verändert werden. Ihre umschließende Funktion sollte sinngemäß beibehalten werden. Gleicher gilt für das die Buttons haltende Element mit der **ID** `ui-dialog-buttonpane`. Alle anderen Elemente können innerhalb ihrer Elternelemente frei bewegt und auch deren Typ verändert werden.

Das Element mit der **ID** `ui-lightbox-screendimmer` dient zur Abdunklung des restlichen Inhalts.

Das Formular mit der **ID** `virtualBufferForm` wird genutzt, um Screenreader auf Änderungen im Quellcode aufmerksam zu machen. Näheres hierzu in Punkt 3.1.1.

CSS

```

1 font-size: 0.6em;
2 }
3
4 /* Lightbox Styles
5 -----
6 #ui-lightbox-wrapper {
7   display: none;
8   position: absolute;
9   width: 20em;
10  height: auto;
11 }
12 #ui-lightbox-image {
13   height: 10em;
14   margin: 1em 0 0 0;
15   background: url.ajax-loader.gif) no-repeat center center;
16 }
```

Listing 5.12: ariaLightbox: CSS

Die Style-Angaben in diesem Listing sind Pflichtwerte. Das umschließende Element (`ui-lightbox-wrapper`) darf nur in seiner Breite geändert werden. Selbiges gilt für das Element mit der ID `ui-lightbox-image` in Bezug auf die Höhe. Das `margin` und das animierte Hintergrundbild dienen in diesem Fall nur zur Darstellung.

Für die verschiedenen Status der Elemente wurden die `ui-state` CSS-Klassen der *jQuery UI Themes* verwendet. Nähere Informationen hierzu unter Punkt [5.1.4](#).

Aufruf

Für jede Instanz des Widgets muss ein Selektor definiert werden. Dies wird im Normalfall eine CSS-Klasse sein, kann aber auch jeder andere jQuery Selektor sein.

```

1 <script type="text/javascript" src="js/ui.core.js"></script>
2 <script type="text/javascript" src="js/ui.ariaLightbox.js"></script>
3   >
4 <script type="text/javascript">
5   $(function() {
6     var lightbox1 = $(".singleLightbox").ariaLightbox({
7       altText: "alt",
8       descText: "title",
```

Listing 5.13: ariaLightbox: 63-69: Aufruf des Widgets

Der Aufruf des Widgets erfolgt ansonsten wie bei jQuery üblich.

Optionen

Um das Widget zu konfigurieren, können folgende Optionen gesetzt werden:

Name	Typ	Standard
imageArray	[] oder false	false
altText	String	alt
descText	String	title
prevText	String	vorheriges Bild
nextText	String	nächtes Bild
titleText	String	Lightbox
pictureText	String	Bild
ofText	String	von
closeText	String	Schließen [ESC]!
pos	String	auto
autoHeight	Number	50
offsetX	Number	10
offsetY	Number	10
useDimmer	Boolean	true
animationSpeed	String or Number	slow
zIndex	Number	1000
background	String	black
opacity	Decimal	0.8
em	Decimal	0.0568182
disabled	Boolean	false

Tabelle 5.6: ariaLightbox: Übersicht Optionen

Um eine Bildergalerie zu definieren, muss `imageArray: []` gesetzt werden. Mit den Optionen `altText` und `descText` kann das dynamisch auszulesende Attribut (des Bildes) ausgelesen werden, damit diese bei der Großansicht gesetzt werden. Mit Hilfe der Optionen `titleText`, `prevText`, `nextText`, `pictureText`, `ofText` und `closeText` können die statischen Texte für die Überschrift der Lightbox, die Buttons, die Texten, den Positionsanzeiger und den Schließen-Button geändert werden.

Per `pos` kann die Position auf ein bestimmten Wert (anzugeben in Pixeln in der Form `pos: "30,50"`), per `auto` für eine mittige Darstellung oder per `offset` für eine Darstellung relativ zur Mausposition gesetzt werden. In letzterem Fall kann mit `offsetX` und `offsetY` der Abstand zur Mausposition definiert werden.

Die Eigenschaften des per `useDimmer` aktivierbaren Dimmers können mit `background` und `opacity` eingestellt werden.

Der Multiplikator zur Berechnung des relativen Abstands kann mit der Option `em` eingestellt werden. Da das Lightbox [HTML](#)-Konstrukt immer direkt in den `body` Tag

eingespeist wird, muss dieser Wert für gewöhnlich nicht geändert werden. Der Wert berechnet sich aus der vererbten Schriftgröße aller Elternelemente. Mit Tools wie dem *Em-Calculator*¹ kann dieser Wert einfach bestimmt werden.

Events

Events oder auch Callbacks sind vordefinierte Aufrufe zu Funktionen, mit denen sich eigene Programmteile in den Ablauf des Widgets integrieren lassen. Folgende Events sind definiert:

Name	Erklärung
onShow	wird nach dem Injizieren des HTML-Konstrukts ausgelöst
onChangePicture	wird nach erfolgtem Bilder-Wechsel ausgelöst
onClose	wird beim Auslösen der <code>close</code> -Methode aufgerufen
onPrev	wird beim Auslösen der <code>prev</code> -Methode aufgerufen
onNext	wird beim Auslösen der <code>next</code> -Methode aufgerufen

Tabelle 5.7: ariaLightbox: Übersicht Events

Methoden

Um das Widget von außen anzusteuern können verschiedene Methoden genutzt werden, die in der nachfolgenden Tabelle aufgelistet sind.

Name	Aufruf	Erklärung
disable	<code>.ariaLightbox('disable')</code>	deaktiviert das Widget
destroy	<code>.ariaLightbox('destroy')</code>	entfernt die komplette Funktionalität und setzt das HTML zurück in den Ausgangszustand
enable	<code>.ariaLightbox('enable')</code>	aktiviert das Widget
startGallery	<code>.ariaLightbox('startGallery')</code>	startet die Galerie von außen
close	<code>.ariaLightbox('close')</code>	schließt die Lightbox
next	<code>.ariaLightbox('next')</code>	öffnet das nächste Bild in der Serie
prev	<code>.ariaLightbox('prev')</code>	öffnet das vorherige Bild in der Serie
option	<code>.ariaLightbox('option', optionName, [value])</code>	fungiert als Setter bzw. Getter, wenn kein Wert angegeben wird

Tabelle 5.8: ariaLightbox: Übersicht Methoden

1 von Piotr Petrus <http://riddle.pl/emcalc/>

Bei der Methode `startGallery` muss zusätzlich zum Methodennamen noch das `event` mit übergeben werden, damit das Widget die Mausposition feststellen kann.

```

1      });
2      $("#prevlink2").click(function (event) {
3          lightbox2.ariaLightbox('prev');

```

Listing 5.14: ariaLightbox: 102-104: startGallery Methode

Das Element mit der ID `#gallerylink2` steuert in diesem Fall einen einfachen Textlink an. Auf die Variable `lightbox2` wurde zuvor das initialisierte Widget gelegt.

Funktionsweise

Im nachfolgenden wird die Funktionsweise und der Ablauf des Widgets erläutert. Das Schaubild auf Seite 86 soll verdeutlichen, welche Funktionen beim ersten Aufruf der Lightbox von wo aus aufgerufen werden.

Nach dem Aufruf des Widgets werden in der `_init` Funktion alle selektierten Elemente im `imageArray` Array abgespeichert und es wird ein Klick-Event auf diese gelegt. Außerdem wird ein Event auf das Dokumentfenster gelegt, das Änderungen an dessen Größe feststellt und gegebenenfalls das Dimmer-Element anpasst.

Wenn ein Bild angeklickt wird, prüft das Widget in der `_open` Funktion, ob die Lightbox bereits angezeigt wird. Wenn dies der Fall ist, wird dessen Inhalt mit der Funktion `_changPicture` geändert. Andernfalls wird mit `_show` die Lightbox injiziert. In jedem Fall wird hier das zuletzt angeklickte Element gespeichert.

In der `_show` Funktion wird das zu injizierende `HTML`-Konstrukt zusammengebaut und versteckt injiziert. Damit die Reihenfolge im `DOM` eingehalten wird, wird das Element, das den restlichen Inhalt abdunkelt, vorher injiziert. Zu diesem Zeitpunkt wird

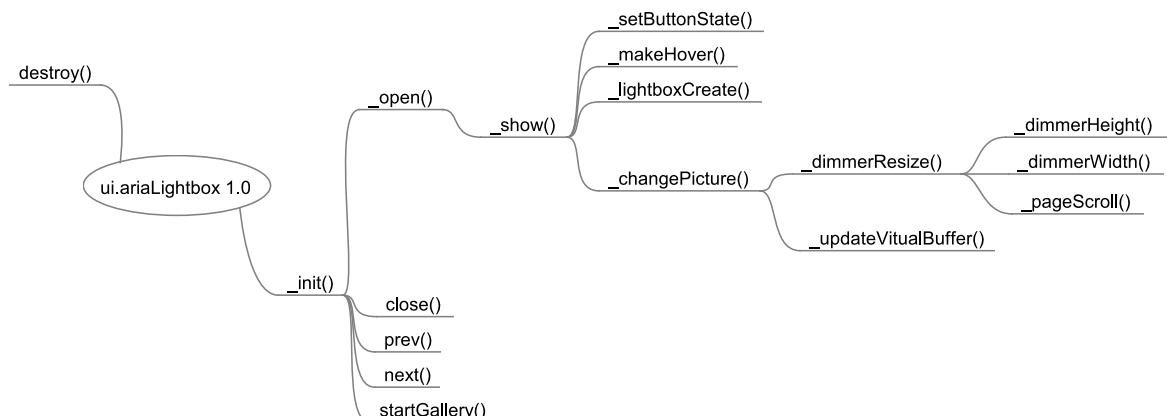


Abbildung 5.2: lightbox: Funktionsübersicht

das Callback `onShow` ausgelöst.

Danach wird die Tastaturbedienbarkeit erstellt und die Events werden auf die verschiedenen Schaltflächen gelegt. Visuelle Effekte wie `hover` und `focus` werden durch Aufruf der `_makeHover` Funktion ebenfalls an dieser Stelle gesetzt.

Nachdem die Position und Größe der Lightbox gesetzt wurden, wird diese eingeblendet. Als letztes wird die Funktion `_changePicture` aufgerufen um das eigentliche Bild und dessen Metadaten zu setzen.

Innerhalb der `_changePicture` Funktion blendet zuerst das derzeit angezeigte Bild aus und setzt die [ARIA](#)-Attribute. Daraufhin wird das Bild vorgeladen, um dessen Größe und damit auch Position der Lightbox zu errechnen.

Abschließend wird das Bild eingeblendet, die Texte werden geändert und die [ARIA](#)-Attribute neu gesetzt, um den Abschluß der Aktualisierung abzubilden. Da sich mit dem Öffnen der Lightbox auch die Höhe des Inhalts geändert haben könnte (das heißt, die Lightbox erzwingt eine horizontale oder vertikale Erweiterung der angezeigten Webseite), wird die `_resizeLightbox` Funktion aufgerufen.

Der Aufruf von `_updateVirtualBuffer` sorgt für eine erzwungene Aktualisierung des Screenreader-Buffers (siehe Punkt [5.2.3](#)).

Wenn eine Galerie geöffnet wurde, kann per Buttons oder per öffentlicher Methode die Serie durchgeklickt werden. Zu diesem Zweck werden die Funktionen `prev` bzw. `next` aufgerufen. Diese ändern jeweils die Variable `activeImage` und rufen dann die Funktion `_changePicture` und `_setButtonState` auf.

Die `_setButtonState` Funktion kontrolliert den Status der Buttons. Wenn beispielsweise das Ende einer Bilderserie erreicht ist, setzt sie den *nächstes-Bild*-Button auf deaktiviert und ändert dessen [CSS](#)-Klasse, um diese Funktionsänderung abzubilden.

Die `close` Funktion kann durch Klick auf den *Schließen*-Button, Klick auf den Dimmer, Drücken der *Escape*-Taste oder durch Ansteuern einer externen Funktion ausgelöst werden. Sie entfernt die Lightbox, gegebenenfalls den Dimmer und das Formular zur Aktualisierung des virtuellen Buffers und setzt schließlich den Fokus zurück auf das zuletzt angeklickte Vorschaubild bzw. den Link.

Anpassungen

Um etwaige Anpassungen der Darstellung oder Änderungen in der Funktion umzusetzen, werden im folgenden einige Codepassagen und ihre Funktion erläutert.

```
1 // build html
2 var html = "\n";
3 html += '<div id="ui-lightbox-wrapper" style="z-index:' + options.
    zIndex+1+ ';" class="ui-dialog ui-widget ui-widget-content ui-
    corner-all" tabindex="-1" role="dialog" aria-labelledby="ui-
```

```

        dialog-titleText-dialog">+'\n";
4   html += ' <div class="ui-dialog-titlebar ui-widget-header ui-
      corner-all ui-helper-clearfix">+'\n";
5   html += '   <span class="ui-dialog-title" id="ui-dialog-title-
      dialog">'+ options.titleText +''+'\n";
6   html += '   <a href="#nogo" id="ui-lightbox-close" class="ui-
      dialog-titlebar-close ui-corner-all" title="'+ options.
      closeText +' role="button">+'\n";
7   html += '     <span class="ui-icon ui-icon-closethick">'+ options
      .closeText +''+'\n";
8   html += '   </a>'+'\n";
9   html += ' </div>+'\n";
10  html += ' <div id="ui-lightbox-content">+'\n";
11  html += '   <div id="ui-lightbox-image"><img src="" aria-
      describedby="ui-lightbox-description" /></div>+'\n";
12  html += '   <p id="ui-lightbox-description"></p>+'\n";
13 // show pager and rage description if its an array of images
14 if (options.imageArray) {
15   html += '   <p id="ui-lightbox-pager"></p>+'\n";
16   html += '   <div id="ui-dialog-buttonpane" class="ui-dialog-
      buttonpane ui-widget-content ui-helper-clearfix">+'\n";
17   html += '     <button id="ui-lightbox-next" type="button" class="
      ui-state-default ui-corner-all">nächstes Bild</button>+'\n";
18   html += '     <button id="ui-lightbox-prev" type="button" class="
      ui-state-default ui-corner-all">vorheriges Bild</button>+'\n"
      ;
19   html += '   </div>+'\n";
20 }
21 html += ' </div> )+'\n";
22 html += '</div>+'\n";

```

Listing 5.15: ariaLightbox.js 118-139: Anpassungen HTML-Konstrukt

Im obigen Listing ist der Zusammenbau des injizierten [HTML](#) zu sehen. Wie bereits unter Punkt 5.3.3 angemerkt, dürfen bestimmte Verschachtelungen – und vor allem die [IDs](#) – nicht verändert werden.

```

1  // decide which position is set
2  switch (options.pos) {
3    case "auto":
4      var viewPos = self._pageScroll();
5      var posLeft = ((document.width() - options.
      wrapperElement.width())/2);
6      var posTop = viewPos[1]+options.autoHeight;
7      break;
8    case "offset":
9      var posLeft = event.pageX+options.offsetX;
10     var posTop = event.pageY-options.offsetY;
11     break;

```

```

12     default:
13         var position = options.pos.split(", ");
14         var posLeft = position[0];
15         var posTop = position[1];
16         break;
17     }
18     // set initial position, fade in and focus
19     options.wrapperElement
20         .css({
21             left: posLeft+"px",
22             top: posTop+"px"
23         })
24         .fadeIn(options.animationSpeed)
25         .focus();

```

Listing 5.16: ariaLightbox.js 191-215: Positionierung des HTML-Konstrukts

Das zu diesem Zeitpunkt bereits injizierte [HTML](#) wird auf seine im [CSS](#) festgelegte Größe abgefragt, um auf dessen Grundlage die Position zu errechnen. Im Fall von `auto` wird vertikal mittig ausgerichtet und horizontal die derzeitige Scroll-Position mit dem definierten Abstand (`options.autoHeight`) addiert.

Wenn die Option `offset` angegeben wurde, wird die übergebene Mausposition mit einem definierten Abstand addiert. Bei Angabe der genauen Position in Form eines Arrays wird diese Angabe direkt als Pixel-Entfernung vom linken, oberen Anzeigebereich der Webseite genutzt.

```

1     // preload image
2     var image = new Image();
3     image.onload = function() {
4         // set new picture properties
5         imageElement
6             .attr('src', element.attr("href"))
7             .attr('alt', element.find("img").attr(options.altText));
8
9         // if em isn't deactivated calculate relative size
10        var calculatedX = (options.em) ? image.width*options.em+"em"
11            : image.width;
11        var calculatedY = (options.em) ? image.height*options.em+"em"
12            : image.height;
12        // set image dimension | set always cause of display problems
13            with relative dimension setting
13        imageElement.css({
14            width: calculatedX,
15            height: calculatedY
16        });
17        // decide which position is set and animate the width of the
18            lightbox element
18        switch (options.pos) {

```

```

19   case "offset":
20     options.wrapperElement.animate({
21       left: event.pageX+options.offsetX+"px",
22       top: event.pageY+options.offsetY+"px",
23       width: calculatedX
24     }, options.animationSpeed);
25     break;
26   case "auto":
27     options.wrapperElement.animate({
28       left: ((document.width() - image.width)/2)+"px",
29       width: calculatedX
30     }, options.animationSpeed);
31     break;
32 }
33 // resize the height of the image wrapper to resize the
34 // lightbox element / wait till finished
35 imageWrapper.animate({
36   height: calculatedY
37 },
38 options.animationSpeed,
39 function () {
40   // fade in the picture
41   imageElement.fadeIn(options.animationSpeed);
42   // change description of the picture

```

Listing 5.17: ariaLightbox.js 237-277: Ändern des Bildes

In diesem Listing wurde kurz vorher das alte Bild ausgeblendet und nun wird das neue vorgeladen. Um die neuen Attribute `alt` und `src` zu setzen, werden aus dem angeklickten Hyperlink und dem darin befindlichen Bild die Attribute ausgelesen (Zeile 4-7). Ein anderer Aufruf des Widgets macht hier Änderungen nötig.

Im nächsten Codeblock wird, wenn nicht deaktiviert, die relative Größe der Lightbox errechnet, wobei die Größe des Bildes selbst nur aus *Performance*-Gründen gesetzt wird. Um eine korrekte Darstellung in allen Browsern zu erreichen, wird dann ab der *switch* Anweisung die Breite des Element `contentWrapper` (das alle Inhaltselemente mit variablem Inhalt umschließt) gesetzt, die Höhe aber nur dem Element `imageWrapper` (das nur das Bild umschließt) zugewiesen. Dies ist nötig, da nur vom Bild selbst eine exakte Größe bekannt ist, nicht aber vom Beschreibungstext, den Buttons und der Positionsanzeige.

Nachdem die Höhe des das Bild haltenden Elements animiert angepasst wurde, wird das Bild wieder eingeblendet.

```

1 // set button attributes
2 _setButtonState: function () {
3   var options = this.options;
4   // activate both buttons

```

```

5     options.buttonpane.find("#ui-lightbox-next, #ui-lightbox-prev")
6         .removeAttr("disabled")
7         .removeClass("ui-state-disabled")
8         .removeClass("ui-state-focus");
9     switch (options.activelimage) {
10        // disable prev
11        case 0:
12            options.buttonpane.find("#ui-lightbox-prev")
13                .attr("disabled", "disabled")
14                .removeClass("ui-state-hover")
15                .addClass("ui-state-disabled");
16            options.buttonpane.find("#ui-lightbox-next").focus();
17            break;
18        // disable next
19        case options.imageArray.length -1:
20            options.buttonpane.find("#ui-lightbox-next")
21                .attr("disabled", "disabled")
22                .removeClass("ui-state-hover")
23                .addClass("ui-state-disabled");
24            options.buttonpane.find("#ui-lightbox-prev").focus();
25            break;
26        }
27    },

```

Listing 5.18: ariaLightbox.js 303-329: Status der Buttons

Im obigen Listing sind die Änderungen an den Buttons zu sehen. Je nach gewünschter Darstellung sollen vielleicht keine Buttons, sondern normale *divs* verwendet werden. Hier wären die Statusänderung anzupassen. Zum Beispiel könnte man hier auch den jeweiligen *div* verstecken und wieder einblenden.

5.4 Tabs

5.4.1 Definition

Da in den jQuery UI Komponenten bereits ein Tabs-Widget (auch Reiternavigation genannt) enthalten ist¹, soll hier versucht werden, dieses barrierefrei zu erweitern und unter anderem mit [WAI ARIA](#) Eigenschaften auszustatten.

Das *Tabs* Widget beinhaltet bereits eine breite Funktionspalette, deren für die Erweiterung relevante Teile hier aufgeführt werden sollen.

Einerseits ist es bei den Tabs möglich, den Inhalt fest im Quellcode auszugeben und erst bei der Initialisierung die Tab-Funktionalität hinzuzufügen. Andererseits ist es auch möglich, nur die Navigationspunkte auszugeben und den eigentlichen Inhalt per

1 <http://jqueryui.com/demos/tabs/>

AJAX-Anfrage nachzuladen. Ein Nachladen kann auch von außen erzwungen werden. Außerdem können Tabs im Nachhinein hinzugefügt oder entfernt werden.

Es ist außerdem möglich, Tabs *zuklappbar* zu machen und das Event, auf das sie reagieren, zu bestimmten, oder einen bestimmten Tab im Nachhinein per öffentlicher Methode oder bei der Initialisierung zu öffnen.

Man kann die Tabs automatisch in zeitlicher Abfolge aufrufen lassen.

Wie bei allen jQuery UI Widgets ist es ebenfalls möglich, das gesamte Widget zu deaktivieren oder komplett zum Ausgangszustand zurückzukehren. Eine Besonderheit stellt hier die Möglichkeit dar, nur bestimmte Tabs zu deaktivieren bzw. aktivieren.

5.4.2 Umsetzung

Grundsätzlich wird hier versucht, die bereits vorhandene Funktionalität durch Erweiterung des bestehenden Widgets zu erweitern bzw. dessen Funktionen barrierefrei zu machen. Das bedeutet, dass keine grundlegenden Funktionen umgeschrieben und keine Anpassungen am HTML-Konstrukt vorgenommen werden sollen. Stattdessen soll bei vertretbarer Menge an neuem Code und ohne Redundanzen versucht werden, die Regeln der WCAG und ARIA zu implementieren.

Dabei soll eine Portierbarkeit zu zukünftigen Versionen der UI Tabs zu gewährleistet werden. Zu diesem Zweck werden die Funktionen abgefangen, das Original-Event angesteuert und dann zusätzliche Funktionalitäten hinzugefügt.

Viele der folgenden Anforderungen sind bereits im Widget umgesetzt.

Die Tabs werden als Liste angelegt¹, die Links enthalten, die entweder Sprungmarken zu den Inhalten oder aber direkte Links auf den per AJAX nachzuladenden Inhalt enthalten². Somit sind die Inhalte auch ohne Java-Script erreichbar. Der Inhalt wird semantisch korrekt angeordnet bzw. injiziert und per CSS zur korrekten Darstellung gebracht.³.

Alle Elemente sollen mit der Textgröße vergrößerbar sein⁴.

Aufgrund des Aufbaus als Linkliste ist das Widget bereits tastaturbedienbar⁵, soll aber zusätzlich mit weiteren Tastaturbefehlen bedienbar sein. Es soll möglich, sein mit den Pfeiltasten durch die Tabs zu blättern und mit der *Pos1*-Taste bzw. der *Ende*-Taste an den Anfang bzw. das Ende der Tabs zu springen⁶. Von einer Umsetzung der von der WAI angedachten Tastenkombination *Steuerung* (STRG-Taste) und *Tabulator* wird hier abgesehen, da diese Kombination in einigen Browsern auch zum Durchschalten der Browser-Tabs genutzt wird.

1 vgl. Techniques for WCAG 2.0 - H48 [WAI \[2009c\]](#)

2 vgl. Techniques for WCAG 2.0 - G117, SCR21 [WAI \[2009c\]](#)

3 vgl. Techniques for WCAG 2.0 - C6, C27, G57, G59, G140 [WAI \[2009c\]](#)

4 vgl. Techniques for WCAG 2.0 - C14, SCR34, G146, G179, C28, G142 [WAI \[2009c\]](#)

5 vgl. Techniques for WCAG 2.0 - H91, SCR35, SCR2 [WAI \[2009c\]](#)

6 vgl. WAI ARAI - 9. Design Patterns - Tabpanel [WAI \[2009f\]](#)

Eine visuelle Markierung beim Überfahren, beim Aktivieren und für den aktiven Tab ist bereits integriert¹.

Die `rotate` Funktion, die ein zeitlich bestimmtes Aktivieren der einzelnen Tabs ermöglicht, kann – je nach Einsatzzweck und den dem Nutzer zugestandenen Kontrollfunktionen – gegen die Richtlinie 2.2 *Enough Time* der WCAG verstößen². Hier ist der jeweilige Konzepter bzw. Entwickler gefragt, um entsprechende Hilfestellungen anzubieten. Zum Beispiel sollte eine Steuerung angeboten werden um die Rotation zu stoppen. Diese Funktionalität wird vom Original Widget angeboten.

ARIA

Um die versteckten Inhaltselemente korrekt als versteckt abzubilden, sollen die ARIA Attribute `aria-hidden` sowie `aria-expanded` eingesetzt werden³.

Der zum angezeigten Inhaltsbereich gehörende Listenpunkt soll mit dem Attribut `aria-selected="true"` gekennzeichnet werden⁴.

Mit Hilfe des `aria-labelledby` Attributes soll das Inhaltselement mit dem dazu gehörigen Tab verknüpft werden⁵. Zu diesem Zweck wird die ID des Tabs in das Attribut des Inhaltselements eingetragen. Auch das Attribut `aria-controls` soll eingesetzt werden, um der AT zu verdeutlichen, dass Interaktion mit den Tabs den Inhalt ändert.

Um das Springen per Tastatur von den Tabs zu den eigentlichen Inhalten zu vereinfachen, soll ein `tabindex` vergeben werden. Mit dem Vergeben eines Wertes von -1 soll verhindert werden, dass der Nutzer nach dem Anwählen eines Tabs durch alle weiteren navigieren muss, bevor er beim eigentlichen Inhalt ankommt. Der angewählte Tab erhält den Wert 0⁶.

Um die Rollen der einzelnen Elemente zu kennzeichnen, wird das Elternelement mit `application`, die Liste mit `tablist`, die einzelnen Tabs mit `tab` und die Inhaltselemente mit `tabpanel` ausgestattet.⁷ Da die Listenelemente selbst keine Funktion erfüllen, sollten diese mit `role="presentation"` markiert werden⁸.

1 vgl. Techniques for WCAG 2.0 - C15, G149, G195, G165, SCR31 WAI [2009c]

2 vgl. mit Punkt 46

3 vgl. WAI ARIA Best Practices - 2. General Steps for Building an Accessible Widget with ARIA - 8. Showing and Hiding Sections in a Widget, 9. Design Patterns - Tabpanel WAI [2009f]

4 vgl. WAI ARIA Best Practices - 9. Design Patterns - Tabpanel WAI [2009f]

5 vgl. Succes Criterion 1.3.1 - Info and Relationship (siehe Punkt 4.4.3)

6 vgl. Yahoo! User Interface Blog - Step 2: Adding Enhanced Keyboard Support Kloots [2009]

7 vgl. Techniques for WCAG 2.0 - G10, G135, G108 WAI [2009c], WAI ARIA - 4.4 Definition of Rules - tab,tabpanel,tablist WAI [2009a]

8 vgl. Yahoo! User Interface Blog - Screen-Reader Specific Tweaks Kloots [2009]

Mit Hilfe von `aria-live` sollen Veränderungen per [AJAX](#) für die [AT](#) sichtbar gemacht werden. Wenn möglich, soll neben `aria-live="polite"` auch `aria-busy` eingesetzt werden, um die Dauer einer Aktualisierung deutlich zu machen. Durch die Kennzeichnung mit `aria-relevant` und dessen Werten `removals` und `additions` soll das Entfernen und Hinzufügen von Tabs und deren Inhaltselementen gekennzeichnet werden.

5.4.3 Dokumentation

Der gesamte Quellcode ist im Anhang einzusehen:

B.3.1 Java-Script

Die Java-Script-Datei wird einfach unter dem Original-Widget eingebunden.

```

1 <script type="text/javascript" src="js/jquery.js"></script>
2 <script type="text/javascript" src="js/ui.core.js"></script>
3 <script type="text/javascript" src="js/ui.tabs.js"></script>
4 <script type="text/javascript" src="js/ui.ariaTabs_min.js"></script>
      >
5
6 <script type="text/javascript">
7   $(function() {
8     var tabs1 = $("#tabs")
9     .tabs();
10
11    $("#destroylink").click(function (event) {
12      tabs1.tabs('destroy');
13    });

```

Listing 5.19: ariaTabs: 11-23: Aufruf

Mehr Einstellungen müssen nicht vorgenommen werden. Es können aber ganz normal Einstellungen für das jQuery UI Tabs Widget gemacht werden.

Angemerkt sei hier, dass die in den Beispielen des UI Tabs Widgets gezeigte `sortable` Funktion nicht komplett barrierefrei ist, da diese Funktion dem UI Sortable Widget zugrunde liegt. Diese wiederum ist nicht mit der [ARIA](#) konform.

Das folgende Listing zeigt die Erweiterung des UI Tabs Widgets, wie sie zum Beispiel auch beim setzen der Standard Optionen am Ende der meisten UI Widgets verwendet wird.

```

1 (function ($) {
2   $.fn.extend($.ui.tabs.prototype, {
3
4     original_init: $.ui.tabs.prototype._init,
5     // when widget is initiated
6     _init: function () {

```

```

7     var self = this, options = this.options;
8     // fire original function
9     self._original_init();

```

Listing 5.20: ariaTabs.js: 18-26: Aufruf

Zuerst wird die Original-Funktion (in diesem Fall die `_init` Funktion) zwischengespeichert, um sie dann zu überschreiben. Anschließend wird diese Kopie aufgerufen und löst damit die Original-Funktion aus. Vor oder nach dem Aufruf der Original-Funktion kann eigener Code stehen. Auf diese Weise wurden alle benötigten Methoden und Funktionen erweitert.

In der `_init` Funktion werden die verschiedenen Elemente mit ihren **ARIA** Attributen ausgestattet. Mit Hilfe der `_ariaInit` Funktion werden jedem Tab und Inhaltselement alle benötigten Attribute zugewiesen. Abschließend werden die Tastaturevents gesetzt, die jeweils die öffentliche Standard-Methode `select` nutzen. Die `_ariaInit` Funktion enthält noch eine Besonderheit:

```

1  if (this.options.collapsible) {
2      $(this.anchors[index]).bind(this.options.event, function(
3          event) {
4              // get class to negate it to set states correctly when
5              // panel is collapsed
6              self._ariaSet(index, !$(self.panels[index]).hasClass("ui-
7                  tabs-hide"));
8          });
9      }

```

Listing 5.21: ariaTabs.js: 127-132: Collapsible-Funktionalität abbilden

Um die Möglichkeit des *zuklappens* abzubilden, wird ein zusätzliches Event auf die Tabs gelegt. Dieses erkennt den derzeitigen Status des Inhaltselement und setzt dementsprechend die **ARIA** Attribute.

Die `_init` Funktion ruft standardmäßig die `load` Funktion auf, um den ersten Tab zu öffnen. Diese Funktion wird immer dann angesteuert, wenn ein Tab geöffnet wird, auch wenn zum Beispiel eine Rotation durchgeführt wird.

Die `load` Funktion wurde so erweitert, dass zuerst bei allen Tabs und deren Inhaltselementen die **ARIA** Attribute als deaktiviert gesetzt werden. Dies geschieht per Aufruf der `_ariaSet` Funktion und per Entfernung der evtl. verbleibenden *Live Region* Attribute. Anschließend werden die **ARIA** *Live Region* Attribute für den ausgewählten Tab gesetzt, der Inhalt per Aufruf der Original `_init` Funktion geladen bzw. angezeigt und die Eigenschaften der **ARIA** Attribute nochmals verändert, um der **AT** die Aktivierung des Tabs kenntlich zu machen.

Wenn ein Tab per `add` Funktion des Original-Widgets eingebracht wird, muss dieser entsprechend initialisiert werden:

```

1      _original_add: $.ui.tabs.prototype.add,
2      // called when a tab is added
3      add: function(url, label, index) {
4          // fire original function
5          this._original_add(url, label, index);
6          // ARIA
7          this.element
8              .attr("aria-live", "polite")
9              .attr("aria-relevant", "additions");
10
11         // if no index is defined tab should be added at the end of the
12         // tab list
13         if (index) {
14             this._ariaInit(index);
15             this._ariaSet(index, false);
16         } else {
17             this._ariaInit(this.anchors.length -1);
18             this._ariaSet(this.anchors.length -1, false);
19         }
20     },

```

Listing 5.22: ariaTabs.js: 135-153: add Funktionalität

Zuerst wird die Original-Funktion ausgelöst und der [AT](#) per *Live Regions* mitgeteilt, dass ein Tab hinzugefügt wurde. Anschließend wird der Tab und dessen Inhaltselement per `_ariaInit` initialisiert und per `_ariaSet` aktiviert.

Die `remove` Funktion wurde erweitert, damit der [AT](#) per *Live Regions* mitgeteilt wird, wenn ein Tab entfernt wurde.

Der Vollständigkeit halber wurde die `destroy` Funktion erweitert, um alle gesetzten Attribute aus den Elementen zu entfernen.

5.5 Sortierbare Tabellen

5.5.1 Definition

Um die Nutzbarkeit von tabellarischen Daten zu verbessern wird dem Anwender oft die Möglichkeit gegeben Spalten zu sortieren, zum Beispiel eine Liste mit Produkten nach Preis oder Beliebtheit. Diese Funktionalität kann auch clientseitig mit Java-Script erfolgen. Mit dem Widget soll in dieser Art die Darstellung und Nutzbarkeit von tabellarischen Daten barrierefrei verbessert werden.

Hierbei soll es möglich sein eine Tabelle lediglich um die Sortierfunktion zu erweitern, aber auch komplexe Änderungen an den dargestellten Daten vorzunehmen. Das beinhaltet zum einen nur eine begrenzte Anzahl von Reihen anzuzeigen und die Daten

„durchzublättern“. Diese sogenannte *Pager* Funktion wird als optionale Steuerung eingebaut, da sie in vielen Variationen umsetzbar ist.

Zum anderen sollen bestimmte Spalten und Reihen dynamisch abschaltbar sein, um sie nicht mehr anzuzeigen. Mit dieser Funktionalität wäre es zum Beispiel möglich dem Nutzer eine vielspaltige Tabelle anzubieten, bei der er selbst entscheiden kann welche Spalten aktuell dargestellt werden. Eine andere Möglichkeit stellt eine Suche innerhalb der Tabelle dar. Beide Beispiele werden rudimentär erstellt, dienen aber nur Demonstrations- und Testzwecken und sind somit nicht auf Barrierefreiheit optimiert.

Bestimmte Spalten sollen von der Sortierung ausgenommen werden können, um eine serverseitige, komplexe Sortierung anzustoßen. Innerhalb der Applikation soll eine Sortierung nach folgenden Kriterien möglich sein:

- Text (alphabetisch)
- natürliche Zahl
- Dezimalzahl (deutsch mit Komma)
- Dezimalzahl (ISO mit Punkt)
- Datum (deutsch DD.MM.YYYY)
- Datum (ISO YYYY-MM-DD)
- Datum (US MM/DD/YYYY)
- Zeit (HH:MM:SS)

Diese Arten der Sortierung sollen einfach erweiterbar sein.

5.5.2 Umsetzung

Laut [WCAG](#) 2.0 sind für die tabellarische Darstellung von Daten das [HTML](#) Element `table` und seine Kindelemente zu verwenden¹. Diese bestehen aus einem Tabellenkopf (`<thead>`) in dem sich die Kopfzellen befinden und einem Datenbereich (`<tbody>`). In beiden Bereichen befinden sich ein oder mehrere Zeilen (`tr`) und in diesen wiederum Zellen (`td`) bzw. Kopfzellen (`th`).

Um die Beziehungen zwischen Tabellenkopf (`th` Element) und Zellen (`td` Element) zu verdeutlichen, werden die `headers` Attribute der Zellen mit den `IDs` der Tabellenköpfe

¹ vgl. Techniques for WCAG 2.0 - H51 [WAI \[2009c\]](#)

verknüpft¹.

Außerdem soll mit dem scope Attribut und dessen Werten row, col festgelegt werden ob die Zelle als Kopf fungiert². Beispielsweise sollten bei einer einfachen Tabelle die th Elemente mit scope="col" ausgestattet werden um sie als Spaltenkopf zu deklarieren. Analog hierzu wird die entsprechende Zelle mit dem Attribut role und den Werten gridcell oder rowheader bzw. columnheader versehen³.

Entsprechend werden die Zeilen mit role="row" ausgestattet⁴.

Das Elternelement table wird durch das innerhalb positionierte caption Element mit einer Überschrift versehen⁵. Diese Überschrift wird zusätzlich mit der Tabelle verknüpft indem dieser das ARIA Attribut aria-labelledby mit der ID des caption Elements gegeben wird. Die Tabelle sollte außerdem mit dem Attribut summary, das einen Kurzüberblick enthält, versehen werden. Außerdem wird sie mit dem Attribut role="grid" als Tabelle und per aria-readonly="true" als nicht beschreibbar gekennzeichnet⁶.

Die Applikation soll tastaturbedienbar sein. Die Spalte soll mit den Pfeiltasten *Links* und *Rechts* sowie mit der *Tabulator*-Taste (ggf. in Verbindung mit der *Shift*-Taste) zu wechseln sein. Per Klick auf einen Spaltenkopf, durch Drücken der *Enter*- oder *Leertaste* kann die Sortierung der selektierten Spalte aktiviert werden⁷. Mit der *Hoch*- bzw. *Runtertaste* können die Datensätze der gesamten Tabelle durchblättert werden⁸. Auf eine Funktion zur Selektion von einzelnen Zellen wird zur Wahrung der universellen Einsetzbarkeit verzichtet. Aus diesem Grund wird auch von einer genauen Umsetzung, wie in der WAI ARIA Spezifikation vorgegeben, abgesehen und stattdessen die native Tabreihenfolge beibehalten, was den Vorteil hat, dass zum Beispiel Links innerhalb der Tabelle wie gewohnt per Tastatur bedienbar bleiben.

Um die Änderungen innerhalb der Kopfzellen und den eigentlichen Daten der Tabelle abzubilden, werden jeweils der tbody und der thead mit aria-live="polite" und aria-relevant="text" versehen.

Um Änderungen an der Sortierung deutlich zu machen werden den Kopfzellen title Attribute und CSS Klassen zugewiesen, die nach jeder Sortierung geändert werden, um den neuen Status abzubilden. Mit dem ARIA Attribut aria-sort soll dabei der

1 vgl. Techniques for WCAG 2.0 - H43 [WAI \[2009c\]](#)

2 vgl. Techniques for WCAG 2.0 - H63 [WAI \[2009c\]](#)

3 vgl. WAI ARIA Best Practice - 4.4. Definition of Roles - grid, row, gridcell [WAI \[2009f\]](#)

4 vgl. WAI ARIA Best Practice - 4.4. Definition of Roles - row [WAI \[2009f\]](#)

5 vgl. Techniques for WCAG 2.0 - H73, H39 [WAI \[2009c\]](#)

6 vgl. WAI ARIA Best Practice - 4.4. Definition of Roles - grid [WAI \[2009f\]](#)

7 vgl. Techniques for WCAG 2.0 - G90, G21 [WAI \[2009c\]](#) und WAI-ARIA Best Practices - 3.2.5 [WAI \[2009f\]](#)

8 vgl. WAI-ARIA Best Practices - 9. Design Patterns - Grid [WAI \[2009f\]](#)

derzeitige Stand der derzeit sortierten Spalte abgebildet werden¹.

Der sogenannte *Pager* wird mit Hilfe der [HTML](#) Elemente `button` erstellt und zur Verbesserung der Barrierefreiheit ebenfalls mit [ARIA](#) Attributen ausgestattet. Zum einen soll das Elternelement mit einem `aria-controls` mit der [ID](#) der Tabelle ausgestattet werden um deutlich zu machen, dass die Buttons die Tabelle steuern. Die Buttons werden per `aria-labelledby` mit dem Beschreibungstext (in diesem Fall z.B. *Seite:*) verbunden. Wenn einer der Buttons aktiviert ist – das heißt die entsprechenden Datensätze in der Tabelle angezeigt werden – wird dieser Button mit `aria-selected="true"` gekennzeichnet. Um zusätzliche Semantik in die Anwendung zu bringen, werden die Attribute `aria-valuenow`, `aria-valuemax` und `aria-valuemin` benutzt. Diese Funktion wird nur dem *Pager* Element und nicht der Tabelle selbst zugewiesen, da das Widget nicht darauf ausgelegt ist, strikt nach einem Seitenprinzip zu arbeiten und die genannten [ARIA](#) Attribute nicht für die Anzeige einer Spanne (nach dem Muster: *Datensatz 10-20 aus 50*) ausgelegt sind.

5.5.3 Dokumentation

Der gesamte Quellcode ist im Anhang einzusehen:

[B.4.1 HTML](#)

[B.4.2 Java-Script](#)

HTML und CSS

Die Applikation wird auf eine bestehende Tabelle und deren Daten angewendet. Das Widget benötigt also für seine Standardfunktionalität eine existierende [HTML](#) Tabelle mit `thead`, `tbody` und einem `caption` Element. Innerhalb des Tabellenkopfes muss ein `tr` mit mehreren `th` Elementen vorhanden sein. Im `tbody` Element müssen dann die eigentlichen Datensätze bestehend aus einer Zeile (`tr`) und ihren Zellen (`td`) definiert sein.

Die Anzeige der Sortierrichtung der einzelnen Spalten wird über die [CSS](#) Klasse des jeweiligen `th` Elements gesteuert. Für die verschiedenen Fokus-Zustände werden `ui-state` [CSS](#) Klassen der jQuery UI Themes verwendet. Nähere Informationen hierzu unter Punkt [5.1.4](#).

Aufruf

Der eigentliche Aufruf des Widgets wird wie üblich per Übergabe eines Selektors vorgenommen. Im Tabellenkopf sollten die `th` Elemente mit [CSS](#) Klassen ausgestattet

1 vgl. WAI-ARIA Best Practices - 5.6. Definitions of States and Properties [WAI](#) [2009f]

werden, die definieren nach welchen Kriterien sortiert werden soll. Folgende Klassen können vergeben werden:

Klasse	Sortierung	Anmerkung
ui-table-number	Natürliche und Dezimalzahlen (123 oder 123.456)	-
ui-table-number-de	Deutsche Dezimalzahlen (123,456)	-
ui-table-date	US Datum (07/28/2009)	-
ui-table-date-de	Deutsches Datum (28.07.2009)	-
ui-table-date-iso	ISO Datum (2009-07-28)	-
-	Alphabetisch, Uhrzeit (20:00:13)	Standard Sortierung
Besondere Klassen		
ui-table-deactivate	keine	Deaktiviert die betreffende Spalte.
ui-state-active	keine	Kennzeichnet eine vom bereits vom Server sortierte Spalte.

Tabelle 5.9: ariaSorTable: Übersicht Sortiermöglichkeiten

Standardmäßig wird eine alphabetische Sortierung vorgenommen. Durch die Klasse ui-state-active kann eine Sortierung, die von Anfang an zum Beispiel durch sortierte Ausgabe der Daten vom Server aktiv ist, gekennzeichnet werden. Um diese für den Nutzer abzubilden, sollte mit ui-table-asc bzw. ui-table-desc die derzeitige Sortierungsrichtung markiert werden.

Im folgenden Listing ist beispielhaft der Quellcode eines Tabellenkopfes dargestellt.

```

1   <thead>
2     <tr>
3       <th class="ui-table-asc ui-state-active ui-table-number"
4         style="min-width: 4.2em;"><a href="#server_site_order">
5           UID</a></th>
6       <th class="ui-table-number" style="min-width: 10em;"><a
7         href="#server_site_order">Number</a></th>
8       <th class="ui-table-number-de" style="min-width: 10em;">
9         Decimal DE</th>
10      <th class="ui-table-number" style="min-width: 6em;"><a href
11        ="#server_site_order">Decimal</a></th>
12      <th class="ui-table-date-de" style="min-width: 8em;"><a
13        href="#server_site_order">Date DE</a></th>
14      <th class="ui-table-date-iso ui-table-deactivate" style="
15        min-width: 8em;"><a href="#server_site_order">Date ISO</
16        a></th>
17      <th class="ui-table-text" style="min-width: 10em;"><a href=
```

```

10      "#server_site_order">String</a></th>
11      <th class="ui-table-deactivate" style="min-width: 12em;"><a
12          href="#server_site_order_only">false</a></th>
</tr>
</thead>

```

Listing 5.23: ariaSorTable: Tabellenkopf

Durch das Aktivieren des Widgets mit einer Dummy-Tabelle, die lediglich die Kopfzellen enthält und dem nachträglichen Hinzufügen der Daten durch Änderung des `originalData` Arrays, kann eine externe Datenzufuhr zum Beispiel mit [JSON¹](#) erreicht werden.

Optionen

Um das Widget zu konfigurieren, können folgende Optionen gesetzt werden:

Name	Typ	Standard
rowToStart	Number	1
rowsToShow	Number	false
colScopeRow	Number	1
defaultSortBy	String	asc
colsToHide	Array	false
rowsToHide	Array	false
keyboard	Boolean	true
pager	Boolean	false
textPager	String	Page:
textAsc	String	Sort ascending
textDesc	String	Sort descending
disabled	Boolean	false

Tabelle 5.10: ariaSorTable: Übersicht Optionen

Mit den Optionen `rowToStart` wird der Startpunkt (Index beginnt mit eins) bestimmt und mit `rowsToShow` die Anzahl der angezeigten Reihen. Mittels `colScopeRow` kann eine Zelle (bzw. Spalte) als Kopf ihrer Zeile definiert werden. Im nachfolgenden Beispiel soll diese Option erläutert werden.

¹ JavaScript Object Notation

Name	Haarfarbe	Größe
John Doe	braun	1,85
Jane Doe	schwarz	1,75

Tabelle 5.11: ariaSorTable: Beispiel für Option colScopeRow

In diesem Fall wäre es sinnvoll den Namen der Person als colScopeRow zu definieren um zu verdeutlichen, dass die Haarfarbe und die Größe zur Person gehören. Es sollte also eine Eins als Wert konfiguriert werden.

Die Option defaultSortBy legt die Sortierrichtung der Spalte beim ersten Aufruf fest. Mit den Optionen colsToHide und rowsToHide können Spalten bzw. Reihen ausgeblendet werden. Sie sind als Array zu konfigurieren, wobei der Index jeweils von Null an gezählt wird und als Wert true gesetzt werden muss.

Im folgenden Listing ist der exemplarische Aufruf mit zwei versteckten Spalten dargestellt.

```

1  var table1 = $("table").ariaSorTable({
2      rowsToShow: 10,
3      pager: true,
4      onInit: function() {
5          demoControl();
6      },
7      colsToHide: [
8          1: true,
9          7: true
10     ]
11 });

```

Listing 5.24: ariaSorTable: Aufruf

Mit keyboard und pager lässt sich die Tastaturbedienbarkeit und die Anzeige des Pager aktivieren bzw. deaktivieren.

Mit den Variablen textPager, textDesc und textAsc lassen sich die auszugebenden Texte konfigurieren.

Events

Events oder auch Callbacks sind vordefinierte Aufrufe zu Funktionen, mit denen sich eigene Programmteile in den Ablauf des Widgets integrieren lassen. Folgende Events sind definiert:

Name	Erklärung
onInit	Wird ausgelöst wenn Initialisierung abgeschlossen
onUpdateData	Wird nach Aktualisieren des Daten Arrays aufgerufen
onSetHTML	Wird ausgelöst nachdem das HTML ersetzt wurde
onRowSort	Wird ausgelöst nachdem die Daten neu sortiert wurden, aber bevor diese Änderung durch Aufruf von setHTML sichtbar wird.

Tabelle 5.12: ariaSorTable: Übersicht Events

Methoden

Um das Widget von außen anzusteuern, können verschiedene Methoden genutzt werden.

Name	Aufruf	Erklärung
disable	.ariaSorTable('disable')	Deaktiviert das Widget.
destroy	.ariaSorTable('destroy')	Entfernt komplett die Funktionalität und setzt das HTML zurück in den Ausgangszustand.
enable	.ariaSorTable('enable')	Aktiviert das Widget.
updateData	.ariaSorTable('updateData')	Aktualisiert die Daten im Array durch Abgleich mit colsToHide und rowsToHide.
setHTML	.ariaSorTable('setHTML')	Aktualisiert die in der Tabelle dargestellten Daten.
rowSort	.ariaSorTable('rowSort')	Sortiert das Daten-Array und ruft dann setHTML auf.
colSwitch	.ariaSorTable('colSwitch', value)	Verschiebt die selektierte Spalte um den übergebenen Wert.
buildPager	.ariaSorTable('buildPager')	Injiziert den Pager anhand der aktuellen Daten im Array
setPager	.ariaSorTable('setPager', value)	Setzt, bei aktiviertem Pager, die aktive Seite auf den übergebenen Wert.
option	.ariaSorTable('option', optionName, [value])	Fungiert als Setter bzw. Getter wenn kein Wert angegeben wird.

Tabelle 5.13: ariaSorTable: Übersicht Methoden

Funktionsweise

Im nachfolgenden wird die Funktionsweise und der Ablauf des Widgets erläutert. Das Schaubild auf Seite 105 soll verdeutlichen, welche Funktionen der Applikation von wo aus aufgerufen werden.

Nach dem Aufruf des Widgets wird durch die `_init` Funktion zunächst ein eindeutiger Identifikator für die Tabelle gesetzt. Dieser kann die `ID` der Tabelle sein oder wird, falls keine vorhanden ist, aus einem Zeitstempel gewonnen. Dieser Identifikator wird dann als `ID` gesetzt und in der Option `uid` abgespeichert, da eine solche nötig ist, um einige `ARIA` Attribute und später auch `HTML` Tabellen Relationen setzen zu können.

Im Anschluß wird zunächst die Zeile der Tabellenköpfe mit einem Event ausgestattet, um bei Klick auf einen der Köpfe eine Sortierung durch die `rowSort` Funktion auszulösen. Dann werden alle Tabellenköpfe einzeln eingelesen und abgespeichert. Diese werden nun auf ihre `CSS` Klassen und einen vorhandenen Link getestet. Wenn kein Link vorhanden ist, eine Sortierung aber möglich sein soll, wird ein Link hinzugefügt. Andernfalls wird die Kopfzelle selbst mit einem `tabindex` versehen, um alle Kopfzellen per Tabulator-Taste durchgehen zu können. Allen Links wird außerdem ein `Hover`-Event zugewiesen.

Falls eine Spalte mit der `CSS` Klasse `ui-table-active` bereits als sortiert definiert wurde, wird diese aktive Spalte in der Option `activeCol` gespeichert.

Im folgenden wird jede Zeile und deren Zellen der Original Tabelle einzeln aufgerufen, um die vorhandenen Daten im `originalData` Array abzuspeichern. Durch Aufruf der `updateData` Funktion werden diese Daten in das Arbeits-Array `tableData` kopiert. Dabei werden etwaige ausgeblendete Spalten und Zeilen nicht übernommen.

Abschließend für die Initialisierung werden die angezeigten Daten durch Aufruf der `setHTML` Funktion geändert und, falls aktiviert, die Tastatursteuerung und der `Pager` initialisiert.

Die `setHTML` Funktion baut, unter Beachtung verschiedener `ARIA` Attribute und `HTML` Relationen, das neue `HTML` aus dem Arbeits-Array `tableData` zusammen und injiziert es. Beim ersten Aufruf der Funktion wird der alte `tbody` versteckt und ein neuer hinzugefügt, da das Löschen von großen Tabellen *Performance*-Probleme verursacht. Anschließend werden nicht benötigte (im Array `colsToHide` als ausgeblendet definierte) `th` Elemente versteckt und benötigte wieder angezeigt. Dadurch ist es möglich verschiedene Attribute (wie die Größe, spezielle `CSS` Klassen, etc.) zu definieren und diese im Laufe der Applikationsnutzung beizubehalten. Außerdem wird dadurch die *Performance* – im Gegensatz zum Löschen und neu Hinzufügen der Elemente – gesteigert.

Da in dieser Funktion das `DOM` geändert wird, ruft `setHTML` abschließend auch die `updateVirtualBuffer` Funktion auf, um diese Änderung Screenreadern bekannt zu machen.

Die Tastatursteuerung prüft bei einer Interaktion ob die ausgewählte Aktion möglich ist. Falls ja, wird je nach Aktion die Option `rowToStart` neu gesetzt und mit

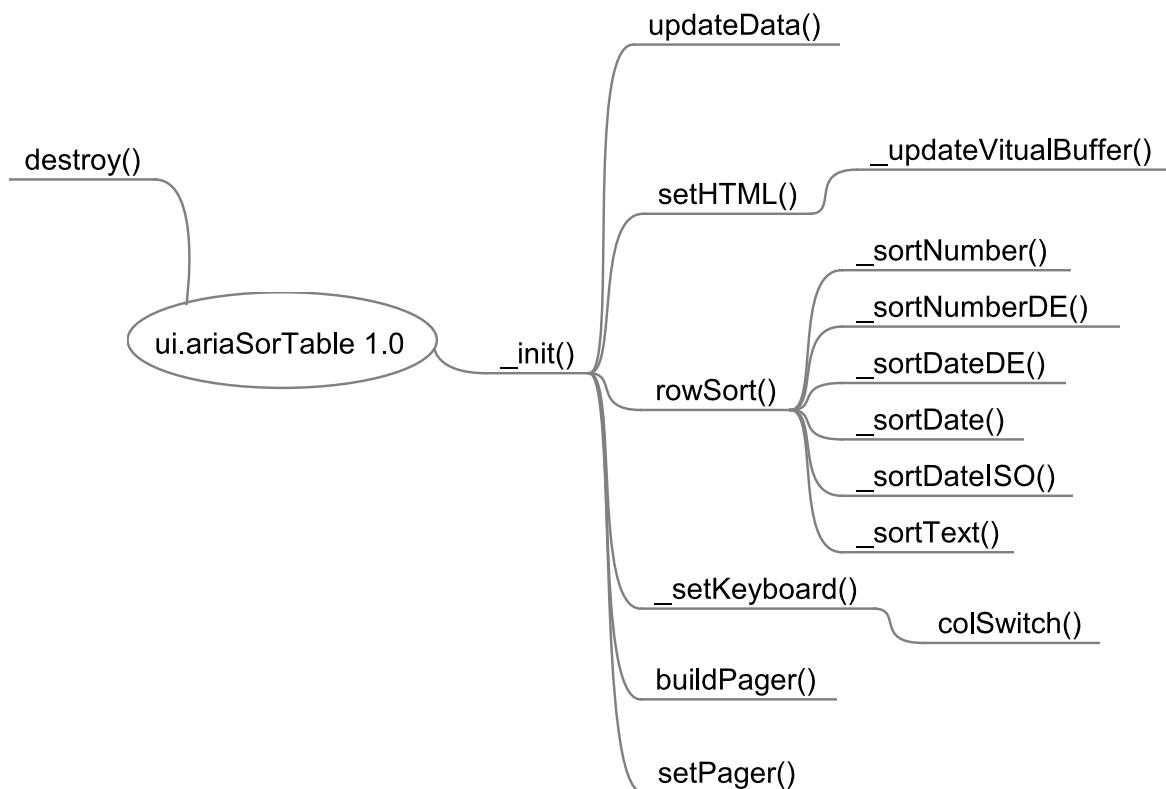


Abbildung 5.3: ariaSorTable: Funktionsübersicht

Aufruf der `setHTML` Funktion das injizieren des neuen [HTML](#) gestartet oder aber (bei einem Wechseln der Spalte) die `colSwitch` Funktion aufgerufen.

Innerhalb der `colSwitch` Funktion werden alte [CSS](#) Klassen gelöscht, neue hinzugefügt und die Option `selectedCol` mit dem der Funktion übergebenem, aktuellen Wert überschrieben. Abschließend wird der Fokus auf die Kopfzelle bzw. dessen Link gesetzt.

Der *Pager* baut zunächst das zu injizierende [HTML](#) bestehend aus Elternelement, Überschrift und `button` Elementen zusammen und versieht diese Elemente mit [ARIA](#) Attributen. Die Buttons werden nach dem Injizieren mit Events belegt. Bei Klick auf einen davon werden die Funktionen `setHTML` und `setPager` aufgerufen um die Änderung der Tabelle anzustoßen und abzubilden.

Die `setPager` Funktion dient der Änderung von [CSS](#) Klassen und von [ARIA](#) Attributen der Buttons, um die aktuell angezeigte Seite für den Nutzer sichtbar zu machen.

Wenn der Nutzer durch Klicken oder Drücken der Leer- bzw. *Entertaste* den Sor-

tiervorgang startet wird die `rowSort` Funktion ausgelöst. Diese erkennt zuerst an Hand der [CSS](#) Klasse die Sortierart und ruft die native Array Funktion `sort` mit entsprechender Vergleichsfunktion auf. Um das Widget um eine Sortierart zu erweitern, müsste hier eine weitere Bedingung und als gesonderte Funktion die dazugehörige Vergleichsfunktion hinzugefügt werden. Falls nötig wird nun noch die Reihenfolge des Arrays umgekehrt. In jedem Fall werden an den Kopfelementen und ihren Links Änderungen an Attributen und [CSS](#) Klassen vorgenommen.

Abschließend wird mit dem Aufruf der `setHTML` Funktion eine Aktualisierung der Tabelle angestoßen.

Anpassungen

Die Applikation wurde als eigenständig arbeitendes Widget ausgelegt, kann aber auch in bestehende Logik eingebaut oder erweitert werden. Wie eine solche Erweiterung aussehen könnte wurde zu Demonstrations- und Testzwecken in einer einfachen, weder besonders optimierten noch barrierefreien, Form in der Datei `demo.js` ausgearbeitet. Die Suche zum Ausblenden bestimmter Zeilen und das Formular mit Checkboxen zum Ein- und Ausblenden von Spalten kann in der Demo getestet werden.

Im nachfolgenden soll eine der Erweiterungen erläutert werden, um so die Funktionsweise des Widgets und seiner öffentlichen Methoden zu verdeutlichen und die Entwicklung eigener Erweiterungen zu vereinfachen. Außerdem werden zunächst intern verwendete Optionen aufgeführt, die zum Verständnis nötig sind.

Name	Typ	Erklärung
activeCol	Number	Enthält die z.Z. sortierte Spalte.
selectedCol	Number	Enthält die z.Z. selektierte Spalte.
originalData	Array	Enthält die Original Daten der Tabelle.
tableData	Array	Enthält die bereinigte Arbeitsversion der Daten der Tabelle.
headers	Array	Enthält alle th Elemente.
shift	Boolean	True wenn die Shift Taste gedrückt ist.
uid	String	Enthält eindeutigen Identifikator für die Instanz.
pager	Object	Enthält das Pager Elternelement.
pagerButtons	Array	Enthält alle Buttons des Pager.

Tabelle 5.14: ariaSorTable: Übersicht interne Optionen

Diese Variablen werden intern eingesetzt und sollten nur verändert werden, wenn eine Erweiterung des Widgets angestrebt wird.

Um die Funktionalität der Demo-Steuerung zu implementieren, muss zunächst das [HTML](#) für die Bedienoberflächen zusammengesetzt und injiziert werden. Diese bestehen für die Suche aus einem Textfeld mit Absendebutton und für die Spaltenssteuerung aus mehreren Checkboxen (für jede Spalte eine) und einem Absendebutton. Nachdem einige *Hover*-Effekt gesetzt wurden, muss noch die eigentliche Funktionalität eingebaut werden.

```
1  colSelect.submit( function ( event ) {  
2      event.preventDefault();  
3      var checkboxes = colSelect.find( "4      for ( var x = 0; x < checkboxes.length; x++ ) {  
5          var test = ( $(checkboxes[x]).filter( ':checked' ).length ) ? false  
6              : true;  
7          colsToHide[x] = test;  
8      }  
9      widget.ariaSorTable( 'updateData' );  
10     widget.ariaSorTable( 'setHTML' );  
11  });
```

Listing 5.25: ariaSorTable: Erweiterung

Im oben stehenden Listing wird das *Submit*-Event der Checkboxen bzw. ihres *Submit*-Buttons abgefangen und, um das Neuladen der Seite zu verhindern, das Standard-Event verhindert. Dann werden alle Checkboxen durchlaufen und ihr Status direkt in die Option `colsToHide` übertragen. Diese Status werden dann durch Aufruf der Funktion `updateData` auch im Arbeits-Array `tableData` abgebildet. Durch Aufruf der Funktion `setHTML` werden die vom Nutzer gewählten Spalten in der Tabelle angezeigt.

KAPITEL 6

Fazit

Die Zielgruppe für barrierefreie Internetseiten ist größer als weithin angenommen. Die Verschmelzung von Nutzbarkeit und Zugänglichkeit macht eine strikte Trennung schwierig und somit fallen nicht nur Menschen mit einer anerkannten Behinderungsform in den Kreis der Nutznießer von barrierefreien Internetseiten. Auch Anfänger mit dem Medium Internet, Menschen mit starker Sehschwäche, Menschen mit Legasthenie oder Konzentrationsschwäche, ältere Menschen und viele weitere profitieren direkt von Maßnahmen für mehr Barrierefreiheit. Noch mehr profitieren stärker eingeschränkte, zum Beispiel blinde Menschen von speziellen Anpassungen für die von Ihnen verwendete Hilfstechnologie. Gerade moderne Internetseiten, die Seiten der Generation Web 2.0 und danach, beherbergen durch Verwendung von Java-Script-Technologie wie [AJAX](#) und multimediale Inhalte oftmals Hürden, die ohne spezielle Anpassungen kaum zu überwinden sind.

Die Menge der Arten von technischen Hilfsmitteln für Menschen mit Behinderung ist überschaubar – die Menge der verschiedenen Produkte, ihrer Funktionen, Eigenheiten beim Bedienen und nicht zuletzt auch ihre Abhängigkeiten sind es weniger. Beispielsweise hängt die korrekte Funktionsfähigkeit und damit der Nutzen eines Screenreaders zusammen mit der [WAI ARIA](#) Spezifikation von drei Faktoren ab: dem Betriebssystem, dem Browser und dem Screenreader selbst. Oft sind bestimmte Funktionalitäten in einen Teil dieses Gefüges schlicht noch nicht implementiert oder bestimmte Kombinationen können nicht alle Zustände und Eigenschaften abbilden.

Der nicht genau informierte Internetentwickler kann sich also grundsätzlich nicht sicher sein, ob die von ihm implementierte Technologie für einen speziellen Nutzerkreis auch wirklich einen zusätzlichen Nutzen erwirkt. Das kaum verlässlichen Zahlen und Statistiken (wie zum Beispiel bei Browern und deren Funktionen) zur Verbreitung, Version der [AT](#) und des Betriebssystems sowie zu den Verwendungsgewohnheiten der Nutzer verfügbar sind, macht es zusätzlich schwierig, spezifische Anpassungen umzusetzen und verlässliche, aussagekräftige Testszenarien zu entwickeln.

Informationen zu Kompatibilitäten und zur Funktionsweise der sich schnell weiterentwickelnden Technologie finden sich (oftmals nur sehr fragmentiert) in Form von Funktionstests sowie passenden Übersichten und Empfehlungen, Musterimplementierungen und Anleitungen auf verschiedenen Internetseiten von Experten und Communities.

Neben dem zum genauen Verständnis hilfreichem Hintergrundwissen sind jedoch vor allem technische Umsetzungsempfehlungen für die tägliche Arbeit von Internetentwickler unerlässlich. Bei der Analyse der verschiedenen für den Internetbereich relevanten Richtlinien stellte sich heraus, dass die [WAI WCAG](#) die wichtigste Vorgabe darstellt. Ihre enorme Verbreitung und der Einfluß auf andere Richtlinien macht sie – zusammen mit der Ausführlichkeit und Aktualität der dazugehörigen Dokumente – quasi zum Standard, der durch ihre andauernde Weiterentwicklung gefestigt wird. Das zur [WCAG](#) gehörende *Techniques* Dokument bietet mit seinen vielen Praxisempfehlungen und Negativbeispielen eine direkte Hilfestellung für die tägliche Arbeit von Internetentwicklern. Zusammen mit der allgemein anerkannten und auch schon recht gut implementierten [WAI ARIA](#) Spezifikation stellt die [WCAG](#) 2.0 die passende Grundlage für die praktische Umsetzung dieser Arbeit dar.

Bei der Entwicklung der vier Widgets ergaben sich in Bezug auf die barrierefreie Umsetzung Probleme bei der Konzeption der Funktionsweise und der Beschaffenheit. Die eigentlichen Regeln der [WCAG](#) sind allgemein gültig formuliert und konnten auch durch Zuhilfenahme der Praxisbeispiele der [WCAG](#) *Techniques* nicht immer auf eine spezifische Problemstellung adaptiert werden. Die Konzeption bestimmter Teilbereiche erforderte ein intensiveres Einarbeiten in die Thematik mit Hilfe der oben genannten Ressourcen und durch den Austausch mit Experten.

Auch mussten immer wieder Abwägungen in Bezug auf die Struktur und den Funktionsumfang gemacht werden, um neben den barrierefreien Aspekten auch die einer möglichst einfachen und universellen Integration in Kundenprojekte zu berücksichtigen.

An einigen Stellen waren bestimmte Vorgaben aufgrund von technischen Bedingungen und Restriktionen schwierig umzusetzen oder erforderten unterschiedliche Herangehensweisen, um einen effizienten Einsatz der Applikation zu ermöglichen. Auch traten immer wieder rein technische Probleme auf – wie die Untauglichkeit von [AJAX](#) Requests Dateien zu übertragen oder die Erweiterung von vorhandenen jQuery UI Applikationen. Diese konnten aber durch umfangreiche Recherche, Einarbeitung und Tests zufriedenstellend gelöst werden.

Die erstellten Applikationen entsprechen den Ansprüchen einer aktuell gestalteten und mit moderner Technik ausgestatteten *State-of-the-art*-Internetseite, genauso wie denen einer barrierefrei gestalteten. Das technisch-konzeptuelle Ziel, die Widgets modular, einfach anpassbar – eben für den Agentur-Alltag gut nutzbar – und im Betrieb performant zu gestalten, wurde erreicht.

Auch wenn nicht jeder Nutzer von allen implementierten Techniken profitieren kann, hilft die Einhaltung der Richtlinien grundsätzlich Jedem bei der Erfassung der Informationen und einer entsprechenden Reaktion auf diese. Mit Fortschreiten der Implementierung in den [UAs](#) und der [AT](#) wird sich der Nutzen spezieller Eigenschaften für Barrierefreiheit weiter erhöhen.

Das größte Problem für Barrierefreiheit im Netz stellt weiterhin das unzureichende Bewusstsein für die Problematik dar. Zu wenig Entwickler beherzigen zumindest einige, einfach umzusetzende und grundsätzliche Regeln für ein zugängliches, offenes Internet. Da nur wenige verbindliche gesetzliche Vorgaben existieren – und diese selbst von Regierungsseite kaum eingehalten werden¹ – kann diesem Missstand nur durch massive Aufklärungs- und Lobbyarbeit abgeholfen werden.

Der Autor wird versuchen, hierzu seinen Beitrag zu leisten.

1 vgl. eAccessibility of public sector services in the European Union - Section 3 Summary of result across EU [network](#) [2005]

KAPITEL 7

Literaturverzeichnis

[eaf 2009] *Diskussion: Harmonisierung von europäischen Barrierefreiheitsrichtlinien.* European Accessibility Forum Frankfurt am Main, Konferenz am 27. März 2009. <http://vimeo.com/4149320>. Version: 2009, Abruf: 04. Mai 2009 (Zitiert auf Seiten 40, 41 und 51)

[Administration 1998] Administration, U.S. General S.: *Section 508 Standards.* Version: 1998. <http://www.section508.gov/index.cfm?FuseAction=Content&ID=12>, Abruf: 18. Juli 2009 (Zitiert auf Seite 31)

[Agency 2006] Agency, Swedish Administrative D.: *Swedish National Guidelines for Public Sector Websites.* Version: 2006. <http://verva.24-timmarswebben.se/upload/english/swedish-guidelines-public-sector-websites.pdf>, Abruf: 18. Juli 2009 (Zitiert auf Seiten 11 und 30)

[ASA 2009] ASA, Opera S.: *Opera 9.5 for Windows Changelog.* Version: 2009. <http://www.opera.com/docs/changelogs/windows/950/>, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 21)

[Bakaus u. the jQuery UI Team. 2009a] Bakaus, Paul ; UI Team. the j.: *UI/Theming/API.* Version: 2009. <http://jqueryui.com/docs/Theming/API>, Abruf: 12. Mai 2009. Internetseite (Zitiert auf Seite 61)

[Bakaus u. the jQuery UI Team. 2009b] Bakaus, Paul ; UI Team. the j.: *UI/Theming/CustomThemes.* Version: 2009. <http://jqueryui.com/docs/Theming/CustomThemes>, Abruf: 12. Mai 2009. Internetseite (Zitiert auf Seite 61)

[Berners-Lee 2009] Berners-Lee, Sir Timothy J.: *Startseite WAI.* Version: 2009. <http://www.w3.org/WAI/>, Abruf: 20. April 2009. Internetseite (Zitiert auf Seite 1)

[BIK 2008] BIK: *Teil 1: Vorstellung der WCAG 2.0.* Version: 2008. <http://www.bik-online.info/info/pruefung/wcag2/vorstellung.php>, Abruf: 14. Juli 2009. Internetseite (Zitiert auf Seite 50)

[for the Blind 2008] Blind, American F. t.: *Facts and Figures on Americans with Vision Loss*. Version: 2008. <http://www.afb.org/Section.asp?SectionID=15&DocumentID=4398>, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)

[Bundesregierung 2008] Bundesregierung, Die: *Antwort auf die Große Anfrage der Abgeordneten Markus Kurth, Irmgard Schewe-Gerigk, Kerstin Andreea, weiterer Abgeordneter und der Fraktion BÜNDNIS 90/DIE GRÜNEN*. Version: 2008. <http://dip21.bundestag.de/dip21/btd/16/092/1609283.pdf>, Abruf: 13. Juli 2009 (16/9283) (Zitiert auf Seite 41)

[der Bundesrepublik Deutschland 2007] Bundesrepublik Deutschland, Auswärtigen A.: *Pressemitteilung: Demografischer Wandel ist auch Chance für Europa*. Version: 2007. http://www.eu2007.de/de/News/Press_Releases/April/0417BMFSFJ.html, Abruf: 26. Juli 2009 (Zitiert auf Seite 11)

[of Canada 2007] Canada, The Treasury B.: *Common Look and Feel for the Internet 2.0*. Version: 2007. <http://www.tbs-sct.gc.ca/clf2-nsi2>, Abruf: 18. Juli 2009 (Zitiert auf Seiten 12 und 31)

[Caspers 2007] Caspers, Tomas: *Fünf Jahre mit der BITV - eine Bestandsaufnahme*. Version: 2007. <http://www.einfach-fuer-alle.de/artikel/bitv-reloaded/>, Abruf: 12. Juli 2009. Internetseite (Zitiert auf Seite 40)

[Caspers 2009a] Caspers, Tomas: *Gleichstellungsgesetze im Bund und in den Ländern*. Version: 2009. <http://www.einfach-fuer-alle.de/artikel/bitv/bgg/>, Abruf: 13. Juli 2009. Internetseite (Zitiert auf Seite 38)

[Caspers 2009b] Caspers, Tomas: *Reine Formsache - Barrierefreie Formulare mit HTML, CSS JavaScript*. Version: 2009. <http://www.einfach-fuer-alle.de/artikel/barrierefreie-formulare-mit-html-css-und-javascript/>, Abruf: 17. Mai 2009. Internetseite (Zitiert auf Seiten 64 und 66)

[Chen 2009] Chen, Charles L.: *Fire Vox: Talking Browser Extension for Firefox*. Version: 2009. http://clc4tts.clcwORLD.net/clc-firevox_doc.html, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 20)

[Community 2009] Community, Codetalks: *Set of ARIA Test Cases*. Version: 2009. http://wiki.codetalks.org/wiki/index.php/Set_of_ARIA_Test_Cases, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 17)

[comScore 2009] comScore, Inc.: *Social Networking Explodes Worldwide as Sites Increase their Focus on Cultural Relevance*. Version: 2009. <http://www.comscore.com/press/release.asp?press=2396>, Abruf: 04. Mai 2009. Internetseite (Zitiert auf Seite 3)

- [Corporation 2009a] Corporation, Microsoft: *Mapping ARIA Roles, States, and Properties to UI Automation*. Version: 2009. [http://msdn.microsoft.com/en-us/library/cc891505\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/cc891505(VS.85).aspx), Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 16)
- [Corporation 2009b] Corporation, Microsoft: *Web Development*. Version: 2009. <http://msdn.microsoft.com/en-us/library/aa155073.aspx>, Abruf: 17. Mai 2009. Internetseite (Zitiert auf Seite 64)
- [Deutschland 2008a] Deutschland, Statistisches B.: *Pressemitteilung Nr.258 vom 17.07.2008*. Version: 2008. http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2008/07/PD08_258_227.psml, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)
- [Deutschland 2008b] Deutschland, Statistisches B.: *Pressemitteilung Nr.406 vom 30.10.2008*. Version: 2008. http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2008/10/PD08_406_227.psml, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)
- [U.S. Department of Education u. Research 2007] Education, National D. o. ; Research, Rehabilitation: *2007 Disability Status Report*. Version: 2007. <http://www.ilr.cornell.edu/edi/disabilitystatistics/>, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)
- [Faulkner 2009a] Faulkner, Steve: *Comparison of ARIA roles exposed via MSAA and UI Automation in IE8*. Version: 2009. <http://www.paciellogroup.com/blog/?p=241>, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seiten 16, 17 und 18)
- [Faulkner 2009b] Faulkner, Steve: *Using WAI ARIA Landmark Roles*. Version: 2009. <http://www.paciellogroup.com/blog/?p=106>, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 19)
- [Faulkner 2009c] Faulkner, Steve: *WAI-ARIA Implementation in JavaScript UI Libraries*. Version: 2009. <http://www.paciellogroup.com/blog/?p=313>, Abruf: 1. Juli 2009. Internetseite (Zitiert auf Seiten 26 und 27)
- [Foundation 2009] Foundation, The D.: *Dojo A11Y Statement*. Version: 2009. <http://www.dojotoolkit.org/developer/a11yStatement>, Abruf: 1. Juli 2009. Internetseite (Zitiert auf Seiten)
- [Freedom Scientific 2009a] Freedom Scientific, Inc.: *JAWS for Windows ® Screen Reading Software*. Version: 2009. <http://www.freedomsci.com/products/fs/jaws-product-page.asp>, Abruf: 18. Juni 2009. Internetseite (Zitiert auf Seite 19)

- [Freedom Scientific 2009b] Freedom Scientific, Inc.: *What's New in JAWS 10.* Version: 2009. <http://www.freedomscientific.com/downloads/jaws/JAWS-whats-new.asp>, Abruf: 18. Juni 2009. Internetseite (Zitiert auf Seite 19)
- [Gemeinschaften 2005] Gemeinschaften, Europäische: *Pressemitteilung IP/05/322.* Version: 2005. http://ec.europa.eu/employment_social/news/2005/mar/demog_gp_de.html, Abruf: 26. Juli 2009 (Zitiert auf Seite 11)
- [Gez Lemon 2007] Gez Lemon, Steve F.: *Improving Ajax applications for JAWS users.* Version: 2007. <http://juicystudio.com/article/improving-ajax-applications-for-jaws-users.php>, Abruf: 3. Juni 2009. Internetseite (Zitiert auf Seite 17)
- [GW Micro 2009] GW Micro, Inc: *Window-Eyes.* Version: 2009. <http://www.gwmicro.com/Window-Eyes/>, Abruf: 18. Juni 2009. Internetseite (Zitiert auf Seite 19)
- [Haenel 2009] Haenel, Matthias: *Die Braillezeile.* Version: 2009. <http://www.matthias-haenel.de/wazeile.html>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 24)
- [Hassell 2009] Hassell, Dr. J.: *Web Accessibility Guidelines in the UK.* European Accessibility Forum Frankfurt am Main, Konferenz am 27. März 2009. <http://vimeo.com/39466790>. Version: 2009, Abruf: 18. Juli 2009 (Zitiert auf Seite 30)
- [Heilmann 2009] Heilmann, Christian: *Accessible Web Applications.* European Accessibility Forum Frankfurt am Main, Konferenz am 27. März 2009. <http://vimeo.com/3924498>. Version: 2009, Abruf: 04. Mai 2009 (Zitiert auf Seite 3)
- [Hoffman 2009] Hoffman, Chris: *jARIA jQuery Plugin.* Version: 2009. <http://www.outstandingelephant.com/jaria/>, Abruf: 1. Juli 2009. Internetseite (Zitiert auf Seite 26)
- [Inc. 2009] Inc., Apple: *VoiceOver Application Support.* Version: 2009. <http://www.apple.com/accessibility/voiceover/applications.html>, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 21)
- [des Innern 2002] Innern, Bundesministerium des: *Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz.* Version: April 2002. <http://bundesrecht.juris.de/bitv/>, Abruf: 12. Juli 2009 (Zitiert auf Seite 38)
- [Johansson 2009] Johansson, Roger: *Safari 4 public beta with WAI-ARIA support. Or not?* Version: 2009. http://www.456bereastreet.com/archive/200903/safari_4_public_beta_with_wai-aria_support_or_not/, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 21)

- [Joshua Hailpern 2009] Joshua Hailpern, Richard Boardman Srinivas A. Loretta Guarino Reid R. Loretta Guarino Reid: *WEB 2.0: Blind to an Accessible New World*. Version: 2009. <http://www2009.org/proceedings/pdf/p821.pdf>, Abruf: 25. Juli 2009 (Zitiert auf Seiten 16 und 17)
- [Kliehm 2009] Kliehm, Martin: *BITV 2.0 am grünen Tisch?* Version: 2009. <http://blog.namics.com/2008/06/bitv-2-0-am-gruenen-tisch.html>, Abruf: 13. Juli 2009. Internetseite (Zitiert auf Seite 41)
- [Kloots 2009] Kloots, Todd: *Enhancing TabView Accessibility with WAI-ARIA Roles and States*. Version: 2009. <http://yuiblog.com/blog/2008/07/30/tabview-aria/>, Abruf: 07. Juli 2009. Internetseite (Zitiert auf Seite 93)
- [Lemon 2009] Lemon, Gez: *Introduction to WAI ARIA*. Version: 2009. <http://dev.opera.com/articles/view/introduction-to-wai-aria/>, Abruf: 19. Mai 2009. Internetseite (Zitiert auf Seite 53)
- [Ltd 2009] Ltd, Dolphin Computer A.: *Prices for Hal.* Version: 2009. <http://www.dolphinuk.co.uk/productdetail.asp?id=5&z=9&curr=USD>, Abruf: 1. Juli 2009. Internetseite (Zitiert auf Seite 19)
- [Machell 2009] Machell, Matt: *Safari 4 - Quickfire ARIA Testing*. Version: 2009. <http://eclecticdreams.com/blog/safari-4-quickfire-aria-testing>, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 21)
- [May 2009] May, Matt: *Inaccessibility of CAPTCHA*. Version: 2009. <http://www.w3.org/TR/turingtest/>, Abruf: 15. Mai 2009. Internetseite (Zitiert auf Seite 65)
- [McClain-Nhlapo 2009] McClain-Nhlapo, Charlotte: *The Business Value of Accessability*. European Accessibility Forum Frankfurt am Main, Konferenz am 27. März 2009. <http://vimeo.com/4153786>. Version: 2009, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)
- [Müller 2009] Müller, Melanie: *Harmonisierung von europäischen Barrierefreiheitsrichtlinien*. European Accessibility Forum Frankfurt am Main, Konferenz am 27. März 2009. <http://vimeo.com/4149140>. Version: 2009, Abruf: 04. Mai 2009 (Zitiert auf Seiten 38 und 41)
- [kabinet nachrichten 2006a] nachrichten kabinet: *EU-Mittel künftig von Barrierefreiheit abhängig*. Version: 2006. http://www.kabinet-nachrichten.org/cipp/kabinet/custom/pub/content_lang_1/oid_11659/, Abruf: 13. Juli 2009. Internetseite (Zitiert auf Seiten 12 und 38)
- [kabinet nachrichten 2006b] nachrichten kabinet: *Sind in Österreich barrierefreie Internetseiten eine Verpflichtung?* Version: 2006. http://www.kabinet-nachrichten.org/cipp/kabinet/custom/pub/content_lang_1/oid_10498, Abruf: 17. Juli 2009. Internetseite (Zitiert auf Seite 11)

[network 2005] network, European Public A.: *eAccessibility of public sector services in the European Union*. Version: 2005. <http://archive.cabinetoffice.gov.uk/e-government/resources/eaccessibility/>, Abruf: 04. August 2009 (Zitiert auf Seite 111)

[Organization 2006] Organization, World H.: *Fact Sheet N°300, Deafness and hearing impairment*. Version: 2006. <http://www.who.int/mediacentre/factsheets/fs300/en/>, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)

[Organization 2009] Organization, World H.: *Fact Sheet N°282, Visual impairment and blindness*. Version: 2009. <http://www.who.int/mediacentre/factsheets/fs282/en/>, Abruf: 26. Juli 2009 (Zitiert auf Seite 10)

[Project. 2009a] Project., The G.: *Orca/Firefox*. Version: 2009. <http://live.gnome.org/Orca/Firefox>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 20)

[Project. 2009b] Project., The G.: *Orca/Firefox/ARIAWidgets*. Version: 2009. <http://live.gnome.org/Orca/Firefox/ARIAWidgets>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 20)

[Project. 2009c] Project., The G.: *Orca/Firefox/LiveRegion*. Version: 2009. <http://live.gnome.org/Orca/Firefox/LiveRegions>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 20)

[Resig u. the jQuery Team 2009] Resig, John ; Team the j.: *UI/Developer Guide*. Version: 2009. http://docs.jquery.com/UI/Developer_Guide, Abruf: 12. Mai 2009. Internetseite (Zitiert auf Seite 61)

[Rieske 2009] Rieske, Thomas: *Behindertengerechtes Internet*. Version: 2009. http://www.tecchannel.de/webtechnik/entwicklung/401717/behindertengerechtes_internet, Abruf: 25. Juni 2009. Internetseite (Zitiert auf Seite 22)

[de Rooij 2009] Rooij, Raph de: *Web Accessibility Guidelines in the Netherlands*. European Accessibility Forum Frankfurt am Main, Konferenz am 27. März 2009. <http://vimeo.com/3955217>. Version: 2009, Abruf: 18. Juli 2009 (Zitiert auf Seiten 11 und 30)

[für Blinde und Sehbehinderte 2009a] Sehbehinderte, Informationspool C. u.: *Braillezeilen*. Version: 2009. <http://www.incobs.de/produktinfos/braillezeilen/>, Abruf: 24. Juni 2009. Internetseite (Zitiert auf Seite 24)

[für Blinde und Sehbehinderte 2009b] Sehbehinderte, Informationspool C. u.: *Finanzierung*. Version: 2009. <http://www.incobs.de/infothek/finanzierung/index.php>, Abruf: 24. Juni 2009. Internetseite (Zitiert auf Seite 15)

- [für Blinde und Sehbehinderte 2009c] Sehbehinderte, Informationspool C. u.: *Vergrößerungssoftware*. Version: 2009. <http://www.incobs.de/produktinfos/grossbild>, Abruf: 24. Juni 2009. Internetseite (Zitiert auf Seite 21)
- [Takayuki 2004] Takayuki, WATANABE: *JIS Web Content Accessibility Guideline*. Version: 2004. <http://www.comm.twcu.ac.jp/~nabe/data/JIS-WAI/>, Abruf: 18. Juli 2009 (Zitiert auf Seite 31)
- [Team 2009a] Team, FireVox P.: *CLC-4-TTS*. Version: 2009. <http://clc4tts.clcworld.net>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 20)
- [Team 2009b] Team, FireVox P.: *FireVox Features*. Version: 2009. <http://firevox.clcworld.net/features.html>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 20)
- [Team 2009c] Team, Mozilla Firefox D.: *(ia2) Support IAccessible2 API*. Version: 2009. https://bugzilla.mozilla.org/show_bug.cgi?id=368873, Abruf: 22. Juni 2009. Internetseite (Zitiert auf Seite 16)
- [Team 2009d] Team, NVDA D.: <http://www.nvda-project.org/wiki/FutureGoals>. Version: 2009. <http://www.nvda-project.org/wiki/FutureGoals>, Abruf: 17. Juni 2009. Internetseite (Zitiert auf Seite 18)
- [Teh 2009a] Teh, James: *Mozilla Grant Progress Report*. Version: 2009. <http://www.nvda-project.org/blog/MozillaGrantDec2007ProgressReport>, Abruf: 14. Juni 2009. Internetseite (Zitiert auf Seite 18)
- [Teh 2009b] Teh, James: *NVDA 0.6p3 Released!* Version: 2009. <http://www.nvda-project.org/blog/NVDA0.6p3Released>, Abruf: 12. Juni 2009. Internetseite (Zitiert auf Seite 18)
- [Thatcher] Thatcher, Jim: *Side by Side WCAG vs. 508*. <http://www.jimthatcher.com/sidebyside.htm>, Abruf: 18. Juli 2009. Internetseite (Zitiert auf Seite 31)
- [Tina Kohler 2005] Tina Kohler, Informatikrat Bund S. BIT: *P028 - Richtlinien des Bundes für die Gestaltung von barrierefreien Internetangeboten - Version 1.0*. Version: 2005. <http://www.isb.admin.ch/themen/standards/alle/03237/>, Abruf: 18. Juli 2009 (Zitiert auf Seiten 11 und 30)
- [UK 2007] UK, Cabinet O.: *Naming and registering websites*. Version: 2007. http://www.cabinetoffice.gov.uk/government_it/web_guidelines/consultations.aspx, Abruf: 18. Juli 2009 (Zitiert auf Seite 11)
- [der europäischen Union 2009] Union, Rat der e.: *Mitteilung an die Presse - 2935. Tagung des Rates - Verkehr, Telekommunikation und Energie*. Version: 2009.

http://www.consilium.europa.eu/ueDocs/cms_Data/docs/pressdata/de/trans/107248.pdf, Abruf: 18. Juli 2009 (Zitiert auf Seiten 12 und 31)

[für Normierung e. V. 2008a] V., DIN Deutsches I. e.: *Ergonomie der Mensch-System-Interaktion – Teil 151: Leitlinien zur Gestaltung von Benutzerschnittstellen für das World Wide Web (ISO 9241-151:2008); Deutsche Fassung EN ISO 9241-151:2008*. September 2008 (9241 - Ergonomie der Mensch-System-Interaktion) (Zitiert auf Seite 33)

[für Normierung e. V. 2008b] V., DIN Deutsches I. e.: *Ergonomie der Mensch-System-Interaktion – Teil 171: Leitlinien für die Zugänglichkeit von Software (ISO 9241-171:2008); Deutsche Fassung EN ISO 9241-171:2008*. Juni 2008 (9241 - Ergonomie der Mensch-System-Interaktion) (Zitiert auf Seite 36)

[W3C 2009] W3C: *HTML 5*. Version: 2009. <http://dev.w3.org/html5/spec/Overview.html>, Abruf: 18. Mai 2009. Internetseite (Zitiert auf Seite 52)

[WAI 1999] WAI: *Web Content Accessibility Guidelines 1.0*. Version: 1999. <http://www.w3.org/TR/WCAG10/>, Abruf: 14. Juli 2009. Internetseite (Zitiert auf Seite 51)

[WAI 2009a] WAI: *Accessible Rich Internet Applications (WAI-ARIA) 1.0*. Version: 2009. <http://www.w3.org/TR/wai-aria/>, Abruf: 12. Mai 2009. Internetseite (Zitiert auf Seiten 52, 53 und 93)

[WAI 2009b] WAI: *States and Adaptable Properties Module*. Version: 2009. <http://www.w3.org/WAI/PF/adaptable/StatesAndProperties-20051106.html>, Abruf: 19. Mai 2009. Internetseite (Zitiert auf Seite 52)

[WAI 2009c] WAI: *Techniques for WCAG 2.0*. Version: 2009. <http://www.w3.org/TR/WCAG20-TECHS/>, Abruf: 14. April 2009. Internetseite (Zitiert auf Seiten 51, 60, 64, 65, 66, 79, 80, 81, 92, 93, 97 und 98)

[WAI 2009d] WAI: *Understanding WCAG 2.0*. Version: 2009. <http://www.w3.org/TR/UNDERSTANDING-WCAG20>, Abruf: 14. April 2009. Internetseite (Zitiert auf Seite 48)

[WAI 2009e] WAI: *WAI-ARIA 1.0 User Agent Implementation Guide*. Version: 2009. <http://www.w3.org/WAI/PF/aria-implementation/>, Abruf: 6. Juni 2009. Internetseite (Zitiert auf Seite 16)

[WAI 2009f] WAI: *WAI-ARIA Best Practices*. Version: 2009. <http://www.w3.org/TR/wai-aria-practices/>, Abruf: 14. Mai 2009. Internetseite (Zitiert auf Seiten 52, 53, 56, 80, 92, 93, 98 und 99)

[WAI 2009g] WAI: *WAI-ARIA FAQ*. Version: 2009. <http://www.w3.org/WAI/aria/faq.html>, Abruf: 19. Mai 2009. Internetseite (Zitiert auf Seite 52)

- [WAI 2009h] WAI: *WCAG 2.0 Translations*. Version: 2009. <http://www.w3.org/WAI/WCAG20/translations>, Abruf: 06. Mai 2009. Internetseite (Zitiert auf Seite 42)
- [WAI 2009i] WAI: *Web Content Accessibility Guidelines (WCAG) 2.0*. Version: 2009. <http://www.w3.org/WAI/WCAG20/translations>, Abruf: 06. Mai 2009. Internetseite (Zitiert auf Seiten 42, 43, 44, 45, 46, 47, 48, 49, 62 und 79)
- [Warnke 2006] Warnke, Karsten: *Entstehung der BITV*. Version: 2006. <http://www.barrierefreies-webdesign.de/bitv/entstehung.html>, Abruf: 15. Juli 2009. Internetseite (Zitiert auf Seite 38)
- [WebAIM 2009] WebAIM, Utah State U.: *Survey of Preferences of Screen Readers Users*. Version: 2009. <http://webaim.org/projects/screenreadersurvey/>, Abruf: 3. Juni 2009. Internetseite (Zitiert auf Seite 17)
- [Wikipedia 2009a] Wikipedia: *Barrierefreies Internet — Wikipedia, Die freie Enzyklopädie*. Version: 2009. http://de.wikipedia.org/w/index.php?title=Barrierefreies_Internet&oldid=59015278, Abruf: 20. April 2009. Internetseite (Zitiert auf Seiten 11 und 12)
- [Wikipedia 2009b] Wikipedia: *Braillezeile*. Version: 2009. <http://de.wikipedia.org/wiki/Braillezeile>, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 24)
- [Wikipedia 2009c] Wikipedia: *Screen magnifier*. Version: 2009. http://en.wikipedia.org/wiki/Screen_magnifier, Abruf: 23. Juni 2009. Internetseite (Zitiert auf Seite 21)
- [Wikipedia 2009d] Wikipedia: *Webbrowser*. Version: 2009. <http://de.wikipedia.org/wiki/Webbrowser>, Abruf: 25. Juni 2009. Internetseite (Zitiert auf Seite 22)
- [Zapp 2005] Zapp, Michael: *BITV - noch zeitgemäß?* Version: 2005. http://www.bik-online.info/info/pruefung/bitv_zeitgemaess.php, Abruf: 12. Juli 2009. Internetseite (Zitiert auf Seite 40)
- [Zehe 2009] Zehe, Marco: *NVDA 0.6p3 released, quite some news for Mozilla users!* Version: 2009. <http://www.marcozehe.de/2009/02/16/nvda-06p3-released-quite-some-news-for-mozilla-users/>, Abruf: 14. Juni 2009. Internetseite (Zitiert auf Seite 18)
- [Österreich] Österreich, Bundeskanzleramt: *Web-Accessibility - Internet Zugang für alle* (Zitiert auf Seite 30)
- [Österreich] Österreich, Republik: *Gesetzliche Situation auf europäischer Ebene* (Zitiert auf Seite 11)

KAPITEL 8

Abbildungsverzeichnis

3.1	Nutzung einer softwaregestützten Bildschirmlupe	22
3.2	Screenshot des textbasierten Browsers w3m	23
3.3	Tastatur mit Braillezeile der Firma EuroBraille	25
5.1	formValidator: Funktionsübersicht	72
5.2	lightbox: Funktionsübersicht	86
5.3	ariaSorTable: Funktionsübersicht	105

KAPITEL 9

Quellennachweise für Abbildungen

Abbildung 3.3:

<http://commons.wikimedia.org>

Abbildung 3.1:

Stiftung Zugang für alle

Abbildung 3.2:

<http://de.wikipedia.org>

KAPITEL 10

Quellcodeverzeichnis

5.1	formValidator: 113-137: Aufruf des Widgets	67
5.2	formValidator: 205-214: checkCaptcha Callback-Funktion	70
5.3	formValidator: 231-233: destroy Methode	71
5.4	formValidator.js 82-106: Anpassungen Schalter Live-Validierung	74
5.5	formValidator.js 113-139: Anpassungen Hover/Focus Events	75
5.6	formValidator.js 308-322: Anpassungen Fehlerklasse entfernen	75
5.7	formValidator.js 323-344: Anpassungen Fehler sammeln	76
5.8	formValidator.js 350-360: Anpassungen Fehlerklasse hinzufügen	77
5.9	formValidator.js 369-384: Anpassungen Fehlermeldung injizieren	77
5.10	formValidator.js 489-503: Anpassungen Erfolgsmeldung	78
5.11	ariaLightbox: injiziertes HTML-Konstrukt	82
5.12	ariaLightbox: CSS	83
5.13	ariaLightbox: 63-69: Aufruf des Widgets	83
5.14	ariaLightbox: 102-104: startGallery Methode	86
5.15	ariaLightbox.js 118-139: Anpassungen HTML-Konstrukt	87
5.16	ariaLightbox.js 191-215: Positionierung des HTML-Konstrukts	88
5.17	ariaLightbox.js 237-277: Ändern des Bildes	89
5.18	ariaLightbox.js 303-329: Status der Buttons	90
5.19	ariaTabs: 11-23: Aufruf	94
5.20	ariaTabs.js: 18-26: Aufruf	94
5.21	ariaTabs.js: 127-132: Collapsible-Funktionalität abbilden	95
5.22	ariaTabs.js: 135-153: add Funktionalität	96
5.23	ariaSorTable: Tabellenkopf	100
5.24	ariaSorTable: Aufruf	102
5.25	ariaSorTable: Erweiterung	107
B.1	formValidator: HTML Quellcode	137
B.2	formValidator: JavaScript Quellcode	143
B.3	lightbox: HTML Quellcode	158

B.4	lightbox: JavaScript Quellcode	162
B.5	ariaTabs: JavaScript Quellcode	174
B.6	sorTable: HTML Quellcode	178
B.7	sorTable: JavaScript Quellcode	180

KAPITEL 11

Tabellenverzeichnis

3.1	Java-Script Frameworks	27
5.1	formValidator: Übersicht Regeln	68
5.2	formValidator: Übersicht regulärer Ausdrücke	68
5.3	formValidator: Übersicht Optionen	69
5.4	formValidator: Übersicht Events	70
5.5	formValidator: Übersicht Methoden	71
5.6	ariaLightbox: Übersicht Optionen	84
5.7	ariaLightbox: Übersicht Events	85
5.8	ariaLightbox: Übersicht Methoden	85
5.9	ariaSorTable: Übersicht Sortermöglichkeiten	100
5.10	ariaSorTable: Übersicht Optionen	101
5.11	ariaSorTable: Beispiel für Option colScopeRow	102
5.12	ariaSorTable: Übersicht Events	103
5.13	ariaSorTable: Übersicht Methoden	103
5.14	ariaSorTable: Übersicht interne Optionen	106

KAPITEL 12

Abkürzungsverzeichnis

Zur Verbesserung des Überblicks wurden einige Begriffserklärungen mit dem Abkürzungsverzeichnis zusammengefasst.

Für die Abkürzungen wurde meist ein Blick in [Wikipedia](#) geworfen.

API Application Programming Interface; eine Schnittstelle, die von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird.

AJAX Asynchronous JavaScript and XML; ein Konzept der asynchronen Datenübertragung, das es ermöglicht, vom Server geladene Daten im Browser verfügbar zu machen, ohne die Seite neu zu laden (umgs. „nachladen“).

ARIA Accessible Rich Internet Applications; mehr Informationen siehe Punkt [4.4](#)

AT Assistive Technology; Unterstützungstechnologie; befasst sich mit technischen Hilfen speziell für Menschen mit Behinderungen, zum Beispiel Braillezeilen oder Bildschirmleseprogramme

AT-SPI Assistive Technology Service Provider Interface; Schnittstelle des Betriebssystems zur Bereitstellung von behindertengerechten Benutzeroberflächen

BITV Barrierefreie Informationstechnik-Verordnung

BGG Gesetz zur Gleichstellung behinderter Menschen

BSD ist eine Lizenz für freie Software, entstanden ist die Lizenz an der University of California, Berkeley. Sie erlaubt kopieren, verändern und verbreiten, es muss aber ein Copyright-Vermerk verbleiben enthält aber kein Copyleft.

BIS The British Standards Institution

css Cascading Style Sheets

CLC-4-TTS Core Library Components for Text-To-Speech

CMS Content Management System

CQ Communiqué; ein auf Java basierendes Enterprise-Content-Management-System der Firma Day; aktuell in Version 5 (CQ5).

Captcha Completely Automated Public Turing test to tell Computers and Humans Apart

DOM Document Object Model; eine Spezifikation einer Schnittstelle für den Zugriff auf HTML- oder XML-Dokumente.

DTD Dokumenttypdefinition; ein Satz an Regeln, der benutzt wird, um Dokumente eines bestimmten Typs zu deklarieren.

EAFRA Europäisches Accessibility Forum Frankfurt am Main

GPL General Public License; eine von der Free Software Foundation herausgegebene Lizenz mit Copyleft für die Lizenzierung freier Software.

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

ICF International Classification of Functioning, Disability and Health; eine von der WHO erstellte und herausgegebene Klassifikation zur Beschreibung des funktionalen Gesundheitszustandes, der Behinderung, der sozialen Beeinträchtigung sowie der relevanten Umweltfaktoren von Menschen.

IE Microsoft Internet Explorer

IBM International Business Machines Corporation

ID Identifikator; in Bezug auf HTML und CSS das eindeutig identifizierende Attribut

ISO Internationale Organisation für Normung

JS Java-Script

JAWS Job Access with Speech; Screenreader Software

JIS Japan Industrial Standard

JSON JavaScript Object Notation

MS Microsoft

MSAA Microsoft Active Accessibility; Schnittstelle des Betriebssystems zur Bereitstellung von behindertengerechten Benutzeroberflächen

MIT MIT-Lizenz; eine aus dem Massachusetts Institute of Technology stammende Lizenz für die Benutzung verschiedener Arten von Computersoftware. Sie erlaubt die Wiederverwendung der unter ihr stehenden Software sowohl für Software, deren Quelltext frei einsehbar ist, als auch für Software, deren Quelltext nicht frei einsehbar ist.

NVDA Nonvisual Desktop Access; Screenreader Software

OS Operating System; Betriebssystem

PHP Hypertext Preprocessor

PR Public Relations; Öffentlichkeitsarbeit

RegEx Regular Expression; Regulärer Ausdruck

RIA Rich Internet Application; komplexe, meist benutzerfreundliche Webapplikationen, die mit modernen Techniken funktionieren.

SEO Search Engine Optimization

SAPI Speech Application Programming Interface; Schnittstelle zur Anbindung von Bibliotheken zur Sprachsynthese und Spracherkennung unter dem Betriebssystem MS Windows.

TTS Text-To-Speech

URI Uniform Resource Identifier; im Internet für gewöhnlich eine URL [URL](#)

URL Uniform Resource Locator

UA User Agent; Client-Programm, mit dem ein Netzwerkdienst genutzt werden kann. Es ist die Schnittstelle zum Benutzer, die die Inhalte darstellt und Befehle entgegennimmt.

UIA UI Automation; Weiterentwicklung der MSAA

USB Universal Serial Bus

WCAG Web Content Accessability Guidelines

WAI Web Accessibility Initiative; treibt die Barrierefreiheit im Internet voran und entwickelt dafür Standards. Sie möchte u. a. ein internationales Forum für Industrie, Forschung, Behindertenverbände, Regierungen und andere Interessierte bieten.

W3C World Wide Web Consortium; ein Gremium zur Standardisierung von Internet-technologien.

XML Extensible Markup Language

XHTML Extensible HyperText Markup Language

ANHANG A

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Hamburg, den 13. August 2009

ANHANG B

Angefügte Dokumente

B.1 Formular

B.1.1 HTML Quellcode

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/
TR/xhtml11/DTD/xhtml11.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
3 <head>
4   <meta http-equiv="content-type" content="text/html; charset=ISO
     -8859-1" />
5   <meta name="language" content="DE, AT, CH" scheme="DCTERMS.RFC3066"
     />
6   <meta name="creator" content="Felix Nagel, http://www.felixnagel.
     com for Namics (Deutschland) GmbH, http://www.namics.com" />
7   <title>ui.formValidator – jQuery UI</title>
8   <link type="text/css" href="css/ui-lightness/jquery-ui-1.7.1.custom
     .css" rel="stylesheet" />
9   <link type="text/css" href="css/style.css" rel="stylesheet" />
10  <!--[if gte IE 6]>
11    <style type="text/css">@import url(css/style_ie.css);</style>
12  <![endif]-->
13 </head>
14 <body>
15 <h1>jQuery UI – ui.formValidator</h1>
16 <div id="controls">
17   <a href="#" id="destroy">destroy</a> / <a href="#" id="disable">
     disable</a> / <a href="#" id="send">send</a>
18 </div>
19 <div id="wrapper">
20   <form id="form" action="ui.formValidator.html" method="post"
     enctype="multipart/form-data">
21     <div id="ui-formular-info" class="info ui-state-highlight ui-
       corner-all">
22       <p>
```

```
23      <span class="ui-icon ui-icon-info" style="float: left; margin
24          -right: 0.3em;"></span>
25      Felder mit einem <em>*</em> sind Pflichtfelder.
26      </p>
27  </div>
28  <div id="ui-formular-error"></div>
29  <fieldset class="ui-widget-content ui-corner-all">
30      <legend class="ui-widget-header ui-corner-all">Inputfelder</
31          legend>
32
33      <label for="inputtext">Einzeiliges Eingabefeld</label>
34      <input type="text" id="inputtext" name="inputtext" class="text
35          ui-widget-content ui-corner-all" />
36
37      <label for="txtarea">Mehrzeiliger Eingabebereich <em title="
38          Pflichtfeld">*</em></label>
39      <textarea id="txtarea" name="txtarea" rows="" cols="" class="text
40          ui-widget-content ui-corner-all"></textarea>
41
42      <label for="password1">Passwortfeld <em title="Pflichtfeld">*</
43          em></label>
44      <input name="password1" id="password1" type="password" class="text
45          ui-widget-content ui-corner-all" />
46
47      <label for="password2">Passwortfeld Wiederholung <em title="
48          Pflichtfeld">*</em></label>
49      <input name="password2" id="password2" type="password" class="text
50          ui-widget-content ui-corner-all" />
51  </fieldset>
52  <fieldset class="ui-widget-content ui-corner-all">
53      <legend class="ui-widget-header ui-corner-all">Checkboxen</
54          legend>
55      <div class="pair">
56          <input type="checkbox" id="checkbox_1" name="checkboxset"
57              class="radio" />
```

```
57      <div class="pair">
58          <input type="checkbox" id="checkbox_4" name="checkboxset"
59              class="radio" />
60          <label class="ui-widget-content ui-corner-all" for="checkbox_4">Option 4</label>
61      </div>
62  </fieldset>
63
64  <fieldset class="ui-widget-content ui-corner-all">
65      <legend class="ui-widget-header ui-corner-all">Radioboxen</legend>
66      <div class="pair">
67          <input type="radio" id="vote_1" value="Option 1" name="vote"
68              class="radio"/>
69          <label class="ui-widget-content ui-corner-all" for="vote_1">Option 1</label>
70      </div>
71      <div class="pair">
72          <input type="radio" id="vote_2" name="vote" class="radio"/>
73          <label class="ui-widget-content ui-corner-all" for="vote_2">Option 2</label>
74      </div>
75      <div class="pair">
76          <input type="radio" id="vote_3" name="vote" class="radio"/>
77          <label class="ui-widget-content ui-corner-all" for="vote_3">Option 3</label>
78      </div>
79  </fieldset>
80
81  <fieldset class="ui-widget-content ui-corner-all">
82      <legend class="ui-widget-header ui-corner-all">Sonstige
83          Formularfelder</legend>
84
85      <label for="selectset">Auswahlliste</label>
86      <select name="selectset" id="selectset" multiple="multiple"
87          size="5" class="text ui-widget-content ui-corner-all">
88          <option>Option 1</option>
89          <option>Option 2</option>
90          <option>Option 3</option>
91          <option>Option 4</option>
92          <option>Option 5</option>
93      </select>
94
95      <label for="upload">Datei Upload</label>
96      <input type="file" name="upload" id="upload" class="ui-widget-
          content" accept="image/jpeg, image/gif, image/png" />
</fieldset>
97
98  <fieldset class="ui-widget-content ui-corner-all">
99      <legend class="ui-widget-header ui-corner-all">Buttons</legend>
```



```
127     required: "Der einzeilige Eingabebereich muss
128         ausgefüllt werden." ,
129     regEx: "Das einzeilige Eingabefeld muss eine Email
130         Adresse enthalten."
131 },
132 txtarea: {
133     rules: {
134         required: true ,
135         lengthMin: 2,
136         lengthMax: 10
137     },
138     msg: {
139         required: "Der mehrzeiliger Eingabebereich muss
140             ausgefüllt werden." ,
141             length: "Das mehrzeilige Eingabefeld muss zwischen 2
142                 und 10 Zeichen enthalten."
143     },
144 password1: {
145     rules: {
146         lengthMin: 6,
147         lengthMax: 15,
148         required: false
149     },
150     msg: {
151         required: "Das erste Passwortfeld muss ausgefüllt
152             werden." ,
153             length: "Das erste Passwortfeld muss muss zwischen 6
154                 und 15 Zeichen enthalten."
155     }
156 },
157 password2: {
158     rules: {
159         required: false ,
160         equalTo: "password1"
161     },
162     msg: {
163         required: "Das zweite Passwortfeld muss ausgefüllt
164             werden." ,
165             equalTo: "Die Passwörter müssen übereinstimmen."
166     }
167 },
168 checkboxset: {
169     rules: {
170         lengthMin: 2,
171         lengthMax: 3
172     },
173     msg: {
```

```
169         length: "Es müssen zwischen 2 und 3 Checkboxen  
170         aktiviert werden."  
171     },  
172     vote: {  
173         rules: {  
174             required: true  
175         },  
176         msg: {  
177             required: "Eines der Radioboxen muss ausgefüllt werden."  
178             "  
179         }  
180     },  
181     selectset: {  
182         rules: {  
183             lengthMin: 2,  
184             lengthMax: 3,  
185             required: true  
186         },  
187         msg: {  
188             length: "Es müssen zwischen 2 und 3 Einträge aktiviert  
189             werden.",  
190             required: "Ein Eintrag muss gewählt werden."  
191         }  
192     },  
193     upload: {  
194         rules: {  
195             required: false,  
196             regEx: ".(?:jpg|jpeg|gif|png)"  
197         },  
198         msg: {  
199             required: "Es muss eine Datei hochgeladen werden.",  
200             regEx: "Die hochgeladene Datei muss ein Bild sein."  
201         }  
202     },  
203     confirm: {  
204         rules: {  
205             required: true  
206         },  
207         msg: {  
208             required: "Bitte kontrollieren Sie die Daten."  
209         }  
210     },  
211     captcha: {  
212         rules: {  
213             required: true,  
214             regEx: "captcha"  
215         },  
216         msg: {  
217             required: "Das Captcha muss eingegeben werden."  
218         }  
219     }  
220 }
```

```

216         regEx: "Das Captcha ist falsch eingegeben"
217     }
218   },
219   validateLive: true ,
220   disabled: false ,
221   submitHowTo: "iframe" ,
222   submitUrl: "server.php" ,
223   onInit: function(){
224     //console.warn("Init abgeschlossen");
225   },
226   onShowErrors: function(){
227     //console.warn("Error angezeigt");
228   },
229   checkCaptcha: function(event, value){
230     return (value == "123456") ? true : false;
231   }
232 );
233
234 $("#destroy").click(function (event) {
235   formular.formValidator('destroy');
236 });
237 $("#disable").click(function (event) {
238   formular.formValidator('disable');
239 });
240 $("#send").click(function (event) {
241   formular.formValidator('formSubmitted');
242 });
243
244 });
245 </script>
246 </body>
247 </html>

```

Listing B.1: formValidator: HTML Quellcode

B.1.2 JavaScript Quellcode

```

1 /*
2 * jQuery UI FormValidator 1.0 (01.07.09)
3 *
4 * Copyright (c) 2009 Felix Nagel for Namics (Deutschland) GmbH
5 * Licensed under Creative Commons Attribution-Share Alike 3.0
6 * Unported (http://creativecommons.org/licenses/by-sa/3.0/)
7 * Depends: ui.core.js
8
9 USAGE ::::::::::::::::::::
10 * Take a look in the html file or the (german) pdf file delivered
   with this example

```

```
11 * To validate a form element specify its properties in the options
12 forms array:
13 options
14   forms
15     ID [of the element]
16     rules
17     msgs
18   * Forms Array and its children are necessary , possbile values for
19     rules and msg are:
20       required
21       lengthMin
22       lengthMax
23       equalTo
24       regEx
25   * Use a regular expression or a predefined value for the RegEx rule:
26     number
27     numberDE
28     numberISO
29     email
30     url
31     plz
32     dateDE
33     dateISO
34     captcha (this is a callback for server side validation ,
35       look example)
36
37 * Other widget options are:
38 validateLive Boolean turn on or off live validation
39 validateTimeout Number time till live validation
40 validateOff String msg for disabling live validation
41 validateOn String msg for disabling live validation
42 submitHowTo String ajax , iframe (for "ajax" and file upload) , post
43   (native)
44 submitUrl String url for ajax and iframe submition
45 submitError String predefined error msg
46 submitSuccess String predefined succes msg
47 disabled Boolean disable widget
48
49 * Callbacks
50 onInit
51 onformSubmitted
52 onShowErrors
53 onShowSuccess
54 checkCaptcha (must deliver a boolean value)
55
56 * public Methods
57 disable
58 destroy
59 enable
60 formSubmitted submits the form
```

```
57
58  */
59 (function(\$) {
60
61 \$ .widget( "ui.formValidator" , {
62
63     _init: function() {
64         var options = this.options , self = this;
65
66         // add virtual buffer form / should be added immediatly
67         self._updateVirtualBuffer();
68
69         // set submitUrl to form action if no one is defined
70         if (options.submitUrl == "") options.submitUrl = self.element.
71             attr("action");
72
73         // prevent submitting the form
74         self.element.submit( function (event) {
75             // check if widget is disabled or temp set to disabled by
76             // script cause we need to post data
77             if (!options.disabled) {
78                 //event.preventDefault();
79                 self.formSubmitted();
80             }
81             return options.disabled;
82         });
83         // add info Text and provide link to prevent live validating
84         if(options.validateLive && !options.disabled) {
85             // add the deactivate live validation message
86             self.element.find("#ui-formular-info").append("\t<p><a id=\"ui-
87                 formular-live\" href=\"#nogo\">" + options.validateOff +"</a
88                 ></p>\n\t\t");
89
90             self._updateVirtualBuffer();
91
92             // toggle live validating and text of the link
93             self.element.find("#ui-formular-live").toggle(
94                 function () {
95                     options.validateLive = false;
96                     $(this).attr("aria-live","polite")
97                         .attr("aria-relevant","text")
98                         .html(options.validateOn);
99                     self._updateVirtualBuffer();
100                },
101                function () {
102                    options.validateLive = true;
103                    $(this).attr("aria-live","polite")
104                        .attr("aria-relevant","text")
105                        .html(options.validateOff);
```

```
103         self._updateVirtualBuffer();
104     }
105   );
106 }
107 // set hover and focus for reset and submit buttons
108 self._makeHover(self.element.find("input:submit, input:reset"));
109
110 // get the error array
111 errors = self.options.errorsArray;
112
113 // go through every given form element
114 $.each(options.forms, function(id){
115   // instance the associative arry with index = id of the form
116   // element
117   errors[id] = [];
118
119   // save element and which form type / add event handler / ARIA
120   var element = self.element.find("#"+id);
121   //check if radio group or checkbox group or single checkbox
122   if (!element.length) {
123     // get all group elements
124     element = self.element.find("input[name='"+id+"']");
125     // no element found? Only developers should see this
126     if (!element.length) {
127       alert("Error: Configuration corrupted!\n\nCan't find element
128         with id or name = "+id);
129     } else {
130       value = "group";
131       // change label class when hover the label
132       self._makeHover(element.next());
133       // change label class when hover the form element
134       element.bind("mouseenter", function(){ $(this).next().
135         addClass('ui-state-hover'); })
136         .bind("mouseleave", function(){ $(this).next().
137           removeClass('ui-state-hover'); })
138         .bind("focus", function(){ $(this).next().addClass('ui-
139           state-focus'); })
140         .bind("blur", function(){ $(this).next().removeClass('ui-
141           state-focus'); });
142     }
143   } else {
144     // form element hover
145     self._makeHover(element);
146     // ARIA
147     if (options.forms[id].rules.required) {
148       element.attr("aria-required", true);
149     }
150     if (element[0].nodeName.toLowerCase() == "select") {
151       // element is a selectfield
152       value = "select";
```

```
147     } else {
148         // normal textinput or textarea or file upload
149         value = "single";
150     }
151 }
152 // save info
153 options.forms[id].element = element;
154 options.forms[id].type = value;
155
156
157 // dont bind events if live validation is disabled
158 if (options.validateLive) {
159     // necessary for not getting too much events
160     if (options.forms[id].type != "group") {
161         var eventBinder = "keypress";
162         // options need change events
163         if (options.forms[id].type == "select") eventBinder = "
164             change";
165     } else {
166         var eventBinder = "change";
167     }
168     // add event listener
169     options.forms[id].element.bind(eventBinder, function () {
170         // look up if live validation is turned off or widget is
171         // disabled
172         if (options.validateLive && !options.disabled) {
173             // delete old timeout
174             if(options.forms[id].timeout) window.clearTimeout(options
175                 .forms[id].timeout);
176             // extend timeout to prevent server overload
177             if (id == "captcha"){
178                 var time = options.validateTimeout*3;
179             } else {
180                 var time = options.validateTimeout;
181             }
182             // wait before fire event
183             options.forms[id].timeout = window.setTimeout(function ()
184                 {
185                     self._validator(options.forms[id].element, id, errors);
186                     self._showErrors(false);
187                 }, time);
188             }
189         });
190     // save empty errors array
191     options.errorsArray = errors;
192     // Callback
193     self._trigger("onInit", 0);
194 },
195 }
```

```
193
194 // called when interact with the form / validates the forms /
195 // manages which rule applies to which element
196 _validator: function(element, id, errors) {
197     var options = this.options, self = this;
198     // get value of the form element(s)
199     var elementValue = self._getValue(id);
200
201     // got through every rule and its ruleValue of every given form
202     // element
203     $.each(options.forms[id].rules, function(rule, ruleValue){
204         if (elementValue == "") {
205             // unset required error if no form value given and form is
206             // not required
207             if (rule != "required") errors[id][rule] = self._whichError(
208                 true, errors[id][rule]);
209             // if form is required set error
210             if (rule == "required" && ruleValue) errors[id][rule] = self.
211                 _whichError(false, errors[id][rule]);
212         } else {
213             // unset required error if form has some value
214             if (rule == "required" && ruleValue) errors[id][rule] = self.
215                 _whichError(true, errors[id][rule]);
216             switch (rule) {
217                 case "regEx":
218                     switch (ruleValue) {
219                         case "number":
220                             errors[id][rule] = self._whichError(self._number(
221                                 elementValue), errors[id][rule]);
222                             break;
223                         case "numberDE":
224                             errors[id][rule] = self._whichError(self._numberDE(
225                                 elementValue), errors[id][rule]);
226                             break;
227                         case "numberISO":
228                             errors[id][rule] = self._whichError(self._numberISO(
229                                 elementValue), errors[id][rule]);
230                             break;
231                         case "email":
232                             errors[id][rule] = self._whichError(self._email(
233                                 elementValue), errors[id][rule]);
234                             break;
235                         case "url":
236                             errors[id][rule] = self._whichError(self._url(
237                                 elementValue), errors[id][rule]);
238                             break;
239                         case "plz":
240                             errors[id][rule] = self._whichError(self._plz(
241                                 elementValue), errors[id][rule]);
242                             break;
```

```

231
232     case "dateDE":
233         errors[id][rule] = self._whichError(self._dateDE(
234             elementValue), errors[id][rule]);
235         break;
236     case "dateISO":
237         errors[id][rule] = self._whichError(self._dateISO(
238             elementValue), errors[id][rule]);
239         break;
240     case "captcha":
241         errors[id][rule] = self._whichError(self._captcha(
242             elementValue), errors[id][rule]);
243         break;
244     }
245     break;
246     case "lengthMin":
247         errors[id][rule] = self._whichError(self._lengthMin(
248             elementValue, ruleValue), errors[id][rule]);
249         break;
250     case "lengthMax":
251         errors[id][rule] = self._whichError(self._lengthMax(
252             elementValue, ruleValue), errors[id][rule]);
253         break;
254     case "equalTo":
255         errors[id][rule] = self._whichError(self._equalTo(
256             elementValue, ruleValue), errors[id][rule]);
257         break;
258     }
259     // save errors
260     self.options.errorsArray = errors;
261 },
262
263 // called when form is submitted
264 formSubmitted: function() {
265     var options = this.options, self = this;
266     // Callback
267     self._trigger("onformSubmitted", 0);
268
269     // delete success or error message
270     self.element.find("#ui-formular-success").remove();
271
272     // get errors array
273     errors = self.options.errorsArray;

```

```
274 // got through every given form element
275 $.each(options.forms, function(id){
276     // is a group of radio buttons or checkboxes already validated?
277     var groupValidated = false;
278     // check if the defined id elements exist in the DOM
279     //var element = self.element.find("#"+id);
280     var element = options.forms[id].element;
281     if (options.forms[id].type == "single") {
282         self._validator(element, id, errors);
283     // if not it must be a radiobox group or a checkbox group
284    } else {
285        // check if the is already validated
286        if (!groupValidated) {
287            groupValidated = true;
288            self._validator(element, id, errors);
289        }
290    }
291 });
292
293     self._showErrors(true);
294 },
295
296 // called when forms are validated / write errorsArray to DOM /
297 // Take care of ARIA
298 _showErrors: function(submitted){
299     var options = this.options, self = this;
300     var isError, addError, removeError = false;
301     var msgs = msg = "";
302     // get error array
303     var errors = self.options.errorsArray;
304
305     // got through every error form element
306     for (var id in errors){
307         // needed to ensure error Class isn't removed if required error
308         // still exists
309         var failure = false;
310         for (var rule in errors[id]){
311             // set error as corrected
312             if (errors[id][rule] == "corrected") {
313                 var target = options.forms[id].element;
314                 // ARIA
315                 target.attr("aria-invalid", false);
316                 // check for radio group or checkbox group
317                 if (options.forms[id].type == "group") {
318                     target = target.next();
319                 }
320                 // unhighlight error field
321                 target.removeClass("ui-state-error");
322                 // ARIA: old error deleted
323                 removeError = true;
```

```

322     }
323     if( errors[id][rule] == "new" || errors[id][rule] == "old" ) {
324         switch (rule) {
325             case "required":
326                 msg = options.forms[id].msg.required;
327                 break;
328             case "regEx":
329                 msg = options.forms[id].msg.regEx;
330                 break;
331             case "lengthMin":
332                 msg = options.forms[id].msg.length;
333                 break;
334             case "lengthMax":
335                 msg = options.forms[id].msg.length;
336                 break;
337             case "equalTo":
338                 msg = options.forms[id].msg.equalTo;
339                 break;
340         }
341         msgs += '           <li><a href="#'+id+'>'+msg+'</a></li>\n';
342         ;
343         // there are errors to show
344         isError = failure = true;
345     }
346     if(errors[id][rule] == "new") {
347         // ARIA: new error added
348         addError = true;
349     }
350     // check at last if there is an error so error class wont be
351     // removed
352     if (failure) {
353         var target = options.forms[id].element;
354         target.attr("aria-invalid", true);
355         // check for radio group or checkbox group
356         if (options.forms[id].type == "group") {
357             target = target.next();
358         }
359         // unhighlight error field
360         target.addClass("ui-state-error");
361     }
362     // take care of ARIA
363     var aria = ' aria-live="assertive"';
364     if (addError || removeError) aria += ' aria-relevant="text';
365     if (addError) aria += ' additions';
366     if (removeError) aria += ' removals';
367     if (addError || removeError) aria += '';
368
369     // build up HTML / no content if no error is found

```

```
370     var html = "\n";
371     if (isError) {
372         html += '<div '+aria+' class="info ui-state-highlight ui-
373             state ui-corner-all">'+"\n";
374         html += '      <p id="ui-error-title">'+"\n";
375         html += '          <span class="ui-icon ui-icon-alert" style="
376             float: left; margin-right: 0.3em;"></span>'+"\n";
377         html += '          '+options.errorsTitle+'\n';
378         html += '      </p>'+"\n";
379         html += '      <ul aria-labelledby="ui-error-title">'+"\n";
380         html += msgs;
381         html += '      </ul>'+"\n";
382         html += '    </div>'+"\n\n\t\t";
383     }
384     // inject error HTML and make onclick event for direct error
385     // correction
386     errorElement = self.element.find("#ui-formular-error");
387     errorElement.html(html);
388
389     // no click events if no error is defined
390     if (isError) {
391         // set link anchor to form
392         errorElement.find("a").click(function(event){
393             event.preventDefault();
394             // get id out of the href anchor
395             var id = $(this).attr("href").split("#");
396             id = id[1];
397             // focus element or first element of a group
398             if(options.forms[id].type == "single") {
399                 var target = options.forms[id].element;
400             } else {
401                 var target = options.forms[id].element[0];
402             }
403             target.focus();
404         });
405         // focus error box when form is submitted
406         if (submitted) errorElement.attr("tabindex",-1).focus();
407         // send data if no errors found
408     } else if(submitted) {
409         self._sendForm();
410     }
411
412     self._updateVirtualBuffer();
413
414     // Callback
415     self._trigger("onShowErrors", 0);
416 },
```

```
// send form
_sendForm: function() {
```

```
417     var options = this.options, self = this;
418
419     switch (options.submitHowTo) {
420         case "post":
421             // prevents revalidating but activates native form event
422             options.disabled = true;
423             // fire native form event
424             self.element.submit();
425             break;
426         case "ajax":
427             $.ajax({ // AJAX Request auslÃ¶sen
428                 data: self.element.serialize(),
429                 type: "post",
430                 url: options.submitUrl,
431                 error: function(msg) {
432                     self._showSuccess(msg);
433                 },
434                 success: function(msg) {
435                     self._showSuccess(msg);
436                 }
437             });
438             break;
439         case "iframe":
440             // save url the form would be submitted
441             options.originalUrl = self.element.attr("action");
442             // change action to ajax server adress
443             self.element.attr("action", options.submitUrl)
444             // inject iframe
445             var frameName = ("upload"+(new Date()).getTime());
446             var uploadFrame = $('<iframe name="'+frameName+'></iframe>');
447             ;
448             uploadFrame.css("display", "none");
449             // when iframe is loaded get content
450             uploadFrame.load(function(data){
451                 self._showSuccess($(this).contents().find("body").html());
452                 // wait till DOM is ready
453                 options.timeout = window.setTimeout(function() {
454                     uploadFrame.remove();
455                 }, 200);
456             });
457             $("body").append(uploadFrame);
458             // submit the form into the iframe
459             self.element.attr("target", frameName);
460             // prevents revalidating but activates native form event
461             options.disabled = true;
462             // fire native form event
463             self.element.submit();
464             break;
465     },
466 }
```

```
466
467 // called when form is submitted
468 _showSuccess: function(value) {
469     var options = this.options, self = this;
470     var msg = "", icon = "";
471     // reenable the widget
472     options.disabled = false;
473
474     // chose icon to show / choose message
475     switch (value) {
476         case "true":
477             msg = options.submitSuccess;
478             icon = "check";
479             break;
480         case "false":
481             msg = options.submitError;
482             icon = "alert";
483             break;
484         default:
485             msg = value;
486             icon = "alert";
487     }
488
489     //build up HTML
490     var html = "\n"
491     html += '      <div id="ui-formular-success">+'\n";
492     html += '          <div aria-live="assertive" class="info ui-state-
493         highlight ui-state ui-corner-all">+'\n";
494     html += '              <p>+'\n";
495     html += '                  <span class="ui-icon ui-icon-'+icon+'" style="
496         float: left; margin-right: 0.3em;"></span>+'\n";
497     html += '                  '+msg+'\n";
498     html += '              </p>+'\n";
499     html += '          </div>+'\n\t\t';
500     html += '      </div>+'\n\t\t';
501     self.element.prepend(html);
502     self.element.find("#ui-formular-success").attr("tabindex",-1).
503         focus();
504     self._updateVitualBuffer();
505     // Callback
506     self._trigger("onShowSuccess", 0);
507 },
508
509 // decides if error is new, old or corrected
510 _whichError: function(error, array) {
511     var value = "";
512     if (!error) {
513         if (array == "new" || array == "old") {
514             value = "old";
515         } else {
```

```
513         value = "new";
514     }
515 } else if (array == "new" || array == "old") {
516     value = "corrected";
517 } else if (array == "corrected") {
518     value = "";
519 }
520 return value;
521 },
522
523 // how many checked / selected options / which value
524 _getValue: function(id) {
525     var options = this.options, self = this;
526     var type = options.forms[id].type;
527     var value = "";
528     switch(type) {
529         case "single":
530             value = options.forms[id].element.val();
531             break;
532         case "group":
533             var result = options.forms[id].element.filter(':checked');
534             value = (result.length) ? result : "";
535             break;
536         case "select":
537             var result = options.forms[id].element.find("option").
538                 filter(':selected');
539             value = (result.length) ? result : "";
540             break;
541     }
542     return value;
543 },
544
545 // make hover and focus effects
546 _makeHover: function(element) {
547     element.bind("mouseenter", function(){ $(this).addClass('ui-
548         state-hover'); })
549     .bind("mouseleave", function(){ $(this).removeClass('ui-state-
550         -hover'); })
551     .bind("focus", function(){ $(this).addClass('ui-state-focus')
552         ; })
553     .bind("blur", function(){ $(this).removeClass('ui-state-focus'
554         ); });
555 },
556
557 // Validators (return true when correct)
558 _regEx: function(elementValue, pattern) {
559     pattern = new RegExp(pattern);
560     return pattern.test(elementValue);
561 },
562 _number: function(elementValue) {
```



```
586 },
587 _equalTo: function(elementValue, ruleValue) {
588   return (elementValue == $("#" + ruleValue).val()) ? true : false;
589 },
590 _captcha: function(elementValue, ruleValue) {
591   return this._trigger("checkCaptcha", null, elementValue);
592 },
593
594 // removes instance and attributes
595 destroy: function() {
596   var options = this.options;
597   // go through every form element
598   $.each(options.forms, function(id){
599     options.forms[id].element
600       .removeClass("ui-state-error")
601       .removeClass("ui-state-hover")
602       .removeAttr("aria-invalid")
603       .removeAttr("aria-required")
604     // remove events
605       .unbind();
606     if (options.forms[id].type == "group") {
607       options.forms[id].element.next()
608         .removeClass("ui-state-error")
609         .removeClass("ui-state-hover")
610         .removeAttr("aria-invalid")
611         .removeAttr("aria-required")
612       // remove events
613         .unbind();
614     }
615   });
616   this.element
617     // remove events
618     .unbind(".formValidator")
619     .unbind("submit")
620     // remove data
621     .removeData('formValidator');
622
623 // set original action url if changed
624 if (options.originalUrl != "") this.element.attr("action",
625   options.originalUrl)
626 // remove injected elements
627 this.element.find("#ui-formular-live, ##ui-formular-error, #ui-
628   formular-success").remove();
629 $("#virtualBufferForm").parent().remove();
630
631 // updates virtual buffer | for older screenreader
632 _updateVirtualBuffer: function() {
633   var form = $("#virtualBufferForm");
634   if (form.length) {
```

```

634     (form.val() == "1") ? form.val("0") : form.val("1")
635 } else {
636     var html = '<form><input id="virtualBufferForm" type="hidden"
637         value="1" /></form>';
638     $("body").append(html);
639 }
640 });
641
642 $.extend($.ui.formValidator, {
643     version: "1.7.1",
644     defaults: {
645         validateLive: true,
646         validateTimeout: 500,
647         validateOff: "Bitte klicken Sie hier um die Live Validierung zu
648             deaktivieren.",
649         validateOn: "Bitte klicken Sie hier um die Live Validierung zu
650             aktivieren.",
651         errorsTitle: "Bitte korrigieren Sie folgende Fehler:",
652         submitHowTo: "ajax",
653         submitUrl: "",
654         submitError: "Bei der Datenübertragung ist ein Fehler
655             aufgetreten. Entschuldigen Sie bitte und versuchen Sie es noch
656             einmal.",
657         submitSuccess: "Die Daten wurden erfolgreich übermittelt. Vielen
658             Dank!",
659         //do not alter these vars
660         errorsArray: [],
661         originalUrl: ""
662     }
663 });
664 });
665
666 })(jQuery);

```

Listing B.2: formValidator: JavaScript Quellcode

B.2 Lightbox

B.2.1 HTML Quellcode

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/
   TR/xhtml11/DTD/xhtml11.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
3 <head>
4   <meta http-equiv="content-type" content="text/html; charset=charset
      =ISO-8859-1" />
5   <meta name="language" content="DE, AT, CH" scheme="DCTERMS.RFC3066"
      />

```

```
6  <meta name="creator" content="Felix Nagel, http://www.felixnagel.
    com for Namics (Deutschland) GmbH, http://www.namics.com" />
7  <title>ui.ariaLightbox – jQuery UI</title>
8  <link type="text/css" href="css/ui-lightness/jquery-ui-1.7.1.custom.
    .css" rel="stylesheet" />
9   <link type="text/css" href="css/style.css" rel="stylesheet" />
10 </head>
11 <body>
12 <h1>jQuery UI – ui.ariaLightbox</h1>
13 <div id="wrapper">
14
15   <div class="box ui-widget-content ui-corner-all ui-helper-clearfix">
16     <h2 class="ui-dialog-titlebar ui-widget-header ui-corner-all">
        Einzelbilder</h2>
17     <a class="singleLightbox" href="pics/DirtyFlowers.jpg"></a>
18     <a class="singleLightbox" href="pics/RealKeepers.jpg"></a>
19     <a class="singleLightbox" href="pics/WhiteLilium.jpg"></a>
20     <a class="singleLightbox" href="pics/TheJerks.jpg"></a>
21     <a class="singleLightbox last" href="pics/LaRevolta.jpg"><img src
        ="pics/thumb/LaRevolta.jpg" style="height: 7.5em; width: 5em;
        " alt="La Revolta" title="Die Band La Revolta beim Emergenza
        Semi-Finale in der Batschkapp Frankfurt am Main am 24.04.08" /
        ></a>
22     <div class="sub">
23       <a href="#nogo" id="closelink1">close()</a>|
24       <a href="#nogo" id="destroylink1">destroy()</a>|
25       <a href="#nogo" id="disablelink1">disable()</a>
26     </div>
27   </div>
28   <div class="box ui-widget-content ui-corner-all ui-helper-clearfix
    ">
29     <h2 class="ui-dialog-titlebar ui-widget-header ui-corner-all">
        Bilderserie</h2>
30     <ul>
31       <li>
```

```
32 <a class="seriesLightbox" href="pics/TheBossHoss_1.jpg"></a>  
36 </li>  
37 <li>  
38   <a class="seriesLightbox" href="pics/TheBossHoss_2.jpg"></a>  
42   </li>  
43 </ul>  
44 <div class="sub">  
45   <a href="#nogo" id="gallerylink2">startGallery</a>|  
46   <a href="#nogo" id="closelink2">close ()</a>|  
47   <a href="#nogo" id="destroylink2">destroy ()</a>|  
48   <a href="#nogo" id="disablelink2">disable ()</a>|  
49   <a href="#nogo" id="prevlink2">prev ()</a>|  
50   <a href="#nogo" id="nextlink2">next ()</a>  
51 </div>  
52 <br /><br />  
53 <br /><br />  
54 <br /><br />  
55 <br /><br />  
56 <br /><br />  
57 </div>  
58 <div id="copyright">  
59 <a rel="license" href="http://creativecommons.org/licenses/by-sa/3.0/  
60   "></a><br />  
61 Diese(s) Werk bzw. Inhalt von <span xmlns:cc="http://creativecommons.org/ns#" property="cc:attributionName">Felix Nagel  
62 f&#252;r Namics (Deutschland) GmbH</span> steht unter einer <a  
63 rel="license" href="http://creativecommons.org/licenses/by-sa/3.0/  
64   ">Creative Commons Namensnennung-Weitergabe unter gleichen  
65 Bedingungen 3.0 Unported Lizenz</a>. <br />&#220;ber diese Lizenz  
66 hinausgehende Erlaubnisse erhalten Sie m&#246;glicherweise unter <  
67 a xmlns:cc="http://creativecommons.org/ns#" href="http://www.  
68   felixnagel.com" rel="cc:morePermissions">http://www.felixnagel.com  
69   </a>. <br />
```

```
61 </div>
62 <script type="text/javascript" src="js/jquery.js"></script>
63 <script type="text/javascript" src="js/ui.core.js"></script>
64 <script type="text/javascript" src="js/ui.ariaLightbox.js"></script
65 >
66 <script type="text/javascript">
67 $(function() {
68     var lightbox1 = $(".singleLightbox").ariaLightbox({
69         altText: "alt",
70         descText: "title",
71         useDimmer: false,
72         title: "Großansicht",
73         pos: "offset"
74     });
75     $("#closelink1").click(function (event) {
76         lightbox1.ariaLightbox('close');
77     });
78     $("#destroylink1").click(function (event) {
79         lightbox1.ariaLightbox('destroy');
80     });
81     $("#disablelink1").click(function (event) {
82         lightbox1.ariaLightbox('disable');
83     });
84
85     var lightbox2 = $(".seriesLightbox").ariaLightbox({
86         imageArray: [],
87         altText: "alt",
88         descText: "title",
89         useDimmer: true,
90         pos: "auto",
91         title: "Bildergallerie",
92         em: false
93     });
94     $("#destroylink2").click(function (event) {
95         lightbox2.ariaLightbox('destroy');
96     });
97     $("#disablelink2").click(function (event) {
98         lightbox2.ariaLightbox('disable');
99     });
100    $("#nextlink2").click(function (event) {
101        lightbox2.ariaLightbox('next');
102    });
103    $("#prevlink2").click(function (event) {
104        lightbox2.ariaLightbox('prev');
105    });
106    $("#gallerylink2").click(function (event) {
107        lightbox2.ariaLightbox('startGallery', event);
108    });
109
```

```

110      });
111
112
113
114  </script>
115 </body>
116 </html>
```

Listing B.3: lightbox: HTML Quellcode

B.2.2 JavaScript Quellcode

```

1  /*
2   * jQuery UI ariaLightbox 1.0
3   *
4   * Copyright (c) 2009 Felix Nagel for Namics (Deutschland) GmbH
5   * Licensed under Creative Commons Attribution-Share Alike 3.0
6   * Unported (http://creativecommons.org/licenses/by-sa/3.0/)
7   *
8   * Depends: ui.core.js
9
9  USAGE ::::::::::::::
10 * Take a look in the html file or the (german) pdf file delivered
11   with this example
12 * The widget gets all the elements in the document which matches
13   choosen selector
14 * There are two modes: singleview and gallerview defined with
15   imageArray: []
16
17 * Options
18 imageArray:      activates galleryview of set to imageArray: []
19 altText:         which attr (within the image) as alt attr
20 descText:        which attr (within the image) as description text
21 prevText:        text on the button
22 nextText:        see above
23 titleText:       titleText of the lightbox
24 pictureText:    string: picture
25 ofText:          string: of
26 closeText:       string: close element
27 pos:             position of the lightbox, possible values: auto, offset,
28   or [x,y]
29 autoHeight:     margin to top when pos: auto is used
30 offsetX:        number: if pos: "offset" its the distance between
31   lightbox and mousclick position
32 offsetY:        see above
33 useDimmer:      boolean, activate or deactivate dimmer
34 animationSpeed: in milliseconds or jQuery keywords aka "slow", "
35   fast"
36 zIndex:          number: z-index for overlay elements
37 background:      color in HTML notation
```

```
32 opacity:      decimal between 1–0
33 em:           multiplicator for relative width (em) calculation,
34   normally not to be edited
35
36 * Callbacks
37 onShow
38 onChangePicture
39 onClose
40 onPrev
41 onNext
42 * public Methods
43 startGallery    parameter: event
44 close
45 next
46 prev
47 disable
48 enable
49 disable
50 destroy
51
52 */
53 (function($) {
54
55 $.widget("ui.ariaLightbox", {
56
57   _init: function() {
58     var options = this.options, self = this;
59
60     // set trigger events / gallery mode
61     if (options.imageArray) {
62       // save all elements
63       options.imageArray[options.imageArray.length] = this.element;
64       var index = options.imageArray.length;
65
66       this.element.click(function (event) {
67         // set active element
68         self.options.activeImage = index - 1;
69         // only activate when widget isn't disabled
70         if (!options.disabled) {
71           event.preventDefault();
72           self._open($(this), event);
73         }
74       });
75     // single image mode
76   } else {
77     self.element.click(function (event) {
78       // only activate when widget isn't disabled
79       if (!options.disabled) {
80         event.preventDefault();
```

```
81         self._open($(this), event);
82     }
83   });
84 }
85 // only set resize event when lightbox is activated
86 if (options.useDimmer)
87 $(window).resize(function(){
88   self._dimmerResize();
89 });
90
91 },
92
93 // call gallery from link
94 startGallery: function (event){
95   var options = this.options, self = this;
96   self._open($(options.imageArray[0]), event);
97 },
98
99 // check if lightbox is already opened
100 _open: function (element, event){
101   var options = this.options, self = this;
102   // save clicked element
103   options.clickedElement = event.target;
104
105   // if wrapper element isn't found, create it
106   options.wrapperElement = $("#ui-lightbox-wrapper");
107   if(!options.wrapperElement.length){
108     self._show(element, event);
109   } else {
110     self._changePicture(element, event);
111   }
112 },
113
114 // called when lightbox wrapper element is not injected yet
115 _show: function (element, event){
116   var options = this.options, self = this;
117
118   // build html
119   var html = "\n";
120   html += '<div id="ui-lightbox-wrapper" style="z-index:' + options.zIndex + '+;" class="ui-dialog ui-widget ui-widget-content ui-corner-all" tabindex="-1" role="dialog" aria-labelledby="ui-dialog-title-dialog">' + "\n";
121   html += ' <div class="ui-dialog-titlebar ui-widget-header ui-corner-all ui-helper-clearfix">' + "\n";
122   html += '   <span class="ui-dialog-title" id="ui-dialog-title-dialog">' + options.titleText + '</span>' + "\n";
123   html += '   <a href="#nogo" id="ui-lightbox-close" class="ui-dialog-titlebar-close ui-corner-all" title="' + options.closeText + '" role="button">' + "\n";
```

```
124     html += '      <span class="ui-icon ui-icon-closethick">' + options
125     .closeText + '</span>' + "\n";
126     html += '    </a>' + "\n";
127     html += '  </div>' + "\n";
128     html += '  <div id="ui-lightbox-content">' + "\n";
129     html += '    <div id="ui-lightbox-image"><img src="" aria-
130       describedby="ui-lightbox-description" /></div>' + "\n";
131     html += '    <p id="ui-lightbox-description"></p>' + "\n";
132     // show pager and rage description if its an array of images
133     if (options.imageArray) {
134       html += '      <p id="ui-lightbox-pager"></p>' + "\n";
135       html += '      <div id="ui-dialog-buttonpane" class="ui-dialog-
136         buttonpane ui-widget-content ui-helper-clearfix">' + "\n";
137       html += '        <button id="ui-lightbox-next" type="button" class="
138           ui-state-default ui-corner-all">nächstes Bild</button>' + "\n";
139       html += '        <button id="ui-lightbox-prev" type="button" class="
140           ui-state-default ui-corner-all">vorheriges Bild</button>' + "\n"
141       ;
142       html += '      </div>' + "\n";
143     }
144     html += '    </div>' + "\n";
145     html += '  </div>' + "\n";
146
147     // create dimmer
148     if (options.useDimmer) self._lightboxCreate();
149
150     // inject HTML
151     $("body").append(html);
152
153     // Call back
154     self._trigger("onShow", 0);
155     options.wrapperElement = $("#ui-lightbox-wrapper");
156
157     // enable keyboard navigation
158     if(options.imageArray) {
159       options.wrapperElement.keydown( function(event){
160         if( event.keyCode == $.ui.keyCode.RIGHT) self.next();
161         if( event.keyCode == $.ui.keyCode.DOWN) self.next();
162         if( event.keyCode == $.ui.keyCode.UP) self.prev();
163         if( event.keyCode == $.ui.keyCode.LEFT) self.prev();
164         if( event.keyCode == $.ui.keyCode.SPACE) self.next();
165         if( event.keyCode == $.ui.keyCode.END) {
166           options.activelimage = options.imageArray.length -2;
167           event.preventDefault();
168           self.next();
169         }
170         if( event.keyCode == $.ui.keyCode.HOME) {
171           options.activelimage = 1;
172           event.preventDefault();
173           self.prev();
174         }
175       });
176     }
177   }
178 }
```

```
168         }
169     });
170     options.buttonpane = options.wrapperElement.find("#ui-dialog-
171         buttonpane")
172     // check button state
173     self._setButtonState();
174
175     // add events for paging and hover if necessary
176     var prev = options.buttonpane.find("#ui-lightbox-prev");
177     prev.click( function() { self.prev(); });
178     self._makeHover(prev);
179     var next = options.buttonpane.find("#ui-lightbox-next");
180     next.click( function() { self.next(); });
181     self._makeHover(next);
182 }
183 options.wrapperElement.keydown( function(event){
184     if(event.keyCode == $.ui.keyCode.ESCAPE) self.close();
185 });
186
187 // add hover and close event
188 var closeElement = options.wrapperElement.find("#ui-lightbox-
189         close");
190 closeElement.click( function() { self.close(); });
191 self._makeHover(closeElement);
192
193 // decide which position is set
194 switch (options.pos) {
195     case "auto":
196         var viewPos = self._pageScroll();
197         var posLeft = ((document.width() - options.
198             wrapperElement.width()) / 2);
199         var posTop = viewPos[1] + options.autoHeight;
200         break;
201     case "offset":
202         var posLeft = event.pageX + options.offsetX;
203         var posTop = event.pageY - options.offsetY;
204         break;
205     default:
206         var position = options.pos.split(",");
207         var posLeft = position[0];
208         var posTop = position[1];
209         break;
210     }
211
212     // set initial position, fade in and focus
213     options.wrapperElement
214         .css({
215             left: posLeft+"px",
216             top: posTop+"px"
217         })
218         .fadeln(options.animationSpeed)
```

```
215     .focus();
216
217     // set picture and meta data
218     self._changePicture(element, event);
219 },
220
221 // called whenever a picture should be changed
222 _changePicture: function (element, event){
223     var options = this.options, self = this;
224     // get elements for reuse
225     var contentWrapper = options.wrapperElement.find("#ui-lightbox-content");
226     var imageWrapper = contentWrapper.find("#ui-lightbox-image");
227     var imageElement = imageWrapper.find("img");
228
229     // fade out the picture, then set new properties and animate
230     // elements
231     imageElement.fadeOut(options.animationSpeed, function() {
232         // ARIA / set attributes and begin manipulations
233         contentWrapper
234             .attr("aria-live", "assertive")
235             .attr("aria-relevant", "additions removals text")
236             .attr("aria-busy", true);
237
238         // preload image
239         var image = new Image();
240         image.onload = function() {
241             // set new picture properties
242             imageElement
243                 .attr('src', element.attr("href"))
244                 .attr('alt', element.find("img").attr(options.altText));
245
246             // if em isn't deactivated calculate relative size
247             var calculatedX = (options.em) ? image.width*options.em+"em"
248                 : image.width;
249             var calculatedY = (options.em) ? image.height*options.em+"em"
250                 : image.height;
251             // set image dimension / set always cause of display problems
252             // with relative dimension setting
253             imageElement.css({
254                 width: calculatedX,
255                 height: calculatedY
256             });
257             // decide which position is set and animate the width of the
258             // lightbox element
259             switch (options.pos) {
260                 case "offset":
261                     options.wrapperElement.animate({
262                         left: event.pageX+options.offsetX+"px",
263                         top: event.pageY+options.offsetY+"px",
264                     },
265                     options.animationSpeed);
266             }
267         }
268     });
269 }
```

```
259         width: calculatedX
260     }, options.animationSpeed);
261     break;
262   case "auto":
263     options.wrapperElement.animate({
264       left: ($(document).width() - image.width)/2)+"px",
265       width: calculatedX
266     }, options.animationSpeed);
267     break;
268   }
269 // resize the hight of the image wrapper to resize the
270 // lightbox element / wait till finished
271 imageWrapper.animate({
272   height: calculatedY
273 },
274 options.animationSpeed,
275 function () {
276   // fade in the picture
277   imageElement.fadeIn(options.animationSpeed);
278   // change description of the picture
279   options.wrapperElement.find("#ui-lightbox-description")
280     .text(element.find("img").attr(options.descText));
281   // if it is a gallery change the pager text
282   if (options.imageArray)
283     options.wrapperElement.find("#ui-lightbox-pager")
284       .text(options.pictureText + ' '+ (options.activeImage+1)
285           +' '+ options.ofText + ' '+ options.imageArray.
286             length);
287   // check if lightbox popup changed body dimension
288   if (options.useDimmer) self._dimmerResize();
289   // update screenreader buffer
290   self._updateVirtualBuffer();
291   // ARIA / manipulations finished
292   contentWrapper.attr("aria-busy", false);
293   // Callback
294   self._trigger("onChangePicture", 0);
295   // END of image changing
296 }
297 );
298 // IE specific, prevent animate gif failures / unload onload
299 image.onload = function(){};
300 };
301 }
302 // set button attributes
303 _setButtonState: function (){
304   var options = this.options;
```

```
306    // activate both buttons
307    options.buttonpane.find("#ui-lightbox-next, #ui-lightbox-prev")
308        .removeAttr("disabled")
309        .removeClass("ui-state-disabled")
310        .removeClass("ui-state-focus");
311    switch (options.activelimage) {
312        // disable prev
313        case 0:
314            options.buttonpane.find("#ui-lightbox-prev")
315                .attr("disabled", "disabled")
316                .removeClass("ui-state-hover")
317                .addClass("ui-state-disabled");
318            options.buttonpane.find("#ui-lightbox-next").focus();
319            break;
320        // disable next
321        case options.imageArray.length -1:
322            options.buttonpane.find("#ui-lightbox-next")
323                .attr("disabled", "disabled")
324                .removeClass("ui-state-hover")
325                .addClass("ui-state-disabled");
326            options.buttonpane.find("#ui-lightbox-prev").focus();
327            break;
328    }
329},
330
331 // close wrappper element
332 close: function () {
333     var options = this.options, self = this;
334     // focus back to first clicked element
335     $(options.clickedElement).parent().focus();
336     options.wrapperElement.fadeOut(options.animationSpeed, function
337         () {
338             $(this).remove();
339         });
340     // remove dimmer
341     if (options.useDimmer) $("#ui-lightbox-screendimmer").fadeOut(
342         options.animationSpeed, function () { $(this).remove(); });
343     // Callback
344     self._trigger("onClose", 0);
345 },
346
347 // change picture to previous image
348 prev: function () {
349     var options = this.options, self = this;
350     if(options.imageArray && options.activelimage > 0) {
351         options.activelimage = options.activelimage -1;
352         self._changePicture($(options.imageArray[options.activelimage]))
353             ;
354         self._setButtonState();
355         // Callback
356     }
357 }
```

```
353         self._trigger("onPrev", 0);
354     }
355 },
356
357 // change picture to next image
358 next: function () {
359     var options = this.options, self = this;
360     if(options.imageArray && options.activelimage < (options.
361         imageArray.length -1)) {
362         options.activelimage = options.activelimage +1;
363         self._changePicture($(options.imageArray[options.activelimage]))
364             ;
365         self._setButtonState();
366         // Callback
367         self._trigger("onNext", 0);
368     }
369 }
370
371 // create lightbox
372 _lightboxCreate: function () {
373     var options = this.options, self = this;
374     // inject html
375     var html = '<div id="ui-lightbox-screendimmer" style="display:
376         none;"></div>';
377     $("body").append(html);
378     // set attributes
379     $("#ui-lightbox-screendimmer")
380         .css({
381             width: self._dimmerWidth(),
382             height: self._dimmerHeight(),
383             zIndex: options.zIndex,
384             background: options.background,
385             position: "absolute",
386             top: "0px",
387             left: "0px",
388             opacity: options.opacity
389         })
390         .fadeIn(options.animationSpeed)
391         // if dimmer is clicked, close lightbox
392         .click( function () {
393             self.close();
394         });
395 }
396
397 // resize dimmer
398 _dimmerResize: function () {
399     var self = this;
400     var dimmer = $("#ui-lightbox-screendimmer");
401     // make dimmer div small / necessary to check if content is
402     // smaller than the dimmer div
```

```
399     dimmer.css({
400         width: 0,
401         height: 0
402     });
403     // check real body dimension
404     var dimension = self._pageScroll();
405     // if page is not scrolled without dimmer div use normal width
406     var dimensionX = (dimension[0] == 0) ? self._dimmerWidth() :
407         dimension[0];
408     dimmer.css({
409         width: dimensionX,
410         height: self._dimmerHeight()
411     });
412
413     // get body height
414     _dimmerHeight: function() {
415         // handle IE 6
416         if ($.browser.msie && $.browser.version < 7) {
417             var scrollHeight = Math.max(
418                 document.documentElement.scrollHeight,
419                 document.body.scrollHeight
420             );
421             var offsetHeight = Math.max(
422                 document.documentElement.offsetHeight,
423                 document.body.offsetHeight
424             );
425             if (scrollHeight < offsetHeight) {
426                 return $(window).height() + 'px';
427             } else {
428                 return scrollHeight + 'px';
429             }
430         // handle "good" browsers
431     } else {
432         return $(document).height() + 'px';
433     }
434 },
435
436     // get body width
437     _dimmerWidth: function() {
438         // handle IE 6
439         if ($.browser.msie && $.browser.version < 7) {
440             var scrollWidth = Math.max(
441                 document.documentElement.scrollWidth,
442                 document.body.scrollWidth
443             );
444             var offsetWidth = Math.max(
445                 document.documentElement.offsetWidth,
446                 document.body.offsetWidth
447             );

```

```
448     if (scrollWidth < offsetWidth) {
449         return $(window).width() + 'px';
450     } else {
451         return scrollWidth + 'px';
452     }
453     // handle "good" browsers
454 } else {
455     return $(document).width() + 'px';
456 }
457 },
458
459 // get scrolling of the page
460 _pageScroll: function() {
461     var xScroll, yScroll;
462     if (self.pageYOffset) {
463         yScroll = self.pageYOffset;
464         xScroll = self.pageXOffset;
465         // Explorer 6 Strict
466     } else if (document.documentElement && document.documentElement.
467         scrollTop) {
468         yScroll = document.documentElement.scrollTop;
469         xScroll = document.documentElement.scrollLeft;
470         // all other Explorers
471     } else if (document.body) {
472         yScroll = document.body.scrollTop;
473         xScroll = document.body.scrollLeft;
474     }
475     arrayPageScroll = new Array(xScroll, yScroll);
476     return arrayPageScroll;
477 },
478
479 // make hover and focus effects
480 _makeHover: function(element) {
481     element.bind("mouseenter", function(){ $(this).addClass('ui-
482         state-hover'); })
483     .bind("mouseleave", function(){ $(this).removeClass('ui-state-
484         -hover'); })
485     .bind("focus", function(){ $(this).addClass('ui-state-focus')
486         ; })
487     .bind("blur", function(){ $(this).removeClass('ui-state-focus'
488         ); });
489 },
490
491 // updates virtual buffer of older screenreader
492 _updateVirtualBuffer: function() {
493     var form = $("#virtualBufferForm");
494     if(form.length) {
495         (form.val() == "1") ? form.val("0") : form.val("1")
496     } else {
```

```
492     var html = '<form><input id="virtualBufferForm" type="hidden">'  
493         + 'value="1" /></form>';  
494     $("body").append(html);  
495 }  
496 },  
497 destroy: function() {  
498     var options = this.options;  
499  
500     this.element  
501         // remove events  
502         .unbind(".ariaLightbox")  
503         .unbind("click")  
504         // remove data  
505         .removeData('ariaLightbox');  
506  
507     $("#virtualBufferForm").parent().remove();  
508     $("#ui-lightbox-screendimmer").remove();  
509     $("#ui-lightbox-wrapper").unbind("keydown").remove();  
510 }  
511});  
512  
513 $.extend($.ui.ariaLightbox, {  
514     version: "1.7.1",  
515     defaults: {  
516         altText: "alt",  
517         descText: "titleText",  
518         prevText: "vorheriges Bild",  
519         nextText: "nächstes Bild",  
520         titleText: "Lightbox",  
521         pictureText: "Bild",  
522         ofText: "von",  
523         closeText: "Schließen [ESC]",  
524         pos: "auto",  
525         autoHeight: 50,  
526         offsetX: 10,  
527         offsetY: 10,  
528         // config screen dimmer  
529         useDimmer: true,  
530         animationSpeed: "slow",  
531         zIndex: 1000,  
532         background: "black",  
533         opacity: 0.8,  
534         em: 0.0568182,  
535         // don not alter this var  
536         activeImage: 0  
537     }  
538 });
```

539 })(jQuery);

Listing B.4: lightbox: JavaScript Quellcode

B.3 Tabs

B.3.1 JavaScript Quellcode

```

1  /*
2   * jQuery UI ariaTabs 1.0
3   *
4   * Copyright (c) 2009 Felix Nagel for Namics (Deutschland) GmbH
5   * Licensed under Creative Commons Attribution-Share Alike 3.0
6   * Unported (http://creativecommons.org/licenses/by-sa/3.0/)
7   *
8   * Depends: ui.core.js
9   *          ui.tabs.js
10  *          ui.sortable.js (optional, see tabs docu)
11
11 USAGE:::::::::::
12 * Take a look in the html file or the (german) pdf file delivered
13 * with this example
14 * Simply add the js file uner the regular ui.tabs.js script tag
15 * Supports all options, methods and callbacks of the original widget
16 * sortable tabs are accessable but the sortable functionality as it
17 *      is provided by the ui.sortable widget doesnt support ARIA
18 */
18 (function($) {
19     $.fn.extend($.ui.tabs.prototype ,{
20
21     _original_init: $.ui.tabs.prototype._init ,
22     // when widget is initiated
23     _init: function() {
24         var self = this, options = this.options;
25         // fire original function
26         self._original_init();
27         // ARIA
28         self.element.attr("role", "application");
29         self.list.attr("role", "tablist");
30         // init aria attributes for each panel and anchor
31         for (var x = 0; x < self.anchors.length; x++) {
32             self._ariaInit(x);
33         }
34
35         // keyboard
36         self.element.keydown( function(event){
37             switch (event.keyCode) {
38                 case $.ui.keyCode.RIGHT:

```

```
39     event.preventDefault();
40     self.select(options.selected+1);
41     break;
42   case $.ui.keyCode.DOWN:
43     event.preventDefault();
44     self.select(options.selected+1);
45     break;
46   case $.ui.keyCode.UP:
47     event.preventDefault();
48     self.select(options.selected-1);
49     break;
50   case $.ui.keyCode.LEFT:
51     event.preventDefault();
52     self.select(options.selected-1);
53     break;
54   case $.ui.keyCode.END:
55     event.preventDefault();
56     self.select(self.anchors.length-1);
57     break;
58   case $.ui.keyCode.HOME:
59     event.preventDefault();
60     self.select(0);
61     break;
62   }
63 });
64 },
65
66 _original_load: $.ui.tabs.prototype.load,
67 // called whenever tab is called but if option collapsible is set
68 // | fired once at init for the chosen tab
69 load: function(index) {
70   // hide all unselected
71   for (var x = 0; x < this.anchors.length; x++) {
72     // anchors
73     this._ariaSet(x, false);
74     // remove ARIA live settings
75     if ($.data(this.anchors[x], 'href.tabs')) {
76       $(this.panels[x])
77         .removeAttr("aria-live")
78         .removeAttr("aria-busy");
79     };
80     // is remote? set ARIA states
81     if ($.data(this.anchors[index], 'href.tabs')) {
82       $(this.panels[index])
83         .attr("aria-live", "polite")
84         .attr("aria-busy", "true");
85     }
86     // fire original function
87     this._original_load(index);
```

```
88     // is remote? end ARIA busy
89     if($.data(this.anchors[index], 'href.tabs')) {
90         $(this.panels[index])
91             .attr("aria-busy", "false");
92     }
93     // set state for the activated tab
94     this._ariaSet(index, true);
95 },
96
97 // sets aria states for single tab and its panel
98 _ariaSet: function(index, state) {
99     var tabindex = (state) ? 0 : -1;
100    // set ARIA state for loaded tab
101    $(this.anchors[index])
102        .attr("tabindex", tabindex)
103        .attr("aria-selected", state)
104    // set ARIA state for loaded tab
105    $(this.panels[index])
106        .attr("aria-hidden", !state)
107        .attr("aria-expanded", state);
108 },
109
110 // sets all attributes when plugin is called or if tab is added
111 _arialnit: function(index) {
112     var self = this;
113     // get widget generated ID of the panel
114     var panelId = $(this.panels[index]).attr("id");
115     // ARIA anchors and li's
116     $(this.anchors[index])
117         .attr("role", "tab")
118         .attr("aria-controls", panelId)
119         .attr("id", panelId+"-tab")
120     // set li to presentation role
121     .parent().attr("role", "presentation");
122     // ARIA panels aka content wrapper
123     $(this.panels[index])
124         .attr("role", "tabpanel")
125         .attr("aria-labelledby", panelId+"-tab");
126     // if collapsible, set event to toggle ARIA state
127     if (this.options.collapsible) {
128         $(this.anchors[index]).bind(this.options.event, function(
129             event) {
130                 // get class to negate it to set states correctly when
131                 // panel is collapsed
132                 self._ariaSet(index, !$(self.panels[index]).hasClass("ui-
133                     tabs-hide"));
134             });
135     }
136 },
```

```
135     _original_add: $.ui.tabs.prototype.add,
136     // called when a tab is added
137     add: function(url, label, index) {
138         // fire original function
139         this._original_add(url, label, index);
140         // ARIA
141         this.element
142             .attr("aria-live", "polite")
143             .attr("aria-relevant", "additions");
144
145         // if no index is defined tab should be added at the end of the
146         // tab list
147         if (index) {
148             this._ariaInit(index);
149             this._ariaSet(index, false);
150         } else {
151             this._ariaInit(this.anchors.length - 1);
152             this._ariaSet(this.anchors.length - 1, false);
153         }
154     },
155
156     _original_remove: $.ui.tabs.prototype.remove,
157     // called when a tab is removed
158     remove: function(index) {
159         // fire original function
160         this._original_remove(index);
161         // ARIA
162         this.element
163             .attr("aria-live", "polite")
164             .attr("aria-relevant", "removals");
165     },
166
167     _original_destroy: $.ui.tabs.prototype.destroy,
168     // removes all the setted attributes
169     destroy: function() {
170         var self = this, options = this.options;
171         // fire original function
172         this._original_destroy();
173         // remove ARIA attribute
174         // wrapper element
175         self.element
176             .removeAttr("role")
177             .removeAttr("aria-live")
178             .removeAttr("aria-relevant");
179         // ul element
180         self.listremoveAttr("role");
181         for (var x = 0; x < self.anchors.length; x++) {
182             // tabs
183             $(self.anchors[x])
184                 .removeAttr("aria-selected")
```

```

184     .removeAttr("aria-controls")
185     .removeAttr("role")
186     .removeAttr("id")
187     .removeAttr("tabindex")
188     // remove presentation role of the li element
189     .parent().removeAttr("role");
190     // tab panels
191     $(self.panels[x])
192         .removeAttr("aria-hidden")
193         .removeAttr("aria-expanded")
194         .removeAttr("aria-labelledby")
195         .removeAttr("aria-live")
196         .removeAttr("aria-busy")
197         .removeAttr("aria-relevant")
198         .removeAttr("role");
199     }
200   }
201 });
202 })(jQuery);

```

Listing B.5: ariaTabs: JavaScript Quellcode

B.4 Sortierbare Tabellen

B.4.1 HTML Quellcode

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/
   TR/xhtml11/DTD/xhtml11.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
3 <head>
4   <meta http-equiv="content-type" content="text/html; charset=ISO
      -8859-1" />
5   <meta name="language" content="DE, AT, CH" scheme="DCTERMS.RFC3066"
      />
6   <meta name="creator" content="Felix Nagel, http://www.felixnagel.
      com for Namics (Deutschland) GmbH, http://www.namics.com" />
7   <title>ui.ariaSortTable – jQuery UI</title>
8   <link type="text/css" href="css/ui-lightness/jquery-ui-1.7.1.custom
      .css" rel="stylesheet" />
9   <link type="text/css" href="css/style.css" rel="stylesheet" />
10  <!--[if gte IE 6]>
11    <style type="text/css">@import url(css/style_ie.css);</style>
12  <![endif]-->
13  <script type="text/javascript" src="js/jquery.js"></script>
14  <script type="text/javascript" src="js/ui.core.js"></script>
15  <script type="text/javascript" src="js/ui.ariaSortTable.js"></script
      >
16  <script type="text/javascript" src="js/demo.js"></script>
17  <script type="text/javascript">

```

```

18      $(function() {
19          var table1 = $("table").ariaSortTable({
20              rowsToShow: 10,
21              pager: true,
22              onInit: function() {
23                  demoControl();
24              },
25              colsToHide: {
26                  1: true,
27                  7: true
28              }
29          });
30
31      $("#destroy").click(function (event) {
32          table1.ariaSortTable('destroy');
33      });
34      $("#disable").click(function (event) {
35          table1.ariaSortTable('disable');
36      });
37  });
38  </script>
39 </head>
40 <body>
41 <h1>jQuery UI – ui.ariaSortTable</h1>
42 <div id="controls">
43     <a href="performance.html">Performance Demo</a> / <a href="#" id="#destroy">destroy</a> / <a href="#" id="#disable">disable</a>
44 </div>
45 <div id="wrapper">
46     <div id="table-wrapper">
47         <table summary="This table holds sample data to test sorting and
48             paging. Please feel free to sort columns by clicking their
49             headers and see all data by clicking at the pager.">
50             <caption class="ui-state-highlight ui-corner-all">Functional
51                 demo table</caption>
52             <thead>
53                 <tr>
54                     <th class="ui-table-asc ui-state-active ui-table-number"
55                         style="min-width: 4.2em;"><a href="#server_site_order">
56                         UID</a></th>
57                     <th class="ui-table-number" style="min-width: 10em;"><a
58                         href="#server_site_order">Number</a></th>
59                     <th class="ui-table-number-de" style="min-width: 10em;">
60                         Decimal DE</th>
61                     <th class="ui-table-number" style="min-width: 6em;"><a href
62                         ="#server_site_order">Decimal</a></th>
63                     <th class="ui-table-date-de" style="min-width: 8em;"><a
64                         href="#server_site_order">Date DE</a></th>
65                     <th class="ui-table-date-iso ui-table-deactivate" style="
66                         min-width: 8em;"><a href="#server_site_order">Date ISO</

```

```

      a></th>
57   <th class="ui-table-text" style="min-width: 10em;"><a href=
      "#server_site_order">String</a></th>
58   <th class="ui-table-deactivate" style="min-width: 12em;"><a
      href="#server_site_order_only">false</a></th>
59   </tr>
60   </thead>
61   <tbody>
62     <tr>
63       <td>1</td>
64       <td>111112</td>
65       <td>119,111</td>
66       <td>111.231</td>
67       <td>01.01.2009</td>
68       <td>2009-01-01</td>
69       <td>abcdefghijklm</td>
70       <td>Lorum ipsum</td>
71     </tr>
72     <tr>
73       <td>2</td>
74       <td>131112</td>
75       <td>119,131</td>
76       <td>311.231</td>
77       <td>01.03.2009</td>
78       <td>2009-03-01</td>
79       <td><a href="#test">test link</a></td>
80       <td>Sorum ipsum</td>
81     </tr>
82     <tr>
83       <td>3</td>
84       <td>13144562</td>
85       <td>11459,131</td>
86       <td>31.231</td>
87       <td>02.03.09</td>
88       <td>2009-03-02</td>
89       <td>rwzkgjgjk</td>
90       <td>Sdrum ipsum</td>
91     </tr>
92     <tr>
93       <td>4</td>
94       <td>17662</td>
95       <td>11459,131</td>

```

Listing B.6: sorTable: HTML Quellcode

Aufgrund der Größe der Tabelle (und der geringen Bedeutung dieser der Demonstration dienenden Daten) soll auf eine komplette Darstellung an dieser Stelle verzichtet werden.

B.4.2 JavaScript Quellcode

1 /*

```
2 * jQuery UI ariaSortTable 1.0 (08.08.09)
3 *
4 * Copyright (c) 2009 Felix Nagel for Namics (Deutschland) GmbH
5 * Licensed under Creative Commons Attribution-Share Alike 3.0
       Unported (http://creativecommons.org/licenses/by-sa/3.0/)
6 *
7 * Depends: ui.core.js
8
9 USAGE ::::::::::::::
10 * Take a look in the html file or the (german) pdf file delivered
      with this example
11 * To set sorting method add css classes, default is text,
      alphabetically
12
13 * Sorting CSS classes (apply to th elements)
14 ui-table-number      123 or 123.456
15 ui-table-number-de   123,456
16 ui-table-date        07/28/2009
17 ui-table-date-de     28.07.2009
18 ui-table-date-iso    2009-07-28
19 ui-table-deactivate  deactivates sorting for this col
20 ui-state-active      class to set a col as pre sorted (server site)
21
22 * Options
23 rowToStart           row to start, begins with 1
24 rowsToShow           How many rows to show? If not set, widget will show
      all rows
25 colScopeRow          Which col has scope? Could be the UID or a names,
      begins with 1
26 defaultSortBy        first sorting action should sort ascending or
      descending?
27 colsToHide           array; set value true if col should be hided, example
      : colsToHide[3] = true;
28 rowsToHide           array; set value true if row should be hided, example
      : rowsToHide[3] = true;
29 keyboard              activate default keyboard control
30 pager                 add default pager control; (do use with rowsToShow < all
      rows in the original table)
31 textPager             String pager
32 textAsc               String for sorting ascending
33 textDesc              String for sorting descending
34 disabled              deactivate the widget
35
36 * Callbacks
37 onInit
38 onUpdateData
39 onSetHTML
40 onRowSort
41
42 * public Methods
```

```
43 updateData
44 setHTML
45 rowSort
46 colSwitch
47 buildPager
48 setPager
49 disable
50 enable
51 disable
52 destroy
53
54 */
55 (function($) {
56 // necessary global var for STRING.sort() function clauses
57 var sortIndex = 0;
58 $.widget("ui.ariaSortTable", {
59     _init: function() {
60         var options = this.options, self = this;
61         // ARIA / make UID if no ID is set by default
62         var elementID = self.element.attr("id");
63         if (elementID != "") {
64             options.uid = elementID;
65         } else {
66             options.uid = new Date().getTime();
67             self.element.attr("id", "ui-table-"+options.uid)
68         }
69         self.element.find("caption").attr("id", "ui-table-"+options.uid+"-caption");
70         self.element
71             .attr("role","grid")
72             .attr("aria-readonly","true")
73             .attr("aria-labelledby", "ui-table-"+options.uid+"-caption");
74
75         // bubbling event for th link elements
76         var theadTr = self.element.find("thead tr")
77             .bind("click", function(e){
78                 if (!options.disabled) {
79                     var el = th = $(e.target);
80                     // get the th element
81                     while (!th.is("th")) {
82                         th = th.parents("th");
83                     }
84                     if (!th.hasClass("ui-table-deactivate")) {
85                         e.preventDefault();
86                         // start sorting / parameter: index of the clicked th
87                         // element
88                         self.rowSort(th.prevAll("th:visible").length);
89                     }
90                 }
91             })
92     }
93 }
```

```

91      })
92      .attr("role", "row");
93
94      // save header elements (th)
95      options.headers = theadTr.find("th");
96      options.headers.each( function(index) {
97          // get single th element
98          var th = $(options.headers[index]);
99
100         // ARIA
101         th.attr("id", "ui-table-" + options.uid + "-header-" + index)
102         .attr("role", "columnheader")
103         .attr("scope", "col");
104
105         // select title text ( next sorting action )
106         var text = (options.defaultSortBy == "asc") ? options.textAsc :
107             options.textDesc;
108         var thA = th.find("a").length;
109         if (!th.hasClass("ui-table-deactivate")) {
110             // no link but JS sort function? Add link
111             if (!thA) {
112                 th.html('<a title="' + text + '" href="#ui-table-dummy">' + th.
113                     html() + '</a>');
114             }
115             // set title attribute / add events
116             th.children("a")
117                 .attr("title", text)
118                 .bind("mouseenter", function() { $(this).parent().addClass('ui-
119                     -state-hover'); })
120                 .bind("mouseleave", function() { $(this).parent().removeClass(
121                     'ui-state-hover'); });
122             // not activated th elements with no link are added to the
123             // tabindex by setting tabindex attribute
124             if (!thA) {
125                 th.attr("tabindex", 0);
126             }
127
128             // save pre sorted (server site) ; aka active col / set
129             // attributes
130             if (th.hasClass("ui-state-active")) {
131                 if (th.hasClass("ui-table-asc")) {
132                     th.attr("aria-sort", "ascending").children("a").attr("title",
133                         "", options.textDesc);
134                 } else if (th.hasClass("ui-table-desc")) {
135                     th.attr("aria-sort", "descending").children("a").attr("title",
136                         "", options.textAsc);
137                 }
138                 options.activeCol = index;
139             }
140         });
141     });
142 
```

```
133
134     // get all table data and save them
135     var rows = self.element.find("tbody tr");
136     // go through every row
137     for (var x = 0; x < rows.length; x++) {
138         options.originalData[x] = [];
139         var cells = $(rows[x]).children("td");
140         for (var y = 0; y < cells.length; y++) {
141             options.originalData[x][y] = $(cells[y]).html();
142         }
143     }
144     // set var to table length if no custom value
145     if (!options.rowsToShow) options.rowsToShow = rows.length;
146
147     // update data to delete hided rows and cols
148     self.updateData();
149     // set new HTML (with ARIA)
150     self.setHTML();
151     // pager?
152     if (options.pager) self.buildPager();
153     // activate Keyboard accessibility
154     if (options.keyboard) self._setKeyboard();
155     // Callback
156     self._trigger("onInit", 0);
157 },
158
159 // make another "cleaned" version of the data array / delete hidden
160 // rows and cols
160 updateData: function () {
161     var options = this.options, self = this;
162     options.tableData = [];
163     var xIndex = 0;
164     for (var x = 0; x < options.originalData.length; x++) {
165         if (!options.rowsToHide[x]) {
166             options.tableData[xIndex] = [];
167             for (var y = 0; y < options.headers.length; y++) {
168                 if (!options.colsToHide[y]) options.tableData[xIndex].push(
169                     options.originalData[x][y]);
170             }
171             xIndex++;
172         }
173     }
174     // Callback
175     self._trigger("onUpdateData", 0);
175 },
176
177 // set new HTML with selected data
178 setHTML: function () {
179     var options = this.options, self = this;
180     // var for diffrent row colors
```

```

181  var second = true;
182  var html = [];
183  html.push("<tbody class=\"ui-table-tbody-active\" aria-
184    live=\"polite\" aria-relevant=\"text\">\n");
185  for (var x = options.rowToStart - 1; x < options.rowToStart - 1 +
186    options.rowsToShow; x++) {
187    // check if row data exists
188    if (options.tableData[x]) {
189      // diffrent row css class
190      var rowClass = (second) ? "class=\"odd\"" : "";
191      second = (second) ? false : true;
192      html.push("\t\t\t\t<tr role=\"row\""+rowClass+">\n");
193      // build table html (with ARIA and HTML table relation
194      // attributes)
195      for (var y = 0; y < options.tableData[x].length; y++) {
196        if (y+1 == options.colScopeRow) {
197          html.push("\t\t\t\t\t<td headers=\"ui-table-"+ options.
198            uid +"-header-"+ y +"\" scope=\"row\" role=\"rowheader
199            \">" + options.tableData[x][y] + "</td>\n");
200        } else {
201          html.push("\t\t\t\t\t<td headers=\"ui-table-"+ options.
202            uid +"-header-"+ y +"\" role=\"gridcell\">" + options.
203            tableData[x][y] + "</td>\n");
204        }
205      }
206      html.push("\t\t\t\t</tr>\n");
207    }
208  }
209  html.push("\t\t\t</tbody>");
210  var str = '';
211  str = html.join('');
212  // replace tbody or hide original and add new / dont remove but
213  // hide because of performance
214  var tbody = self.element.find("tbody.ui-table-tbody-active");
215  if (tbody.length) {
216    tbody.replaceWith(str);
217  } else {
218    self.element.find("tbody").hide();
219    self.element.append(str);
220  }
221
222  // show or hide header cols
223  if (options.colsToHide)
224    options.headers.each( function(index) {
225      if (!options.colsToHide[index]) {
226        $(this).show();
227      } else {
228        $(this).hide();
229      }
230    });

```

```
223
224    // ARIA
225    $(options.headers[0]).parent().parent()
226        .attr("aria-live", "polite")
227        .attr("aria-relevant","text");
228    // update virtual Buffer
229    self._updateVirtualBuffer();
230    // Callback
231    self._trigger("onSetHTML", 0);
232},
233
234 // sort data, build and add the new html to the DOM
235 rowSort: function (index) {
236     var options = this.options, self = this;
237     // get all visible th elements
238     var thArray = options.headers.filter(":visible");
239     // get new (clicked) th element
240     th = $(thArray[index]);
241
242     // set global index
243     sortIndex = index;
244     // check the css class and sort the array
245     if (th.hasClass("ui-table-number")) {
246         options.tableData.sort(self._sortNumber);
247     } else if (th.hasClass("ui-table-number-de")) {
248         options.tableData.sort(self._sortNumberDE);
249     } else if (th.hasClass("ui-table-date")) {
250         options.tableData.sort(self._sortDate);
251     } else if (th.hasClass("ui-table-date-de")) {
252         options.tableData.sort(self._sortDateDE);
253     } else if (th.hasClass("ui-table-date-iso")) {
254         options.tableData.sort(self._sortDateISO);
255     } else {
256         options.tableData.sort(self._sortText);
257     }
258
259     // set new sorted by
260     var asc = th.hasClass("ui-table-asc");
261     if (asc || th.hasClass("ui-table-desc")) {
262         var newSortBy = (asc) ? "desc" : "asc";
263         // no class found? set it by default
264     } else {
265         var newSortBy = options.defaultSortBy;
266     }
267
268     // reverse array if necessary
269     if (newSortBy == "desc") options.tableData.reverse();
270
271     // get active col
272     var thActiveCol = $(thArray[options.activeCol]);
```

```

273    // set class to remove of the active col
274    var sortedBy = (thActiveCol.hasClass("ui-table-asc")) ? "asc" : "desc";
275    // delete css class of the last sorted col
276    thActiveCol
277        .removeClass("ui-table-" + sortedBy)
278        .removeClass("ui-state-active")
279    // set ARIA-sort to none
280        .attr("aria-sort", "none");
281
282    // remove focus classs from selected col
283    $(thArray[options.selectedCol]).removeClass("ui-state-focus");
284
285    // set new title text
286    var newSortByText = (newSortBy == "asc") ? options.textDesc :
287        options.textAsc;
288    var newSortByARIA = (newSortBy == "asc") ? "ascending" : "descending";
289    // add new css class, title and css state to the new active col
290    th.addClass("ui-state-active")
291        .addClass("ui-table-" + newSortBy)
292        .attr("aria-sort", newSortByARIA)
293        .children("a").attr("title", newSortByText);
294
295    // save new sorted and active col
296    options.activeCol = options.selectedCol = index;
297
298    // Callback
299    self._trigger("onRowSort", 0);
300    // update HTML
301    self.setHTML();
302},
303// sorting clauses function
304 _sortNumber: function (a, b) {
305    // 123.456
306    return (a[sortIndex] - b[sortIndex]);
307},
308 _sortNumberDE: function (a, b) {
309    // 123,456
310    return (a[sortIndex].replace(", ", ".") - b[sortIndex].replace(", ",
311        "."));
312},
313 _sortDateDE: function (a, b) {
314    // 28.07.2009
315    var aDate = a[sortIndex].substr(3,2) + "/" + a[sortIndex].substr(0,2) +
316        "/" + a[sortIndex].substr(6,4);
317    var bDate = b[sortIndex].substr(3,2) + "/" + b[sortIndex].substr(0,2) +
318        "/" + b[sortIndex].substr(6,4);
319    return (Date.parse(aDate) < Date.parse(bDate));
320},
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779

```

```
317     _sortDate: function (a, b) {
318         // 07/28/2009
319         return (Date.parse(a[sortIndex]) < Date.parse(b[sortIndex]));
320     },
321     _sortDateISO: function (a, b) {
322         // 2009-07-28
323         var aDate = a[sortIndex].substr(5,2) + "/" + a[sortIndex].substr
324             (8,2) + "/" + a[sortIndex].substr(0,4);
325         var bDate = b[sortIndex].substr(5,2) + "/" + b[sortIndex].substr
326             (8,2) + "/" + b[sortIndex].substr(0,4);
327         return (Date.parse(aDate) < Date.parse(bDate));
328     },
329     _sortText: function (a, b) {
330         // 20:00:13
331         // Text, no html
332         return (a[sortIndex] > b[sortIndex]);
333     },
334     // set keyboard control
335     _setKeyboard: function () {
336         var options = this.options, self = this;
337         // listen and save the shift key event
338         options.shift = false;
339         $(document)
340             .keyup(function(e){
341                 if (e.keyCode == $.ui.keyCode.SHIFT && !options.disabled) {
342                     options.shift = false;
343                     return true;
344                 }
345             })
346             .keydown(function(e){
347                 if (e.keyCode == $.ui.keyCode.SHIFT && !options.disabled) {
348                     options.shift = true;
349                     return true;
350                 }
351             });
352         self.element
353             .keydown(function(e){
354                 if (!options.disabled) {
355                     switch (e.keyCode) {
356                         // go to next page
357                         case $.ui.keyCode.DOWN:
358                         case $.ui.keyCode.PAGE_DOWN:
359                             // check if new value is in range and if there are any
360                             // pages to show
361                             if (options.rowToStart < options.tableData.length -1 &&
362                                 options.rowsToShow != options.tableData.length) {
363                                 if (options.pager) self.setPager(options.rowToStart +
364                                     options.rowsToShow);
365                                 options.rowToStart += options.rowsToShow;
```

```
362         self.setHTML();
363     }
364     break;
365 // go to previous page
366 case $.ui.keyCode.UP:
367 case $.ui.keyCode.PAGE_UP:
368 // check if new value is in range and if there are any
369 // pages to show
370 if (options.rowToStart > 0 + options.rowsToShow &&
371     options.rowsToShow != options.tableData.length) {
372     if (options.pager) self.setPager(options.rowToStart -
373                                     options.rowsToShow);
374     options.rowToStart -= options.rowsToShow;
375     self.setHTML();
376 }
377 break;
378 // go to first page
379 case $.ui.keyCode.HOME:
380 // check if there are any pages to show
381 if (options.rowsToShow != options.tableData.length) {
382     options.rowToStart = 1;
383     self.setHTML();
384 }
385 break;
386 // go to last page
387 case $.ui.keyCode.END:
388 // check if there are any pages to show
389 if (options.rowsToShow != options.tableData.length) {
390     options.rowToStart = ((Math.ceil(options.tableData.
391                                     length / options.rowsToShow)) * options.rowsToShow)
392             - options.rowsToShow + 1;
393     self.setHTML();
394 }
395 break;
396 // go to next or previous page
397 case $.ui.keyCode.TAB:
398 if (options.shift) {
399     if (options.selectedCol > 0) { self.colSwitch(-1) }
400     else { return true; }
401 } else {
402     if (options.selectedCol < options.headers.filter(":"
403                                                 .visible").length - 1) { self.colSwitch(1); } else {
404         return true;
405     }
406 }
407 break;
408 // switch to left col
409 case $.ui.keyCode.LEFT:
410     if (options.selectedCol > 0) self.colSwitch(-1);
411     break;
412 // switch to right col
```

```
404     case $.ui.keyCode.RIGHT:
405         if (options.selectedCol < options.headers.filter(":visible").length - 1) self.colSwitch(1);
406         break;
407         // start sorting
408     case $.ui.keyCode.SPACE:
409         var th = options.headers.filter(":visible");
410         $(th[options.selectedCol]).find("a").click();
411         break;
412     default:
413         return true;
414         break;
415     }
416     return false;
417   }
418 })
419 },
420 // switchh selected col
421 colSwitch: function (dir) {
422   var options = this.options, self = this;
423   // get visible headers
424   var thArray = options.headers.filter(":visible");
425   // remove old selected col css class
426   $(thArray[options.selectedCol]).removeClass("ui-state-focus");
427   // set new selected col
428   options.selectedCol = options.selectedCol + dir;
429   // get new selected col
430   el = $(thArray[options.selectedCol]);
431   // set new focus
432   el.addClass("ui-state-focus");
433   // set focus
434   if (el.find("a").length) { el.find("a").focus(); } else { el.focus(); }
435 },
436 // removes instance and attributes
437 destroy: function() {
438   this.element
439     .unbind(".ariaSorTable")
440     // remove data
441     .removeData('ariaSorTable')
442     // remove attributes
443     .removeAttr("role")
444     .removeAttr("aria-readonly")
445     .removeAttr("aria-labelledby")
446       // remove attributes of the caption
447       .find("caption").removeAttr("id")
448     // remove ARIA attributes from head element
449     .end().find("thead")
450     .removeAttr("aria-live")
451     .removeAttr("aria-relevant")
```

```

452      // remove event and role of the tr and show them all
453      .find("tr")
454      .removeAttr("role")
455      .unbind("click")
456      .end().end()
457      // remove injected HTML
458      .find("tbody.ui-table-tbody-active").remove().end()
459      // show hidden original html
460      .find("tbody").show();
461      // th's
462      $.each(this.options.headers, function() {
463          $(this)
464          .show()
465          .removeAttr("id")
466          .removeAttr("role")
467          .removeAttr("aria-sort")
468          .removeAttr("tabindex")
469          .removeAttr("scope");
470          // search for added links and delete them / remove event
471          var link = $(this).children("a");
472          if (link.length) {
473              link.unbind("mouseenter mouseleave").removeAttr("title");
474              if (link.attr("href") == "#ui-table-dummy") $(this).html(link
475                  .html());
476          }
477      });
478      // pager
479      if (this.options.pager) $("#ui-table-pager").remove();
480      // remove virtual buffer form
481      $("#virtualBufferForm").parent().remove();
482  },
483  // updates virtual buffer / for older screenreader
484  _updateVirtualBuffer: function() {
485      var form = $("#virtualBufferForm");
486      if (form.length) {
487          (form.val() == "1") ? form.val("0") : form.val("1")
488      } else {
489          var html = '<form><input id="virtualBufferForm" type="hidden">
490          value="1" /></form>';
491          $("body").append(html);
492      }
493  });
494
495 $.extend($.ui.ariaSortTable, {
496     version: "1.7.1",
497     defaults: {
498         rowToStart: 1,
499         rowsToShow: false,

```

```

500     colScopeRow: 1,
501     defaultSortBy: "asc",
502     colsToHide: false,
503     rowsToHide: false,
504     keyboard: true,
505     pager: false,
506     textPager: "Page:",
507     textAsc: "Sort ascending",
508     textDesc: "Sort descending",
509     //do not alter these vars
510     selectedCol: 0,
511     activeCol: 0,
512     tableData: [],
513     originalData: []
514   }
515 });
516 $.fn.extend($.ui.ariaSortTable.prototype,{
517   // build a pager
518   buildPager: function () {
519     var options = this.options, self = this;
520     // build html to inject
521     var site = 0;
522     var y = 0;
523     var html = '<div id="ui-table-pager" aria-controls="ui-table-'+
524       options.uid+'>+'\n';
525     html += '<span id="ui-table-'+options.uid+'-pager-title"
526       class="ui-corner-all">' + options.textPager + '</span>+'\n';
527     while (y < options.tableData.length){
528       html += ' <button title="'+options.textPager+' '+ (site + 1) +''
529         " type="button" class="ui-state-default ui-corner-all" aria-
530         selected="false" aria-labelledby="ui-table-'+options.uid+'-
531         pager-title">' + (site + 1) + '</button>+'\n';
532       site++;
533       y = y + options.rowsToShow;
534     }
535     html += '</div>+'\n';
536     self.element.append(html);
537     // ARIA
538     options.pager = $("#ui-table-pager")
539     .attr("aria-valuemin", 1)
540     .attr("aria-valuemax", site);
541
542     // set events / change css classes and sort table
543     options.pagerButtons = options.pager.find("button")
544     .each( function(index) {
545       $(this)
546       .bind("click", function(){
547         // calculate new start position
548         var newRowToStart = (options.rowsToShow * index == 0) ? 1 : (
549           options.rowsToShow * index)+1;

```

```

544     // set pager
545     self.setPager(newRowToStart);
546     // set new start point
547     options.rowToStart = newRowToStart;
548     // set new html
549     self.setHTML();
550   })
551   .bind("mouseenter", function(){ $(this).addClass('ui-state-
      hover'); })
552   .bind("mouseleave", function(){ $(this).removeClass('ui-state-
      hover'); })
553   .bind("focus", function(){ $(this).addClass('ui-state-focus'); })
554   .bind("blur", function(){ $(this).removeClass('ui-state-focus'); })
555 );
556   // set active button after set events
557   self.setPager(options.rowToStart);
558 },
559
560 // sets active page / call before setting new options.rowToStart
561 // with new row as parameter
561 setPager: function (newRow) {
562   var options = this.options, self = this;
563   // calculate new start point and add or remove css classes and
564   // ARIA attributes
565   $(options.pagerButtons[Math.floor(options.rowToStart/options.
      rowsToShow)]).removeClass('ui-state-active').attr("aria-
      selected", false);
566   $(options.pagerButtons[Math.floor(newRow/options.rowsToShow)])
      .addClass('ui-state-active').attr("aria-selected", true);
567   options.pager.attr("aria-valuenow", Math.floor(newRow/options.
      rowsToShow)+1);
568 }
569 });
570 })(jQuery);

```

Listing B.7: sorTable: JavaScript Quellcode

ANHANG C

Danksagung

Vielen Dank an:

- Lilit fürs Korrekturlesen und Schreibstil verbessern.
 - Meinen guten Freund Adrian, der mir während des ganzen Studiums immer wieder mit Rat und Tat zur Seite stand.
 - Besonderen Dank an El Doctore Rosenthal, Tofferino, Claudio und Benni für ihre Geduld und Mühen – ohne euch wäre das nie klar gegangen.
 - Prof. Kampschulte für die Betreuung während des BPS und der Diplomarbeit.
 - Die ganzen tollen Menschen bei **Namics**; im speziellen Andreas, Carina, Chris, Eduard, Hendrik, Janko, Martina und Markus.
 - Alexander Stirn und Martin Kliehm für die fachliche Betreuung.
 - My Lady, für's da sein und beistehen.
 - **Marco Zehe** und **Gez Lemon** für den Support via Twitter
 - Alle meine Kommilitonen – wir waren ein super Team!
 - An alle **LATEX** Entwickler sowie die Macher von **LiveTex**
 - **Matthias Pospiech** für die **LATEX** Vorlage nach der dieses Dokument entstand
 - An alle **Freemind** Entwickler die mir ein tolles Tool zur Erstellung der Funktions-übersichten an die Hand gegeben haben.
 - Alle Wikipedia Autoren - *keep sharing knowledge!*
- ...sowie alle die vergessen gegangen sind.