Nr. 8

KnowWare EXTRA

DM 8,-
€ 4,09

Barrierefreies Webdesign

Barrierefreies Webdesign

Normale Ansicht

Ausgang:

Bitte mindestens eins aus der roten Liste wählen!

- Ich bin über 18
- Ich bin nicht über 18

Erweiterung:

In Sugar Mountain darf keiner über 18 sein! Daher musst du uns auf dein Ehrenwort mitteilen, ob Du schon 18 bist oder nicht:

- *Ich bin über 18*
- *Ich bin nicht über 18*

Ansicht ohne Farbdarstellung

Ausgang:

Bitte mindestens eins aus der roten Liste wählen!

- Ich bin über 18
- Ich bin nicht über 18

Erweiterung:

In Sugar Mountain darf keiner über 18 sein! Daher musst du uns auf dein Ehrenwort mitteilen, ob Du schon 18 bist oder nicht:

- *Ich bin über 18*
- *Ich bin nicht über 18*

Jan Eric Hellbusch

www.KnowWare.de

SFR 8,- ÖS 64,- LFR 194,- LIT 11.000 DM 8,-

4 395087 908001 01008

Barrierefreies Webdesign

Jan Eric Hellbusch, hellbusch@online.de
ISBN 87-90785-75-4, 1. Ausgabe, 1. Auflage: 2001-11

Michael Maardt, verlag@knowware.de - Karl Antz, lektorat@knowware.de
Printer: OTM Denmark, Binder: Gramo Denmark, Published by KnowWare

Nachbestellung für Endverbraucher und Vertrieb für den Buchhandel

Bonner Presse Vertrieb
Möserstr. 2-3
D-49074 Osnabrück
Tel.: +49 (0)541 33145-20
Fax: +49 (0)541 33145-33
knowware@bpv-online.de
Ein Bestellformular findest du online hier:
www.knowware.de

Vertrieb für den Zeitschriftenhandel:

IPV Inland Presse Vertrieb GmbH
Postfach 10 32 46
D-20022 Hamburg
Tel.: (040) 23711-0
Fax: (040) 23711-215

Worum es geht

Hinter **KnowWare** steht der Gedanke, Wissen leichtverständlich und preisgünstig zu vermitteln. Das Projekt startete im April 1993 mit der Herausgabe des ersten Computerheftes in Dänemark. Seitdem sind in vielen Ländern zahlreiche weitere Hefte mit Themen rund um den Computer erschienen.

www.knowware.de

Auf unserer Homepage findest du Beschreibungen und Bilder aller Hefte, geplante Hefte, Online-Bestellung, Anmeldung für einen kostenlosen Newsletter, Tipps & Tricks, Informationen über Sonderdruck für Firmen, neue Autoren, KnowWare in anderen Ländern, Serviceseiten, Händlerlisten usw.

Kostenlose Download

Auf unserer Homepage kannst du kostenlos einige Seiten aus jedem Heft im PDF Format downloaden. Ausverkaufte Hefte: das ganze Heft als PDF ist kostenlos.

Wo und wann sind die Hefte erhältlich?

Die Hefte sind im allgemeinen zwei Monate im Handel, und zwar bei Kiosken, im Bahnhofsbuchhandel und im Buchhandel – bei vielen Verkaufsstellen sowie im Buchhandel auch länger. Alle beim Verlag vorrätigen Titel sind jederzeit nachbestellbar.

Nachbestellung

Es gibt 2 Möglichkeiten:

- bei deinem KnowWarehändler - Bestellformular am Ende des Heftes ausfüllen!
- beim Bonner Presse Vertrieb, siehe links

www.knowware.de

Einleitung

Mit dem World Wide Web, dem WWW, ist ein Medium entstanden, das neue Möglichkeiten der Informationsbeschaffung bietet. Es ist selbstverständlich, dass alle diese Möglichkeit nutzen; ebenso selbstverständlich sollte sein, dass dabei niemand diskriminiert wird. Bei der Gestaltung von Webseiten wird aber oft nicht berücksichtigt, dass Menschen mit einer Sinnesbehinderung – Blinde, Sehbehinderte, Gehörlose – oder anderen körperlichen Einschränkungen – Muskel-, Gelenk- oder Nervenerkrankungen – bei einigen Techniken der Web-Programmierung die Seiten nicht lesen oder navigieren können. Bei solchen "Barrieren" handelt es sich u.a. um folgendes: fehlende Texte zur Bildbeschreibung, eine nur Maus- statt einer Maus- und Tastatur-orientierten Navigation, unglückliche Farbkombinationen, die Farbblinde hindert, Texte und Bilder zu erkennen usw.

Laut der amerikanischen Regierung haben 8% der Surfer eine echte Behinderung und insgesamt 20% eine Einschränkung, wodurch sie mit den unterschiedlichsten Barrieren im Netz konfrontiert werden. Diese große Zahl gibt zu denken und war der Anlass, diese Broschüre zu schreiben!

Außerdem ist jede aufgehobene Barriere nicht etwa nur für Menschen mit einer körperlichen Einschränkung eine große Hilfe; sie nützt auch vielen anderen – sogar dir als dem Autor einer Seite; kann z.B. ein blinder Mensch deine Seiten lesen, kann eine Suchmaschine das ebenfalls!

Eine vorhandene Website anpassen

Die Barrierefreiheit deiner Website sollte schon bei der Planung und Gestaltung des Layouts berücksichtigt werden. Sind die Seiten bereits erstellt, sind folgende Schritte pragmatischer:

- Wichtigste Seiten, d.h. die Startseite sowie viel besuchte Seiten, als erste an vorrangige Anforderungen zur Barrierefreiheit anpassen
- Vorlagen für neue Seiten so gestalten, dass sie die Anforderungen der Barrierefreiheit erfüllen
- Barrierefreiheit in die Kontrolle einbinden
- Weitere, eher nachrangige Seiten an die wichtigen Kriterien anpassen
- Wichtigste Seiten auch an nachrangige Kriterien der Barrierefreiheit anpassen.

Du merkst, es fehlen die "unwichtigen" Seiten. Der Grund: je nach dem Umfang einer Website kann die Überarbeitung aller ihrer Seiten sehr viel Zeit beanspruchen, da die Korrekturen meist "per Hand" gemacht werden müssen. Bietet dein Provider eine Besucher-Statistik mit Auswertung an, erfährst du hier, welche Seiten deines Webauftritts als "wichtig" eingestuft werden können und welche als "weniger wichtig".

Diese Broschüre ist keine Einführung in das Webdesign, sondern beschreibt anhand von Beispielen, wie du Webdesign mit korrektem HTML umsetzt, damit jede Zugangssoftware deine Seiten verarbeiten kann.

Kriterien der Barrierefreiheit

Die Broschüre bietet zunächst ab Seite 5 eine Einführung in das Thema, indem ich auf Behinderungen eingehe und die speziellen Hilfsmittel vorstelle, die z.B. blinde Menschen für den Zugang zu dem Computer einsetzen. Dort findest du auch wichtige Informationen über geltende Standards für Barrierefreiheit im Web. Die Kriterien der Barrierefreiheit werden in dieser Broschüre als vor- bzw. nachrangig eingestuft und in zwei großen Kapiteln beschrieben. Die offiziellen Richtlinien des World Wide Web-Konsortiums (W3C) resp. der Web Accessibility Initiative (WAI) haben eine Aufteilung der Richtlinien auf drei verschiedene Prioritäten. Ich ziehe es vor, einige Kriterien aus der mittleren Gruppe der WAI-Richtlinien als "wichtig" zu betrachten (ab Seite 11) und andere als "nachrangig", um auch deine Arbeit zu erleichtern. Letztere werden ab Seite 50 beschrieben.

In der Mitte der Broschüre findest du vier Seiten mit einer Übersicht der Kriterien zu barrierefreiem Webdesign, die du abtrennen und neben der Tastatur legen kannst, wenn du an der Barrierefreiheit arbeitest. Verweise ich in der Broschüre auf ein Kriterium z.B. mit "WAI-8.1" auf die WAI-Richtlinie 8.1, kannst du dann schnell nachschauen, wie die Richtlinie lautet und welche anderen Regeln damit verwandt sind.

Ab Seite 76 findest du einige Tipps zum Testen der Barrierefreiheit sowie weitere Informationen im WWW, die thematisch relevant sind.

Was ist barrierefreies Webdesign?

Im Angloamerikanischen wird beim Thema "barrierefreies Webdesign" von "Accessibility" gesprochen, was oft übersetzt wird als "Zugänglichkeit". Ich finde dieses Wort sehr unglücklich, da es nicht gerade vielsagend ist; assoziiert überhaupt jemand etwas damit, dann vermutlich nicht etwa die Lesbarkeit von Software, sondern eher den Zugang zum Internet über einen Provider.

Barrierefreies Webdesign bedeutet, Webseiten so zu gestalten, dass sie von jedermann gelesen und bedient werden können. Dabei kann es sich um sehr unterschiedliche Dinge handeln.

Die größten Schwierigkeiten beim Lesen von traditionell erstellten Webseiten haben Benutzer von textorientierten Browsern oder grafischen Browsern mit ausgeschalteter Grafikfunktion: solche Benutzer

- sind nicht bereit, überflüssige Grafiken zu laden und die dadurch entstehenden längeren Ladezeiten in Kauf zu nehmen.
- sind unterwegs, benutzen einen Laptop, wählen sich per Modem ins Internet ein oder haben eine schlechte Grafikauflösung.
- surfen mit einem Palmtop oder einem Handy, was limitierte Datentransfervolumen und sehr hohe Übertragungskosten bedeutet.
- suchen nur Textinformationen.
- benutzen statt eines Monitors eine Sprachausgabe, weil sie blind oder sehbehindert sind.
- ziehen einen Lynx- oder emacs-w3-Browser vor.
- surfen in einem Gebiet mit niedrigen Übertragungsraten, etwa in Entwicklungsländern.

Diese Liste kann beliebig fortgesetzt werden – wobei du die Suchmaschinen nicht vergessen solltest, die wie textorientierte Browser arbeiten!

Was haben diese Beispiele eigentlich gemeinsam, außer der Verwendung eines textorientierten Browsers? Wahrscheinlich nichts! Selbst wenn du dein Besucherprofil relativ gut kennst, wirst du es in der Regel nicht schaffen, allen Anforderungen deines Publikums zu genügen. Platzierst du Werbung in einer Zeitschrift, die von deinem Klientel nicht gelesen wird, hast du einfach nur daneben geschossen. Abgesehen davon, dass die Zeitschrift dein Geld verdient hat, ist kein Unglück passiert. Programmierst du aber deine WWW-Seiten in einer Weise, die manchem Besucher das Lesen erschwert oder unmöglich macht, jagst du ihn schnell zu Konkurrenzangeboten. Außerdem wird er deine Seite kaum weiter empfehlen – eher das Gegenteil.

Menschen mit körperlichen Einschränkungen

Die Zahl der "Abweichungen" vom Durchschnitts-Surfer ist schwer einzuschätzen, sie beträgt aber mit Sicherheit mehr als 20%. Zumindest ist das laut **www.bigub.de**, dem "Verein für Behinderte in Gesellschaft und Beruf", die Quote derer in Deutschland, die wegen einer Einschränkung die eine oder andere Schwierigkeit im WWW haben.

Barrieren ergeben sich vor allem dann, wenn der Surfer schlecht bzw. gar nicht sehen kann, in seiner Bewegung, also der Bedienung der Maus eingeschränkt ist oder schlecht bzw. gar nicht hört. Dazu kommen mehrere weitere Barrieren, die auf kognitiven Gegebenheiten wie Konzentrationsschwäche beruhen. In dieser Broschüre geht es darum, Notwendigkeit und Bedeutung der Barrierefreiheit für Betroffene darzustellen.

Wer schlecht oder gar nicht sieht

Ohne Zweifel stehen Sehbehinderte und Blinde, die mit dem Computer ins Internet wollen, vor den größten Barrieren. Das liegt daran, dass sie spezielle Hilfsprogramme benutzen müssen.

Screen-Reader und Textorientierung

Blinde und viele Sehbehinderte benötigen Software, die die Signale für den Bildschirm abfängt und neu interpretiert. Eine Variante ist die Spezial- oder Brückensoftware, die als "Screen-Reader" bezeichnet wird. Die Übersetzung des Bildschirminhalts durch den Screen-Reader erfolgt entweder in synthetischer Sprache z.B. über eine Soundkarte oder aber in Blindenschrift über die Braille-Zeile; letztere ist ein spezielles Ausgabegerät, das Teile des Bildschirminhaltes in Blindenschrift auf einem taktilen Display wiedergibt und meist mit einer Tastatur als Ein- und Ausgabegerät kombiniert wird.

Dies ist eine Möglichkeit, wie Sehbehinderte und Blinde im WWW surfen. Natürlich ist das mit Einschränkungen verbunden – Inhalte von Bildern und Grafiken bleiben nach wie vor "verborgen".

Die Verwendung von Audio-Browsern ist eine weitere Möglichkeit, zu surfen, ohne den Bildschirm lesen zu müssen. Damit werden Inhalte und Formatierung rein akustisch ausgegeben.

Screen-Magnifier oder andere Veränderungen der Darstellung

Ein Screen-Magnifier stellt einen Teil der "normalen" Darstellung vergrößert auf dem Bildschirm dar. Das bedeutet, dass der Benutzer jeweils nur einen kleinen Teil des Bildschirms wahrnehmen kann. Ist er auf eine vierfache Vergrößerung angewiesen, sieht er also jeweils nur 1/16 des Bildes.

Auch andere Anpassungen der Darstellung durch Veränderungen in der Systemsteuerung von Microsoft Windows, z.B. eine neue Einstellung der Farben, können für Sehbehinderte eine große Hilfe sein. Solche Änderungen der Einstellungen gelten auch für die Programme – und somit auch für Internet-Programme. In manchen Fällen führt das zu Problemen, wenn Webgestalter Farben teils selbst definieren, teils ihre Definition dem Betriebssystem überlassen.

Wer keine Maus benutzen kann/ will

Eine weitere Gruppe von Surfern, die vor Barrieren im Internet steht, sind diejenigen, die keine Maus benutzen können und statt dessen darauf angewiesen sind, die Tastatur zur Navigation bzw. Bedienung der Maus zu verwenden. Oft wird nichtsahnend vorausgesetzt, dass jeder eine Maus benutzt. Stelle dir aber vor, du müsstest deinen Computer mit Boxerhandschuhen oder mit deinen Füßen bedienen. Da wäre eine normale Maus sicherlich keine besondere Hilfe. Hier sind spezielle Tastaturvorrichtungen die nützliche Alternative.

Nun gibt es eine ganze Reihe Menschen, die aus verschiedenen Gründen keine Maus benutzen; gerade auch Experten unter den Computernutzern verwenden gerne die Tastatur statt der Maus. Der wesentliche Vorteil dabei ist, dass die Bedienung des Computers wesentlich schneller geht. Der Einsatz von Tastaturkürzeln auf WWW-Seiten ist zwar nicht unbedingt vorgesehen – die Bedienung von Seiten über die Tastatur ist aber dennoch machbar.

Ist die Navigation auf einer Seite nur mit Hilfe der Maus möglich, wird nicht nur diesen Surfern das Leben schwer gemacht. Die Möglichkeit, eine Website ausschließlich mit der Tastatur zu bedienen, ist also auch ein Kriterium für die Zugänglichkeit und Barrierefreiheit.

Weitere Barrieren

Menschen, die schlecht oder gar nicht hören oder nicht oder nur undeutlich sprechen, sind im Web von heute mit relativ wenigen Barrieren konfrontiert. Erst die Zukunft wird zeigen, inwieweit das WWW zum Kommunikationsmittel ausgebaut und zum akustisch-interaktiven Medium wird. Spätestens dann müssen auch für solche Barrieren Alternativen berücksichtigt werden.

Neben diesen technisch orientierten Barrieren, die sich meist die Einhaltung von "Spezifikationen" oder Normen überwinden lassen, gibt es viele sehr individuell ausgeprägte Barrieren.

Ein großer Teil der Sehbehinderten steht auch dann vor Barrieren im Internet, wenn sie keinen Screen-Reader oder -Magnifier einsetzen müssen. Da dieser Personenkreis aufgrund der Vielfältigkeit der Einschränkungen meist sehr individuelle Anforderungen an die Anzeige des Bildschirms hat, ist es hier schwieriger, optimale Anforderungen zu formulieren. Jedoch treten einige grundsätzliche Schwierigkeiten relativ häufig auf.

Es gibt eine ganze Reihe anderer Einschränkungen, die weitere Barrieren im Internet aufzeigen. Denke etwa an Menschen, die Konzentrationsschwächen haben; da ist eine logisch aufgebaute und leicht nachvollziehbare Navigation eine besondere Unterstützung. Ein weiteres Beispiel: je nach Art der Einschränkung können auch blinkende Elemente zur völligen Ablenkung von den Inhalten auf deiner Seite führen und zum Teil auch gesundheitliche Gefahren bedeuten.

Für diese und andere Barrieren wurden von der „Web Accessibility Initiative“ (WAI) 66 Regeln aufgestellt, die in den WAI-Richtlinien (1997) resp. im letzten HTML 4-Standard (zuletzt 1999 in der Version 4.01 aktualisiert) festgehalten sind. Trotz dieser Standards werden die meisten Webseiten auch heute noch nicht barrierefrei gestaltet, weil die Anbieter von HTML-Editoren diese Richtlinien dem Benutzer lediglich als Option überlassen bzw. die Browser die Funktionalitäten nur bedingt enthalten.

Wer die Regeln aufstellt: W3C und WAI

Das World Wide Web Consortium (W3C) ist die höchste "Instanz" für Entwicklungen im WWW. Dieses Gremium wird von Fachleuten aus drei Universitäten organisiert: dem *MIT Laboratory for Computer Science* (USA), dem *National Institute for Research in Computer Science and Control* (Frankreich) und der *Keio University* (Japan). Diesem Konsortium gehören über 235 zum Teil sehr namhafte Vertreter der Software-Industrie an, etwa Mitarbeiter von Microsoft oder Adobe. Die Hauptaufgabe des W3C ist, Standards wie HTML zu definieren und für deren weitere Entwicklung und Kompatibilität zu sorgen. Auf den Webseiten unter **www.w3.org** findest du viele Informationen über Web-Standards sowie einige Beispiele für deine Quelltexte.

Die *Web Accessibility Initiative* (WAI) ist der Teil des W3C, der für die Zugänglichkeit in den W3C-Standards zuständig ist. Barrierefreies Webdesign ist dabei ein Schwerpunkt, der im Standard für HTML 4.0 im Jahr 1997 erstmals in einer Internet-Norm umgesetzt wurde.

Was die WAI für Barrierefreies Webdesign tut

Als Teil ihrer Bemühungen, Zugänglichkeit im WWW zu fördern, hat die WAI im Rahmen der Standardisierung von HTML 4.0 ihren Einfluss geltend gemacht und ihre eigenen Richtlinien als offiziellen Standard aufstellen können (Web Content Accessibility Guidelines 1.0 von 1997), die die Grundlage dieser Broschüre sind. Hier werden viele Erläuterungen einschl. Checklisten und technische Anleitungen zur Implementierung vorgestellt. Was fehlt, sind Beispiele.

Soll die Lesbarkeit von WWW-Seiten garantiert werden, reicht es natürlich nicht aus, dass der Webgestalter alle "Flaschenhälse" berücksichtigt – auch die Hersteller von Browsern, Editoren und Spezialsoftware müssen ihre Bemühungen darauf ausrichten, das Lesen von WWW-Seiten effektiver zu gestalten. Nur muss dann auch der Webgestalter seine Aufgabe meistern.

Verbesserte Strukturierung

Die gute Strukturierung einer Seite ist ein fundamentaler Bestandteil der Barrierefreiheit im WWW-Design. Dies bezieht sich in erster Linie auf den logischen Aufbau der einzelnen Elemente einer HTML-Seite. Aus diesem Grund hat die WAI einige Elemente und Attribute für Elemente – auch im Zusammenhang mit Style-Sheets – in den HTML 4.01-Standard eingebracht. Diese Elemente ermöglichen auch anderer Zugriffs-Software, etwa Suchmaschinen, den effektiveren Zugang zu den Inhalten einer Seite.

Welche der folgenden HTML-Elemente kennst du?

- **ABBR**- und **ACRONYM**-Elemente
- Das **lang**-Attribut
- das **Q**-Element
- **INS**- und **DEL**-Elemente
- **FIELDSET**- und **LEGEND**-Elemente
- Das **OPTGROUP**-Element
- **scope**-, **headers**- und **axis**-Attribute

Sind dir diese HTML-Elemente fremd, ist das keine Schande. Sie werden bisher kaum benutzt, weil HTML-Editoren ihre Anwendung so gut wie nicht unterstützen. Wir sehen uns diese Elemente im dritten und vierten Kapitel der Broschüre an.

Style-Sheets

HTML wurde ursprünglich entwickelt, um Inhalte zu organisieren. Erst Mitte der Neunziger Jahre, als sich das WWW auch außerhalb des universitären Bereichs verbreitete, wurde die Darstellung der Inhalte zunehmend wichtiger. Statt nun eine Reihe zusätzlicher HTML-Elemente in die HTML 4.0-Norm von 1997 aufzunehmen, wurde die Aufgabe der Präsentation von Inhalten – also Dinge wie Design und Formatierung – Style-Sheet-Sprachen wie Cascading Style-Sheets (CSS) zugeordnet. Obwohl solche Style-Sheets kein integrativer Teil von HTML 4 sind, werden sie im vollen Umfang berücksichtigt. Wie wir später sehen werden, führt die Tatsache, dass dies ein Standard ist, leider keinesfalls dazu, dass er von den Browsern auch eingehalten wird!

Es gibt zwei gute Gründe für den Einsatz von Style-Sheets. Zum einen kannst du die Informationen mit minimalen Änderungen an unterschiedliche Medien bzw. Zielgruppen anpassen. Aus Sicht der Barrierefreiheit bieten Style-Sheets einen weiteren Vorteil: sie erlauben die Formatierung deiner Seiten ohne "Tricks".

Solche Tricks bedeuten nämlich oft Barrieren. So bietet HTML z.B. kein Element an, das die erste Zeile eines Absatzes wie in einem Buch einrückt; da nun aber manche Browser das **BLOCKQUOTE**-Element automatisch einrücken, wird es oft für diesen Zweck missbraucht. Benutzt jemand Audio-Software zum Surfen, wird ein solcher Absatz als Zitat angekündigt, obwohl er kein Zitat enthält – ein typisches Beispiel für den Missbrauch von logischen Elementen für Zwecke der Präsentation. Weitere Beispiele sind der Einsatz von Tabellen und transparenten Grafiken zu Layoutzwecken, die Verwendung der Header-Elemente (**H1**, **H2**, ... **H6**) zur Formatierung von Texten, die keine Überschriften sind, oder die Kursivierung von Texten mit Hilfe des **EM**- statt des **I**-Elements, weil gängige Browser das **EM**-Element kursiv darstellen. Hier bieten Style-Sheets verschiedene Möglichkeiten, alle oder einzelne Absatzblöcke, Wörter und Zeichen auf rein optischer Basis zu gestalten.

Alternative Inhalte

Ein Bild kann mehr als tausend Worte sagen – für manche Leute sind aber einige wenige Worte notwendig, damit sie im Bilde sind! Ob nun sehbehindert oder aufgrund von technischen Gegebenheiten und Präferenzen – einige Beispiele habe ich bereits erwähnt – es gibt genügend Gründe, Grafiken und Multimedia um eine alternative Ausgabe zu ergänzen. Dafür gibt es folgende Möglichkeiten:

- das **alt**-Attribut. Für Grafiken (**IMG**- und **AREA**-Elemente) ist dieses Attribut seit HTML 4.0 eine Spezifikation, für Formularfelder (**INPUT**-Element) und Programmierungen (**APPLET**-Element) weiterhin optional.
- das **title**-Attribut. Mit diesem kannst du fast alle Elemente um eine Kurzbeschreibung ergänzen, die bei grafischen Browsern als Pop-Up erscheint, von textorientierten Browsern aber ebenfalls verstanden wird.
- das **longdesc**-Attribut. Es erlaubt die Einbindung einer längeren Beschreibung in einem Zusatzdokument, z.B. bei Bildern und Abbildungen, die nicht in wenigen Worten erklärt werden können.
- das **CAPTION**-Element und das **summary**-Attribut. Diese können zur Beschreibung von Tabellen verwendet werden;
- das **NOFRAMES**-Element. Es dient der Beschreibung von Inhalten für Browser, die keine Frames darstellen.
- das **NOSCRIPT**-Element. Es beschreibt Inhalte für Browser, die keine Scripts verarbeiten.

Von allen Elementen, die die WAI in HTML 4 eingebracht hat, ist das **OBJECT**-Element das wichtigste –wenn seine Verwendung bis heute auch kaum festzustellen ist. Dieses Element vermag alle Arten von Grafiken, Programmierungen und Multimedia darzustellen. Die Darstellung alternativer Inhalte wird auch flexibel gehalten, indem weitere HTML-Elemente im **OBJECT**-Element eingebettet werden – mit konventionellen Attributen wie **alt** ist das nicht möglich.

Navigation und Orientierung

Sehgeschädigte Menschen haben oftmals größte Barrieren im Netz zu überwinden, wenn nämlich die Navigation in hohem Maße von der visuellen Wahrnehmung abhängt. Image-Maps z.B. sind so gut wie unmöglich zu bedienen, wenn sie keine Alternativtexte haben. Und Link-Texte, die nur ein "Hier klicken" enthalten, vielleicht verbunden mit einer Grafik, sind überhaupt nicht hilfreich. Ein weiteres Beispiel sind ohne Zwischenraum nebeneinander positionierte Links, was besonders für Screen-Reader-Nutzer die Unterscheidung der einzelnen Links sehr erschwert.

Seit HTML 4 zum Standard geworden ist, gibt es einige Elemente zur besseren und einfacheren Navigation auf Webseiten:

- Für Client-seitige Image-Maps kann nunmehr neben dem **MAP**-Element auch das **A**-Element mit ergänzten Koordinatenangaben eingesetzt werden. Damit lassen sich zum einen die aktiven Bereiche der Image-Map definieren, und zum anderen kann eine ansprechende wie auch funktionale Alternativversion der Image-Map erstellt werden – alle Links werden zweimal bereitgestellt.
- Mit dem **title**-Attribut kann das **A**-Element um eine kurze Beschreibung ergänzt werden, um zusätzliche Informationen über einen Link anzubieten.

- Mit dem **accesskey**-Attribut kannst du für wichtige Links auf deiner Seite so genannte "Shortcuts" definieren, die dem Besucher die schnelle Anwahl eines Links über die Tastatur ermöglicht.
- Mit dem **tabindex**-Attribut kannst du für Tastatur-Benutzer die Reihenfolge der über die TAB-Taste erreichbaren Links vorgeben, wenn beispielsweise Navigationselemente vom Quelltext her gesehen eher am Ende liegen und daher sehr spät erreicht werden.
- Das **LINK**-Element im Kopfbereich deines Dokuments erlaubt Browsern in Verbindung mit dem **media**-Attribut, Seiten für spezielle Medien aufzurufen, um die teilweise umständliche Suche nach diesen Seiten zu vermeiden.

Was ebenfalls hilft

Überlege dir das Beispiel der abgesenkten Bürgersteige in unseren Städten. Rollstuhlfahrer haben jahrelang gekämpft, um dies durchzusetzen. Diese und ähnliche Maßnahmen, wie Rollstuhl-gerechte Eingänge zu Gebäuden, Bussen, Zügen, usw., haben aber nicht nur den Rollstuhlfahrern das Leben etwas leichter gemacht. Wie oft profitieren auch Eltern mit Kinderwagen oder Fahrradfahrer von dieser Errungenschaft! Die Vorteile von Innovationen, die eigentlich für Behinderte entwickelt wurden, gelten auch für das Thema Webprogrammierung:

- Informationen mit einem hohen Maß an Strukturierung bedeuten Vorteile für alle bei der Orientierung und Navigation deiner Seiten – so können auch Suchmaschinen deine Seiten besser lesen.
- Der Einsatz von Style-Sheets im HTML an Stelle von Grafiken, um schicke Schriftzüge und ansprechende Layouts zu gestalten, vermindert Download-Zeiten erheblich.
- Das Surfen, wenn man nicht konstant auf den Monitor schauen kann, die Hände nicht frei hat oder mit eingeschränkten Darstellungsmöglichkeiten arbeiten muss, z.B. Palmtops, wird erleichtert.

Eine ausführliche Liste der W3C-Standards sowie technische Anleitungen findest du hier:

www.w3.org/TR

Prioritäten setzen

Die Richtlinien der WAI umfassen 66 Regeln, die als Kriterien für barrierefreies Webdesign gelten. Natürlich richten sich diese Regeln nicht nur an Web-Designer, sondern auch an die Hersteller von Browsern, Gestaltungs-Software und Spezialsoftware wie z.B. Screen-Reader. Manche Barrieren können seitens der Software-Branche aufgelöst, andere nur durch aktives Gestalten seitens der Web-Designer behoben werden. Letzten Endes lässt sich vollständige Barrierefreiheit nicht erreichen, da die Entwicklung auf dem Software-Markt stetig weiter geht.

Die WAI hat ihre Richtlinien in drei Prioritätsstufen aufgeteilt. Ohne diese Prioritäten wären viele Webmaster schlicht überfordert, wollten sie alle 66 WAI-Richtlinien auf einmal umsetzen.

- 1. Priorität: 17 Anforderungen, die zu erfüllen sind, um grundsätzliche Barrieren zu umgehen. Ohne die Einhaltung dieser Regeln sind deine Seiten für vielen Menschen nicht lesbar.
- 2. Priorität: 33 Anforderungen, die zu erfüllen sind, um Barrieren zu umgehen. Werden diese Regeln nicht eingehalten, sind bestimmte Personenkreise weiterhin von deiner Website ausgeschlossen.
- 3. Priorität: Weitere 16 Punkte, die der Barrierefreiheit dienen – nachrangige Regeln, deren Verletzung keine gravierenden Folgen hat.

Wie schon zu Anfang erwähnt solltest du bei einer Umgestaltung deiner Seiten pragmatisch vorgehen. Zunächst solltest du die Startseite und die meistbesuchten Seiten deiner Site auf Einhaltung der wichtigen Kriterien der Barrierefreiheit überprüfen und eventuell gleich anpassen. Dies gilt insbesondere auch für alle Seiten, auf denen der Kontakt zu dir hergestellt werden kann.

Ab dem heutigen Tag solltest du versuchen, alle neuen Seiten konform zu allen WAI-Richtlinien zu erstellen. Am besten erstellst du beim Durchlesen dieser Broschüre deine eigene Vorlage, die die Besonderheiten deiner Seiten berücksichtigt.

Dann sollten nach und nach weitere Bereiche deiner Website an die wichtigen Richtlinien mit der Priorität 1 angepasst werden. Auf längere Sicht kannst du auch nachrangige WAI-Anforderungen auf die Startseite und andere wichtige Seiten deiner Site anwenden.

In dieser Broschüre behandele ich die drei Prioritäten in zwei Abschnitten. Im ersten werden die Anforderungen der Priorität 1 sowie einige der mittleren Prioritätsstufe aufgeführt. Im zweiten Abschnitt findest du Beispiele zu weiteren Kriterien mit der Priorität 2 und solchen mit der Priorität 3. Dies ist sinnvoll auf der Basis des eben beschriebenen Ansatzes; außerdem versuche ich die Barrieren, die durch bestimmte Elemente bedingt sein können, wie z.B. Grafiken, jeweils nur in einem Abschnitt zu behandeln.

Allerdings beinhaltet die Problemstellung des barrierefreien Webdesigns mehr als nur HTML-Elemente. Je nach den bei der Gestaltung und Generierung deiner WWW-Seiten eingesetzten Technologien solltest du auch die Kriterien zur Barrierefreiheit des jeweiligen Anbieters ansehen. So solltest du dich z.B. bei Sun Microsystems (**`www.sun.com/tech/access/`**) informieren, wenn du Java-Programme einsetzt, oder bei Microsoft (**`www.microsoft.com/enable/`**), wenn du ActiveX-Controls nutzt. Beide haben barrierefreies Design in ihre Gestaltungsrichtlinien aufgenommen.

Der schnelle Einstieg

Bevor wir in die Materie einsteigen, möchte ich noch die "Quick Tips for Accessible Web Sites" vorstellen. Diese "Quick Tips" sind eine extrem zusammengefasste Version des WAI-Anforderungskatalogs. Sie sollen dir einen ungefähren Überblick über die folgenden Abschnitte geben, wobei die Themen in verschiedenen Abschnitten immer wieder erscheinen werden.. Mitarbeiter des WAI haben diese Liste auf der Rückseite ihrer Visitenkarten abgedruckt, um Neugierigen einen schnellen und leichten Einstieg in die Thematik zu geben. Das WAI bietet keine offiziellen Übersetzungen dieser Richtlinien an.

Grafiken & Animationen	Verwende das **`alt`**-Attribut, um die Funktionen aller visuellen Objekte zu beschreiben (S. 11).
Image-Maps	Verwende Client-seitige MAP und Text für "Hotspots" (S. 13).
Multimedia	Biete Beschriftung und Beschreibung für Audio, Beschreibung für Video sowie zugängliche Versionen für den Fall an, dass unzugängliche Formate verwendet werden (S. 38, 41).
Hyperlinks	Verwende ausschließlich Wörter, die auch ohne Zusammenhang Sinn ergeben. So ist etwas wie "Hier klicken" unbedingt zu vermeiden (S. 12, 59).
Organisation der Seite	Verwende Überschriften, Listen und eine einheitliche Struktur. Soweit möglich, benutze CSS für Layout und Stil (S. 51, 60, 68).
Grafiken & Tabellen	Fasse zusammen oder verwende das **`longdesc`**-Attribut (S. 11, 29).
Scripts, Applets & Plug-Ins	Stelle alternative Inhalte bereit für den Fall, dass sie nicht zugänglich oder nicht unterstützt werden (S. 25).
Frames	Verwende die **`name`**- oder **`title`**-Attribute (S. 35).
Tabellen	Ermögliche zeilenweises Lesen. Vermeide Tabellen bei Spaltenlayout (S. 29, 54).
Überprüfen der Arbeit	Überprüfe das HTML. Verwende Werkzeuge zur Bewertung und setze einen textorientierten Browser ein, um die Zugänglichkeit zu sichern (S. 49, 76).

Eine komplette Liste der Richtlinien findest du in der Mitte dieser Broschüre ab Seite 43

Wie du Barrierefreiheit sicherst

Alternativtexte

Für jedes Element, das kein Text ist, solltest du einen zusätzlichen Text bereitstellen. Dies gilt besonders für Grafik (**IMG**- und **AREA**-Elemente), aber auch für jede andere Form von Multimedia, ob Video, Applet, Sound oder Bilder.

Bilder

Die Nutzer von Screen Readern, Audio-Browsern und/oder textorientierten Browsern haben grundsätzlich keine Möglichkeit, ein Bild oder eine Grafik zu betrachten. Der einzige Weg, etwas über den Inhalt eines solchen Elements zu erfahren, ist über den Alternativtext, der jeder Grafik im HTML-Quellcode beigefügt werden sollte. Ein Screen-Reader, der beim Übersetzen einer WWW-Seite auf eine Grafik trifft, gibt lediglich die Angabe "Grafik" sowie den Alternativtext aus. Fehlt diese textliche Alternative, kann das dazu führen, dass der Screen-Reader eine Seite wie folgt vorliest: "Grafik Grafik Grafik ...". Es versteht sich, dass Alternativtexte große Bedeutung für Screen-Reader-Nutzer haben.

Kurze Alternativtexte

Willst du kurze Beschreibungen von Bildern und anderen Grafiken in deinem HTML-Quelltext einfügen, musst du der Grafik das **alt**-Attribut mit dem Alternativtext beifügen:

```
<IMG src="home.gif" alt="Startseite">
```

Hier wäre darauf hinzuweisen, dass das W3C in dem Entwurf für HTML 3.0 die eingeschränkten Möglichkeiten des **IMG**-Elements, nämlich nur Grafiken einbinden zu können, durch die Einführung des neuen und flexibleren **FIG**-Elements zu ergänzen versuchte, welches leider nur von wenigen fortgeschrittenen Browsern, nicht aber von den Massen-Browsern unterstützt wird. In der HTML 4.0-Norm wurde das **FIG**-Element durch ein weiterentwickeltes Element ersetzt, das **OBJECT**-Element. Obwohl dieses Element eine erhebliche Flexibilität bei der Grafik- und Multimedia-Gestaltung erlaubt, wird es derzeit von Massen-Browsern auch nicht gut unterstützt.

Dennoch: rufst du deine Bilder mit dem **OBJECT**-Element auf, wird der Alternativtext im Körper zwischen den beiden **OBJECT**-Begrenzungen integriert:

```
<OBJECT data="home.gif" type="image/
gif">
  Startseite
</OBJECT>
```

> Das **OBJECT**-Element wird von älteren Browsern nicht oder nur mit einem Plug-In unterstützt. Es ist daher davon auszugehen, dass die meisten Surfer keinen Browser haben, der die volle Funktionalität des **OBJECT**-Elements korrekt interpretieren kann.
> Zeigen deine Test-Browser das Bild nicht an, so gib die Breite und Höhe der Anzeige mit den Attributen **width** und **height** explizit an!

Die meisten Webgestalter benutzen grafische HTML-Editoren zur Erstellung von Webseiten. Benutzt du ebenfalls einen solchen Editor und fügst eine Grafik ein, kannst du mit der Maus die Eigenschaften der Grafik aufrufen und – je nach Editor – unter ALTERNATIVTEXT, BESCHREIBUNG o.ä. den Alternativtext eingeben, ohne in den Quelltext hineinschauen zu müssen.

Lange Alternativtexte

Gelegentlich kann es notwendig sein, einer Grafik eine längere Erläuterung beizufügen, etwa bei Fotos oder Charts. Hier eignet sich das **alt**-Attribut nur bedingt, etwa um darauf hinzuweisen, worum es sich bei der Grafik handelt. Lassen sich die Zusammenhänge aber nur durch die visuelle Wahrnehmung der Grafik verstehen, ist eine Beschreibung der Grafikinhalte erforderlich. Der Aufruf des alternativen Beschreibungstextes ist mit dem **longdesc**-Attribut möglich, indem du die Beschreibung in einer zusätzlichen Datei ablegst und mit dem **longdesc**-Attribut dorthin verweist:

```
<IMG src="umsatz2001.gif"
alt="Chart: Umsätze 2001"
longdesc="umsatz2001.htm">
```

In der Datei UMSATZ2001.HTM beschreibst du dann den Chart (UMSATZ2001.GIF):

Ein Chart, der die Umsatzentwicklung für das Jahr 2001 darstellt. Die Ergebnisse sind in einem Balkendiagramm nach Monaten aufgeschlüsselt. Januar weist eine 10%-ige Umsatzsteigerung gegenüber Dezember 2000 auf, Februar einen 3%-igen Umsatzrückgang, ...

Bisher unterstützen nicht alle Browser die **longdesc**-Funktion. Deshalb solltest du bei Beschreibungen von Grafiken einen Link zur Beschreibung zufügen, den du unmittelbar hinter der Grafik platzierst. Das WAI empfiehlt als Text für den Link ein "D" (für "description" = Beschreibung):

```
<IMG src="umsatz2001.gif"
alt="Chart: Umsätze 2001"
longdesc="umsatz2001.htm">

[<A href="umsatz2001.htm"
title="Beschreibung des Charts
Umsätze 2001">D</A>]
```

Es ist ganz klar, dass der "D"-Link dem **longdesc**-Attribut vorzuziehen ist, wenn es darum geht, die Barrierefreiheit deiner Seiten zu garantieren. Ebenso klar ist aber, dass die "D"-Links in das optische Erscheinungsbild deiner Site eingreifen können. Hier kannst du dir mit folgender Technik Abhilfe verschaffen:

1. Anstatt des "D"-Links als Text füge eine kleine (1 Pixel) oder transparente Grafik als Link ein.
2. Als **alt**-Attribut füge "D"-Link oder nur "D" ein.

```
<IMG src="umsatz2001.gif"
alt="Chart: Umsätze 2001"
longdesc="umsatz2001.htm">

<A href="umsatz2001.htm"><IMG
src="transparent.gif" alt="D-Link"
width="1" height="1"></A>
```

Dieser Link ist für den "Durchschnittsbenutzer" nicht erkennbar. Ein Screen-Reader erfasst diesen Link aber, und das ermöglicht es dem Benutzer, den Inhalt deiner Grafiken nachzusehen. Das funktioniert aber nur mit einem Alternativtext!

Das **OBJECT**-Element vereinfacht das Einfügen eines längeren Beschreibungstextes etwas, indem die Beschreibung zwischen den Begrenzungen des **OBJECT**-Elements eingeschlossen wird:

```
<OBJECT data="umsatz2001.gif"
type="image/gif">

  Balkendiagramm der
Umsatzentwicklung 2001.

  Eine <A href="
umsatz2001.htm">Textbeschreibung</A>
steht zur Verfügung.

</OBJECT>
```

Im Gegensatz zum **alt**-Attribut kannst du im **OBJECT**-Element HTML einbinden, d.h. auch Links zu weiteren Seiten.

Lassen sich deine Grafiken nicht beschreiben, solltest du lieber auf sie verzichten – in diesem Falle musst du nämlich davon ausgehen, dass viele andere auch das Bild nicht verstehen.

Grafische Darstellung von Text, hierunter auch von Symbolen

Bestimmung des Alternativtextes

Besondere Bedeutung kommt dem Alternativtext zu, wenn du Text als Grafik auf deiner Seite einbindest. Ein typisches Beispiel sind Buttons mit Text, die etwa eine Navigationsleiste mit höherem optischen Anspruch gestalten lassen. Besonders als Link eingesetzte Grafiken bedürfen aus der Sicht von Screen-Reader-Anwendern und von Benutzern nicht-grafischer Browser eine präzise Textbelegung. Ein "Bitte hier klicken" in gleich mehrfacher Ausführung oder ein NAV1024.GIF (4K), wobei die Grafiken dann ein für Sehende leicht erkennbares Symbol anzeigen, ist nicht besonders hilfreich. Vielmehr müssen Alternativtexte für Grafiken den Sinn der Grafik widerspiegeln. Wird ein Bild als Navigationselement eingesetzt, sollte der Text auf die dahinter stehende Information schließen lassen:

```
<A href="kontakt.htm"><IMG
src="mail.jpg" alt="Kontakt"></A>
```

Der Alternativtext muss nicht besonders ausführlich sein. Bei Buttons reicht durchaus der Text, der auf dem Button abgebildet ist. Bei anderen Symbolen und kleinen Grafiken ist die Bedeutung meist in wenigen Worten erklärt.

Setzt du ein Logo oder eine andere mehrfach wiederholte Grafik auf deinen Seiten ein, solltest du versuchen, den Alternativtext knapp zu halten. Eine gesonderte Erläuterung von der Seite aus, wo die Grafik zum ersten Mal erscheint, reicht im Prinzip aus. Hier geht es darum, Grafik-lastige Kopfzeilen von Seiten, die von Screen-Readern gnadenlos jedes Mal vollständig vorgelesen werden, so zu reduzieren, dass ein Mittelmaß zwischen Erläuterungsbedarf und Redundanz erzielt wird.

Soweit möglich Text statt Grafik

Anders sieht die Sache aus, wenn Absätze oder Textpassagen als Grafik dargestellt werden. Bindest du Grafiken in Texten ein, um bestimmte Effekte zu erzielen (z.B. Schriftart bei Logo oder Effekte wie Schattierung und 3D), ist der Alternativtext wie folgt einzubauen:

```
<P>Dies ist ein <IMG
src="beispiel.gif" alt="Beispiel">
dessen!</P>
```

Willst du aber längere Texte einscannen, z.B. einen Zeitungsartikel, und im HTML-Quelltext als Grafik platzieren, macht ein Alternativtext wenig Sinn, also muss der Text mit dem eben beschriebenen **longdesc**-Attribut oder dem "D"-Link in einer eigenen Datei dargestellt werden. Du könntest solche Texte aber gleich in HTML erstellen! Die WAI-3.1 empfiehlt, Informationen statt durch Grafik innerhalb der HTML-Elemente anzubieten, falls entsprechende Elemente bereit stehen. Mit CSS ist übrigens sehr viel möglich, wie du ab Seite 23 erfahren wirst.

Spezialfall: grafische Dokumente

Ein großes Problem für blinde und sehbehinderte Surfer stellen Dokumente in grafischem Format dar, etwa PDF-Dateien. Dank ihrer Unveränderlichkeit – die ja der Zweck der Bereitstellung in diesem Format ist – sind sie für Screen-Reader-Benutzer gar nicht und für Sehbehinderte kaum lesbar. Für Sehbehinderte, die mit den Standard-Farbeinstellungen ihrer Betriebssysteme nicht zurecht kommen und nur mit dunkler Hintergrundfarbe entspannt lesen können, ist der weiße Hintergrund solcher Dateien eine Barriere. Inzwischen hat Adobe diesen Misstand erkannt und daher im Acrobat Reader 5 einige Hilfen zur Beseitigung von Barrieren eingebaut; letztendlich bedeuten die grafischen Formate aber Barrieren.

Also sollte bei der Erstellung eines Dokuments eine reine Textausgabe zur Verfügung gestellt werden, die dann als Text- oder HTML-Datei neben die grafischen Datei ins WWW gestellt werden kann.

> Text-orientierte Dateien benötigen übrigens viel weniger Speicher und verkürzen Download-Zeiten ganz erheblich!

> Adobe bietet einen Service zur Konvertierung von PDF- in Textdateien an. Sendest du eine solche Datei an **pdf2txt@sun.trace.wisc.edu**, erhältst du automatisch dein Dokument in reinem Text zurück – allerdings ohne Grafiken.

Image-Maps

Server-seitig oder Client-seitig

Bei *Image-Maps* handelt es sich um Grafiken, die in verschiedene mit der Maus anklickbare vordefinierte Punkte oder Bereiche aufgeteilt sind. So kannst du unterschiedliche Funktionen in einer Grafik bereitstellen. Eine Image-Map wird meist mit dem **MAP**-Element erstellt und kann Server-seitig eingesetzt werden, wobei der Server eine Koordinate in deiner Grafik verarbeitet, oder aber Client-seitig, wobei der URL durch den Browser verarbeitet wird.

Server-seitige Image-Maps, die mit **ismap** definiert werden, erfordern einen Mausklick! Also solltest du grundsätzlich Client-seitige Image-Maps einsetzen, die mit **usemap** definiert werden, um den Zugang mit Screen-Readern zu erlauben. Da aber auch Client-seitige Image-Maps für ältere Screen-Reader unüberwindbare Barrieren sind, ist eine parallele Textseite unumgänglich, wenn deine Seiten barrierefrei gestalten willst. Eine Image-Map ist barrierefrei, wenn alle ihre Funktionen ohne Maus bedienbar und alle Links mit Alternativtexten belegt sind.

Die einzelnen Auswahlbereiche in einer Client-seitigen Image-Map sind von Screen-Readern nur über Alternativtexte der Bereiche zu erfassen. Da Image-Maps oft zur Navigation, Menüauswahl u.ä. benutzt werden, also eine der wichtigsten Funktionen ausfüllen, sind die Alternativtexte so auszuwählen, dass sie auch dann Sinn machen, wenn sie ohne den umschließenden Text oder die Grafikinhalte gelesen werden.

> Da Image-Maps für Navigationsfunktionen eingesetzt werden, solltest du den Abschnitt zu **accesskey** und **tabindex** auf Seite 61 lesen, der den Zugang von Webseiten über die Tastatur beschreibt.

Image-Maps, ob Client- oder Server-seitig, können als Absende-Button von Formularen eingesetzt werden. Dieser Aspekt von Image-Maps wird auf Seite 66 behandelt.

Server-seitige Image-Maps sind wegen der Notwendigkeit eines Mauszeigers grundsätzlich eine Barriere! Nach WAI-9.1 sind daher nur Client-seitige Image-Maps einzusetzen, außer wenn sie nicht die geometrischen Formen für die Auswahlbereiche deiner Grafik bieten! Alle Image-Maps benötigen eine Textalternative.

Alternativtexte in Image-Maps

Setzt du Image-Maps ein, solltest du parallel grundsätzlich eine Text-orientierte Zugangsmöglichkeit auf deiner Seite bieten. Benutzt du z.B. auf deiner Startseite ein Image-Map als Navigations-"Zentrum", solltest du mit einem gesonderten Link auf eine Textversion der Seite verweisen, wo die Inhalte nochmals als einfache Liste aufgeführt sind. Moderne Screen-Reader können nun aber bedingt mit Client-seitigen Image-Maps umgehen: definierst du Alternativtexte für die einzelnen Bereiche des Image-Maps, kann der Screen-Reader diese ebenfalls ausgeben.

Benutzt du das **AREA**-Element zur Darstellung deines Image-Maps, solltest du im **MAP**-Element für die Zuordnung der Alternativtexte für die einzelnen Bereiche das **alt**-Attribut verwenden:

```
<MAP name="navigation1">
  <AREA shape="rect"
coords="0,0,30,30"
href="produkte.htm" alt="Produkte">
  <AREA shape="rect"
coords="34,34,100,100"
href="bestellen.htm"
alt="Bestellformular">
</MAP>
...
<IMG src="auswahl.gif" alt="Image-
Map zur Navigation auf unserer
Website" usemap="#navigation1">
```

Du kannst auch mit dem **OBJECT**-Element ein Image-Map in deine Seite einsetzen und dabei umfassendere Informationen beifügen:

```
<MAP name="navigation1">
  <AREA shape="rect"
coords="0,0,30,30"
href="produkte.htm" alt="Produkte">
  <AREA shape="rect"
coords="34,34,100,100"
href="bestellen.htm"
alt="Bestellformular">
</MAP>
...
<OBJECT data="auswahl.gif"
type="image/gif"
usemap="#navigation1">
  Die Navigation auf unserer Website
wird über ein Image-Map gesteuert.
  Sie haben die folgende Auswahl:
  <A href="produkte.htm" title
="Produkte">Produkte</A>: sortiert
nach ... und ...
  <A href="bestellen.htm"
title="Bestellformular">Bestellformu
lar</A>: ...
</OBJECT>
```

Redundante alternative Links

Es ist "gute Programmierung", wenn du alle Regeln der Barrierefreiheit bei der Erstellung deiner Seiten berücksichtigst. Eine Client-seitige Image-Map muss bei korrektem Einsatz der Alternativtexte keine alternativen Links erhalten, um von Screen-Reader-Nutzern bedient zu werden – belegst du die Image-Map aber obendrein mit redundanten alternativen Links, ist der Zugang zu deiner Seite in jedem Fall garantiert.

Benutzt du anstatt des **AREA**-Elements das **A**-Element für die Definition der aktiven Bereiche in deiner Image-Map, kannst du die redundanten Links ohne Zusatzaufwand gleich bei der Gestaltung deiner Image-Map erstellen:

```
<OBJECT data="navigation1.gif"
type="image/gif"
usemap="#navigation2">
<MAP name="navigation2">
  <P>Navigation für die Website.
  [<A href="produkte.htm"
shape="rect" coords="0,0,30,30"
title="Produkte">Produkte</A>]
  [<A href="bestellen.htm"
shape="rect" coords="34,34,100,100"
title="Bestellformular">Bestellformu
lar</A>]
</MAP>
</OBJECT>
```

Siehst du genau hin, erkennst du, dass die Auswahlbereiche der Image-Map innerhalb des Objekts selbst definiert werden und weiter, dass die alternativen Links nur angezeigt werden,

wenn – aus welchem Grund auch immer – die Image-Map (NAVIGATION1) nicht angezeigt wird. Damit hast du zwei Fliegen mit einer Klappe geschlagen!

Die Links wurden von eckigen Klammern eingeschlossen. Das hat für Benutzer älterer Screen-Reader den Vorteil, dass die Links nicht alle nacheinander vorgelesen werden, quasi als einziger Link, sondern mit kurzen Unterbrechungen, und für den sehenden Surfer trennt dies optisch die einzelnen Links. Zu der Gestaltung beieinander liegenden Textlinks siehe Seite 60.

Alternative Links für Server-seitige Image-Maps

Nach WAI-1.2 ist die Bereitstellung alternativer Links zu einer Image-Map ein "Muss"-Kriterium für die Sicherstellung der Zugänglichkeit einer Webseite. Dabei sollte jeder aktive Bereich der Image-Map über eine gesonderte Textalternative verfügen. Bist du auf den Einsatz von Server-seitigen Image-Maps angewiesen, gibt es drei Techniken, um diese zugänglich zu machen:

1. Das OBJECT-Element

Wie im vorangegangen Beispiel für alternative Links in Client-seitigen Image-Maps fügst du im Körper eines **OBJECT**-Elements die entsprechenden Links (**A**-Elemente) bei.

2. Das IMG-Element

Benutzt du **IMG** zum Einfügen der Grafik, solltest du eine Liste der in der Image-Map angebotenen Links anbieten und z.B. über den Alternativtext entsprechend darauf hinweisen.

```
<A href="/cgi-bin/auswahl">
  <IMG src="willkommen.jpg"
alt="Willkommen! (Text-Links
folgen)" ismap>
</A>
<P>[<A href="auswahl1.htm">Auswahl
1</A>]
[<A href="auswahl2.htm">Auswahl
2</A>]
...
</P>
```

3. Zusätzliche Seite erstellen

Sind die ersten beiden Möglichkeiten nicht ausreichend oder möglich, ist eine ergänzende Seite als parallele "Nur-Text"-Seite notwendig. Hier sollten in einer Liste oder Tabelle alle Mausklicks der Image-Map erneut nachgebildet werden, so dass auch der Screen-Reader-Nutzer Zugang zur Information dieser Seite erhält. Image-Maps, die mit **ismap** eingebunden werden, sind nämlich nur mit der Maus und nicht mit der Tastatur bedienbar. Das bedeutet, dass sie für Blinde und Menschen mit Behinderungen der Bewegung nicht zugänglich sind.

Großzügige Flächen für den Mausklick

Hat jemand motorische Einschränkungen, hat er weniger Kontrolle über seine Bewegungen und somit auch über die Bewegung der Maus. Je nach Einschränkung kann das zu einer schleppenden oder einer ruckartigen Bewegung der Maus führen – als ob die Maus mit Boxerhandschuhen oder mit den Füßen bewegt würde. Also ist wichtig, dass der anklickbare Bereich einer Image-Map eine angemessene Größe hat. Kleine Flächen sind tendenziell für alle Benutzer lästig. Auch solltest du darauf achten, dass die Navigationselemente auf allen Seiten gleich aufgebaut werden.

Animationen

Für animated GIFs gelten dieselben Anforderungen wie für sonstige Grafiken. Nur solltest du daran denken, dass Menschen mit fotosensitiver Epilepsie Anfälle bereits ab vier "Blitzen" in der Sekunde bekommen können. Die gefährlichsten Situationen sind Lichtblitz-Raten von ungefähr 20 Blitzen in der Sekunde sowie schnelle Wechsel zwischen hell und dunkel (wie ein Stroboskop).

Auch Menschen mit Konzentrationsschwierigkeiten, etwa auf Grund von schweren Erkrankungen des Nervensystems, haben Probleme mit Blinken und ähnliche Multimedia-Effekten und können sich wegen solcher Animationen gar nicht auf die sonstigen Inhalte deiner Seiten konzentrieren.

Du solltest also bei der Erstellung animierter GIFs und anderer sich wiederholenden Elemente auf diesen sehr wichtigen Punkt achten. Ich komme auf Seite 28 wieder darauf zurück.

Grafische Aufzählungszeichen

Aufzählungen und Listen sind ein wichtiges Darstellungsmittel auf Internetseiten. Aufzählungszeichen, die bei nummerierten Listen und nicht sortierten Bullet-Listen über das **LI**-Element vom Browser erzeugt werden, werden vielfach durch eigene Grafiken ersetzt, um die Darstellung an das Layout anzupassen oder zu verbessern.

In der Tat haben diese Grafiken auch bei der Verwendung von Screen-Readern ihre Vorteile – die Standardzeichen von unsortierten Bullet-Listen werden nämlich vom Screen-Reader nicht unbedingt vorgelesen. Eine optisch eindeutige Gliederung geht akustisch verloren. Beim Einsatz sortierter Listen mit dem **OL**-Eleme **nt** werden die Zahlen bzw. Buchstaben dagegen von Screen-Readern wie normale Zeichen vorgelesen.

Wichtig für den hörenden Surfer ist, dass die Grafiken einen entsprechenden Alternativtext erhalten. Du solltest Grafiken als Aufzählungszeichen bei **DL**-Listen (*description list*) möglichst vermeiden und statt ihrer das dafür vorgesehene **UL**-Element verwenden. Solltest du diese Technik dennoch einsetzen, belege die Grafiken mit einem Alternativtext. Bei Aufzählungszeichen kann dies unkompliziert mit einem Bindestrich oder einem Sternchen gemacht werden:

```
<DL>
  <DD><IMG src="stern.jpg"
alt="*">Hans</DD>
  <DD><IMG src="stern.jpg"
alt="*">Peter</DD>
  <DD><IMG src="stern.jpg"
alt="*">Oliver</DD>
</DL>
```

Insbesondere ist auch hier auf die visuelle Kennzeichnung durch Aufzählungszeichen zu achten! Setzt du z.B. zur Unterscheidung von Listenelementen verschiedene Farben oder unterschiedliche Größen ein, solltest du im Alternativtext deren Bedeutung klarstellen, etwa:

```
<DL>
  <DD><IMG src="kugel_blau.jpg"
alt="Neu:">PARC</DD>
  <DD><IMG src="kugel_gelb.jpg"
alt="Vorherige Version:">PARC-
Entwurf</DD>
</DL>
```

Persönlichen Grafiken für Bullet-Listen (**UL**-Element) werden am besten mit Cascading Style-Sheets (CSS) gestaltet (Seite 69). Leider bietet der mit CSS erzielte Austausch der Aufzählungszeichen keine Möglichkeit für alternative Texte. Was tun? Solche Grafiken kannst du mit einem einleuchtenden Namen benennen, etwa KUGEL_BLAU.PNG oder DOPPELPFEIL.JPG. Gibt es keinen Alternativtext, geben manche Screen-Reader den Dateinamen der Grafik aus. Du bindest die Alternativtexte in die Liste ein, indem du aus einer normalen Liste:

```
<UL>
... <LI>Ein Listenpunkt</LI> ...
</UL>
```

eine etwas aufwendigere Liste machst:

```
<UL>
... <IMG src="kugel.jpg" alt="*">Ein
Listenpunkt<BR /> ...
</UL>
```

Dabei wird auf das **LI**-Element verzichtet. Die Einrückung erfolgt über **UL**. Diese Möglichkeit funktioniert aber nur, wenn du kurze Begriffe u.ä. aufzählen willst – bei Zeilenumbruch beginnt die zweite Zeile unterhalb der Grafik und du musst für den gewünschten Effekt Tabellen einsetzen.

"Spacers"

Oft werden sogenannte *Spacers* oder *Dummies*, d.h. transparente Grafiken, zum Schaffen von Abständen zwischen Textteilen eingesetzt. Da sie unsichtbar sind, erhalten sie keinen Alternativtext. Für Blinde sind Grafiken ohne Textangabe ein Rätsel – sie wissen nicht, ob sich ein informatives Bild dahinter verbirgt. Dummies unterbrechen auch den Lesefluss, da sie von Screen-Readern ausgewertet werden. Kannst du nicht auf diese Dummies verzichten (z.B. indem Style-Angaben verwendet werden), müssen die Grafiken klare Namen tragen, etwa LAYOUT.GIF oder TRANSPARENT.GIF statt PIX_T.GIF, damit die Zweitinformation des Dateinamens klar wird.

HTML ist keine Layout-Sprache; ihre Aufgabe ist vielmehr die Organisation und Strukturierung von Informationen. Für das Layout hat das W3C die Einbindung von Style-Sheets standardisiert. Eben dieses Beispiel mit den "Dummies" lässt sich elegant mit Style-Sheets lösen, was wir uns ab Seite 51 genauer ansehen wollen.

Verlasse dich nicht auf Farben!

Die Gestaltung deiner Texte und Grafiken mit Farben soll das optische Erscheinungsbild deiner Site aufbessern. Mit gezielter Farbauswahl kannst du sogar auf relativ einfache Weise optische Orientierungen geben. Verlässt du dich aber allein auf Farben, um bestimmte Texte hervorzuheben, Listenpunkte zu differenzieren, zwischen aktiven und passiven Elementen zu unterscheiden und dergleichen mehr, können Menschen mit Sehschwächen und -behinderungen auf von dir nicht beabsichtigte Barrieren stoßen. Kontraste spielen beim Design ebenfalls eine Rolle – je geringer der Kontrast, desto weniger Menschen können die Inhalte lesen.

Die Definition von Farbe und Kontrast ist insbesondere bei der Gestaltung von Grafiken wie z.B. Navigations-Buttons wichtig. Während Textfarbe und Seitenhintergrund im Extremfall vom sehbehinderten User überschrieben werden können, sind die Farben von Grafiken unveränderlich.

Unterscheidungsmerkmal Farbe

Farben als alleiniges Unterscheidungsmerkmal sind vor allem für folgende Gruppierungen nicht oder nur bedingt wahrnehmbar:

- Menschen mit leichten Sehstörungen, die bestimmte Farben nicht unterscheiden können. Rot-Grün-Kombinationen sind besonders ungünstig, da ein beachtlicher Teil aller Menschen eine Rot-Grün-Sehschwäche haben – und viele das nicht einmal wissen.
- Sehbehinderte Menschen, die z.B. stark blendempfindlich sind und deswegen ihre System- und Browserfarben umgestellt haben. Nach diesem Eingriff werden Text- und Hintergrundfarben auf Webseiten völlig ignoriert!
- Screen-Reader-Nutzer und andere Nutzer nicht-visueller Ausgabeprogramme. Textorientierte Browser und Audio-Browser geben in der Regel nur den Text und logische Formatierungen aus, nicht aber stilistische.
- Nutzer von Monochrombildschirmen oder Grafikkarten mit kleiner Farbpalette.

Auswahl der Farben

Bei der Auswahl von Farben ist der Kontrast zwischen Information (Text, Symbol, ...) und Hintergrund sehr wichtig, wenn deine Seiten gut lesbar sein sollen. Es leuchtet den meisten ein, dass rote Schrift auf grünem Hintergrund schwieriger zu erkennen ist als gelb auf blau oder schwarz auf weiß. Überhaupt ist die Kombination rot/grün sehr kritisch, weil Millionen von Menschen eine Rot-Grün-Sehschwäche haben.

Auch wenn man den Browser so einstellen kann, dass Farben der Website ignoriert werden, sind viele Sehbehinderte nicht darauf angewiesen. Jedoch trifft man immer wieder auf Seiten, deren Farbkombinationen für viele unglücklich sind – und auf denen daher die Freude am Lesen sinkt.

Es gibt Leute, die die Anwendung von Farben gänzlich verpönen wollen; das halte ich aber für Unsinn. Es sollte genügen, wenn du dir klare Gedanken über die Lesbarkeit machst. Dazu gehört die kontrastreiche Gestaltung von Informationen möglichst frei von Verzierungen wie kunstvollem Text oder Hintergrundgrafik.

> Als WWW-konforme, d.h. in allen Browsern darstellbare Farben sind nur die anzusehen, die ein bestimmtes Standard-Schema einhalten. Die viel verbreitete Methode, mit Namen (z.B. **`dimgrey`** für Hellgrau) Farben zu bestimmen, ist keinesfalls WWW-konform! Alle Bildschirm-Farben werden aus drei hexadezimale Zeichenpaaren kombiniert. Nur Kombinationen der Zeichenpaare 00, 33, 66, 99, CC und FF sind für alle Browser darstellbar (Bsp.: **`#00FFCC`**). Außerdem gibt es neben diesen 216 Farben noch 40 Farben, die sich nicht mit den genannten Zeichenpaaren darstellen lassen.

Grafiken etwa im GIF-Format können bekanntlich mit der Farbe "Transparent" gefüllt werden, was für stark blendempfindliche Menschen eine Barriere bedeuten kann, wenn sie mit invertierten Farben surfen. Stell dir vor, du hast ein Logo oder ein Symbol nur in schwarz dargestellt und benutzt als Grafikhintergrund "Transparent". Das, was du darstellen willst, erscheint schwarz ... richtig: auf schwarz! Du solltest also grundsätzlich sowohl Farbe für den Vordergrund wie auch für den Hintergrund wählen. Stellst du den Hintergrund weiß bzw. mit der Farbe des Seitenhintergrunds dar, merken die meisten Besucher deiner Seite das gar nicht.

Informationen per Farbe immer ergänzen

Nach WAI-2.1 müssen sämtliche Informationen, die nur durch ihre Farbgebung übermittelt werden sollen – ob Text oder Grafik – auch in anderer Form dargestellt werden. Für Grafiken hat dieses Kriterium die Priorität 1! Wie sich dies in einer Aufzählungsliste gestalten lässt, sahst du Seite 15, wo die unterschiedlichen Grafiken mit jeweiligen Alternativtexten belegt wurden. Du kannst auch weitere Zusatzinformationen einbringen, etwa einen Rahmen:

```
<UL>
  <IMG src="kugel_blau.png"
alt="NEU:" border="2">
Diskussionsforum<BR />
  <IMG src="kugel_rot.png"
alt="ALT:"> Newsletter abonnieren<BR
/>
  <IMG src="kugel_rot.png"
alt="ALT:"> Kontaktadressen
</UL>
```

Mit dem **border**-Attribut setzt du einen Rahmen um eine Grafik, der Sehbehinderten die Wahrnehmung von Hervorhebungen erleichtert, die sonst nur durch Farbwahrnehmung zu erkennen sind,.

Es geht hier um Farbe als Unterscheidungsmerkmal – nicht um die Farben des Gesamtlayouts! Es geht z.B. darum, Aufforderungen wie "Wählen Sie einen Eintrag aus der roten Liste!" zu umgehen, weil manche Leute Rot gar nicht erkennen. Die "rote Liste" muss mit zusätzlichen Schriftformatierungen wie fett, kursiv, Schriftgröße, Schriftart versehen werden, mit zusätzlichem Text, etwa einer Beschreibung oder Überschrift, und/oder mit anderen Hervorhebungen. Natürlich macht es Sinn, jede Art von Hervorhebung mit weiteren Merkmalen zu versehen, denn auch die Schriftart kann vom Browser übergangen werden.

Im folgenden Beispiel setze ich Cascading Style-Sheets ein, um Texte anders darzustellen. Es gibt viele Möglichkeiten, Text hervorzuheben – grundsätzlich sind aber die HTML-Elemente für den Aufbau deiner Seite vorgesehen und Style-Sheets für die Formatierung!

Text ergänzend formatieren

Stell dir vor, dass man Interesse für den Freizeitpark "Sugar Mountain" auf dessen Webseite anmelden soll. Der Webmaster hat in der Eile nur folgenden Quelltext als Ergänzung zu den persönlichen Angaben (Name, Email und Anschrift) "hingeklotzt":

```
<P>Bitte mindestens eins aus der
roten Liste wählen!</P>
<FORM action="senden.asp"
method="post">
  <INPUT type="radio" name="rot"
value="ja"><SPAN
style="color:#FF0000">Ich bin über
18</SPAN><BR />
  <INPUT type="radio" name="rot"
value="nein"><SPAN
style="color:#FF0000">Ich bin nicht
über 18</SPAN><BR />
  <INPUT type="radio" name="normal"
value="ja">Mehr Informationen<BR />
  <INPUT type="radio" name="normal"
value="ja">Keine Informationen<BR />
  <INPUT type="submit" name="weiter"
value="Abschicken">
</FORM>
```

> Das **SPAN**-Element dient der Zuweisung von Style-Angaben und hat sonst keine Auswirkung auf den Text. Neben dem **style**-Attribut sind auch **class**- und **id**-Attribute erlaubt.

Wie wir bald sehen werden, ist – abgesehen von der Tatsache, dass der Text viel zu knapp ist – die Farbe möglicherweise gar nicht zu erkennen. Was tun?

Zunächst sollte der Webmaster eine ergänzende Formatierung einführen. Er könnte z.B. den roten Text kursiv gestalten oder ein Ausrufezeichen, eine andere Formatierung oder ähnliches einbringen. Der einleitende Text würde dann etwa so heißen: "Bitte mindestens eins aus der roten Liste (kursiv) wählen!". Die **SPAN**-Elemente würden ergänzt:

```
<SPAN style="color:#ff0000;
background:#ffffff;font-
style:italic"> ... </SPAN>
```

Du solltest für Farben grundsätzlich Zahlenangaben wie **color:#ff0000** und **color:rgb(100%, 0%,0%)** verwenden statt der Wortbezeichnung **color:red**. Für Texte benutzt du verschiedene Style-Angaben: **color** (Vordergrund bzw. Text), **background-color** (Hintergrund), **border-color** und **outline-color** (Rahmen)

Text in einem Kasten setzen

Willst du einen Text hervorheben, setzt du ihn in einen Kasten. Fas erreichst du hier, indem du im Kopfbereich oder in einer CSS-Datei eine CSS-Klasse **kasten** definierst:

```
<STYLE type="text/css"><!--
.kasten
{color:#FF0000;background:#FFFF00;bo
rder:2pt;border-style:groove;}
--></STYLE>
```

Style-Angaben – auch wenn sie im Kopfbereich einer WWW-Seite vorkommen – sollten immer mit **<!--**und **-->** auskommentiert werden, da sie andernfalls von Browsern, die CSS nicht interpretieren, vollständig angezeigt werden.

Die Klasse **kasten** definiert eine rote Schrift mit gelbem Hintergrund sowie einen 2 Punkt dicken Rahmen für das mit dieser Klasse belegte Element. Enthält unsere rote Liste zwei Elemente, die in ein und demselben Kasten hervorgehoben werden sollen, benötigen wir ein HTML-Element, das die beiden **INPUT**-Elemente umgibt; dazu dient das **DIV**-Element, das einen unsichtbaren Kasten um alle Elemente zwischen den **DIV**-Begrenzungen erzeugt. Die Punkte aus unserer "roten Liste" erhalten so eine Umgebung, die mit einer Klasse belegt werden kann. Dasselbe lässt sich mit dem **SPAN**-Element machen.

```
<FORM action="senden.asp" method="post">
  <DIV class="kasten" style="width:200pt">
  <INPUT type="radio" name="rot" value="ja"><SPAN
style="color:#FF0000;font-style:italic">Ich bin über 18</SPAN><BR />
  <INPUT type="radio" name="rot" value="nein"><SPAN
style="color:#FF0000;font-style:italic">Ich bin nicht über 18</SPAN><BR />
  </DIV>
...
</FORM>
```

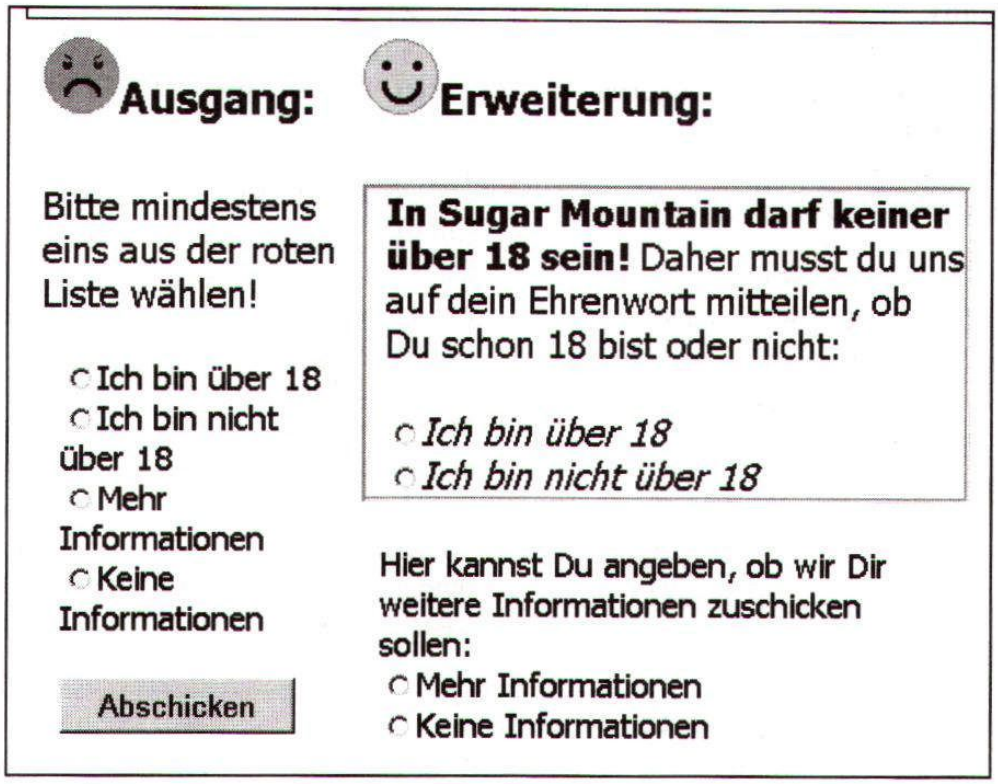
Mit dem **style**-Attribut kannst du weitere CSS-Angaben hinzufügen. Hier beschränken wir die Breite des **DIV**-Elements auf 200 Pixels.

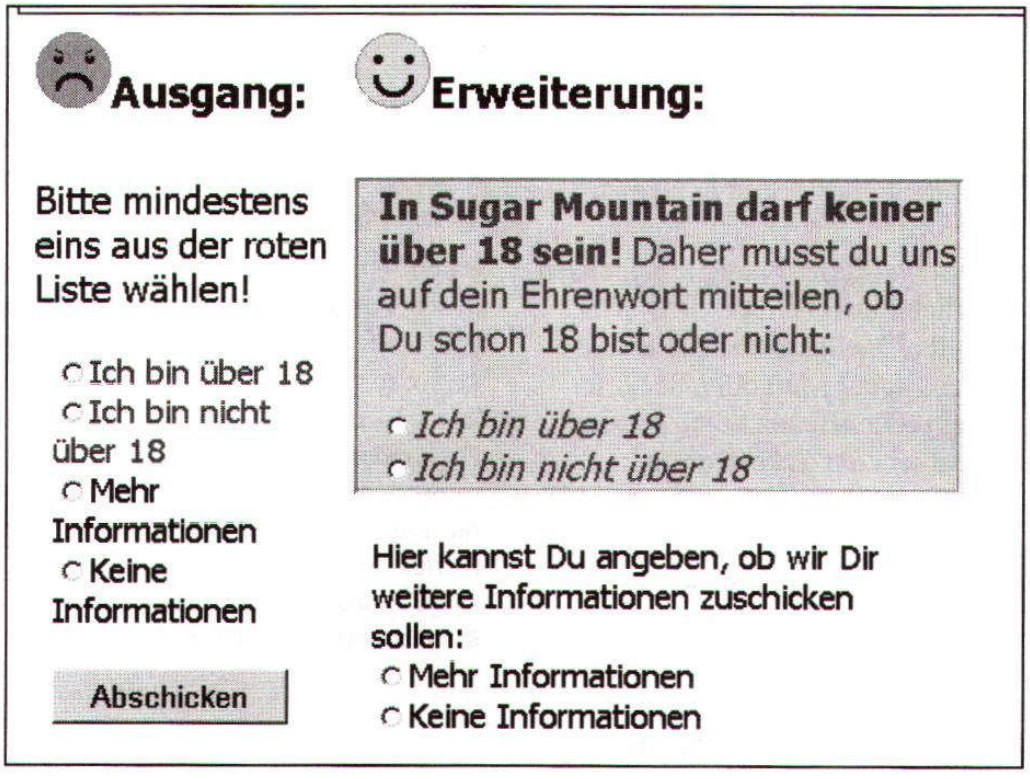
Diese Methode wird häufig für Navigationsleisten und andere Hervorhebungen benutzt. Das W3C verwendet sie z.B. für Quelltext-Beispiele. Die Notwendigkeit zusätzlicher Methoden der Informationsvermittlung rückt offensichtlich immer mehr in das Bewusstsein der Web-Designer.

Vielleicht denkst du, dass sich dies mit Tabellen und ihrem **border**-Attribut einfacher machen ließe – was sicher richtig ist, falls du mit einem grafischen HTML-Editor arbeitest. Wie ich in dieser Broschüre immer wieder betone, sollten HTML-Elemente aber zur Strukturierung der Seiteninhalte benutzt werden und möglichst nicht für Layout und Hervorhebungen. Wann immer möglich, solltest du für Textformatierungen Style-Sheets einsetzen.

Sehen wir uns nun an, was ein Browser aus dieser Vorgabe erstellt – im vorliegenden Fall handelt es sich um den Microsoft Internet Explorer für Windows in der Version 5.5.

Die beiden Screenshot sind hier natürlich nur schwarz-weiß wiedergegeben – du kannst die farbigen Originale aber von der Webseite des Heftes downloaden. Sie heißen **sugarmountainmitfarbe.bmp** bzw. **sugarmountainohnefarbe.bmp**


Ausgang:
Erweiterung:
Bitte mindestens eins aus der roten Liste wählen!
Ich bin über 18
Ich bin nicht über 18
Mehr Informationen
Keine Informationen
Abschicken
In Sugar Mountain darf keiner über 18 sein! Daher musst du uns auf dein Ehrenwort mitteilen, ob Du schon 18 bist oder nicht:
Ich bin über 18
Ich bin nicht über 18
Hier kannst Du angeben, ob wir Dir weitere Informationen zuschicken sollen:
Mehr Informationen
Keine Informationen


Ausgang:
Erweiterung:
Bitte mindestens eins aus der roten Liste wählen!
Ich bin über 18
Ich bin nicht über 18
Mehr Informationen
Keine Informationen
Abschicken
In Sugar Mountain darf keiner über 18 sein! Daher musst du uns auf dein Ehrenwort mitteilen, ob Du schon 18 bist oder nicht:
Ich bin über 18
Ich bin nicht über 18
Hier kannst Du angeben, ob wir Dir weitere Informationen zuschicken sollen:
Mehr Informationen
Keine Informationen

"Stopper" für Screen-Reader

Geht aus dem Kontext nicht deutlich hervor, welche Elemente der Liste rot sind und welche nicht, wäre ein Screen-Reader-Nutzer nun zum Raten gezwungen, was unser Beispiel betrifft. Der Screen-Reader liest normalerweise keine Formatierungen vor – es sei denn der Anwender fragt sie über spezielle Tastenbefehle ab oder schaut im Quelltext nach. Wir wollen darum, ähnlich wie beim "D"-Link, einen "Stopper" für das Ende unserer "roten Liste" einbauen – für Screen-Reader, aber auch für andere Browser, die keine Style-Sheets interpretieren. Wir ergänzen das Style-Sheet um die neue Zeile:

```
.verstecken {display:none}
```

Ein HTML-Element, das mit dieser Klasse belegt ist, wird in CSS2-fähigen Browsern nicht dargestellt. Am Ende unserer "roten Liste" fügen wir folgenden Quelltext ein:

```
<SPAN class="verstecken">Ende der
roten Liste</SPAN>
```

> Das Verstecken von Text erfolgt nur, wenn CSS2 vom Browser interpretiert wird. Versteht der Browser CSS2 nicht, werden keine der im Beispiel angebrachten Formatierungen angezeigt, also auch nicht die rote Farbe!

Aus dem Kontext verständlich machen

Auch mit zusätzlichem Text oder einem Hinweis bzw. Zeichen kannst du ergänzende Merkmale für bestimmte Texte einfügen. Bei Formularen werden Felder, die ausgefüllt werden müssen, meist mit einem Sternchen (*) versehen. Aufmerksamkeit wird aber auch mit kleinen Grafiken oder Überschriften erzielt.

Um in unserem Beispiel zu bleiben: die rote Liste kann schlicht mit folgendem Satz eingeleitet werden: "In Sugar Mountain darf keiner über 18 sein. Daher musst du uns auf dein Ehrenwort mitteilen, ob Du schon 18 bist oder nicht." Auch alle anderen Bereiche können mit solchen Texten oder Überschriften versehen werden. Du kannst natürlich auch den "Stopper" benutzen, indem du zu Beginn der "roten Liste" folgende Zeile einfügst:

```
<SPAN class="verstecken">Beginn
der roten Liste</SPAN>
```

Die beiden Screen-Shots stellen ein und dieselbe Seite mit verschiedenen Einstellungen dar. Wie du siehst, ist die Farbe der "roten Liste" nur mit den Standard-Einstellungen erkennbar, nicht aber in der Darstellung mit invertierten Farben.

Was auch immer du tust, um deine Seite barrierefrei zu gestalten – du musst es mit den gängigen Browsern testen. Schalte die Nutzung von Style-Sheets ein und aus, um dir einen Eindruck davon zu verschaffen, wie Browser ohne CSS deine Seite ausgeben.

Kombinationen aus Vordergrund- und Hintergrundfarbe

Auf ausreichenden Kontrast solltest du auch bei der Wahl von Vorder- und Hintergrundfarben für deine Seitenelemente achten. Prinzipiell ist zu sagen, dass jede Hintergrundfarbe die Gefahr birgt, Texte unleserlich zu machen.

Nachfolgend einige Farbenkombinationen, die besonders gut zu lesen sind:

- Schwarz und Weiß
- Weiß und Rot
- Weiß und Schwarz
- Blau und Weiß
- Gelb und Blau

Folgende Farbkombinationen sind schlecht zu sehen:

- Orange und Grün
- Gelb und Weiß
- Orange und Rot
- Rot und Blau
- Orange und Gelb
- Blau und Orange

Farben und Kontraste testen

Willst du die Güte deines Layouts einschätzen, soweit es Farben und Kontraste betrifft, tust du das mit Hilfe der drei folgenden Prüfungen.

Farbinvertierung

Normalerweise kann ein Benutzer selber die Darstellungsfarben in seinem Standard-Browser bestimmen, was den Vorrang hat gegenüber den in den Webseiten definierten Farben. Im Netscape-Browser wählst du dazu BEARBEITEN|EINSTELLUNGEN und dann unter FARBEN weiß für die Textfarbe, schwarz für die Hintergrundfarbe und je nachdem gelb für die Linkfarben; sichere dich, dass STATT DOKUMENTFARBEN IMMER NETSCAPE-FARBEN BENUTZEN aktiviert ist. Im MSIE5.x wählst du EXTRAS|INTERNETOPTIONEN, dann unter ALLGEMEIN den Punkt FARBEN und hier dieselben Farben wie eben für den Netscape-Browser beschrieben. Danach wählst du wieder unter ALLGEMEIN die EINGABEHILFEN und sicherst dich, dass FARBANGABEN AUF WEBSEITE IGNORIEREN angeklickt ist. Nun solltest du deine Seiten ohne Informationsverlust lesen können.

Schwarz/weiss-Darstellung

Willst du wissen, ob deine WWW-Seiten auch ohne Farben lesbar sind, setzt du entweder einen Monochrombildschirm ein oder wählst in den Einstellungen deines Browsers weiße Textfarbe, schwarze Hintergrundfarbe und Graustufen als Linkfarben. Dann unterdrückst du die Farben des Webs wie oben beschrieben. Auch mit diesem Test stellst du fest, ob deine Texte barrierefrei gestaltet sind.

Kontraste

Willst du auch deine Grafiken auf Lesbarkeit testen, druckst du deine Seiten im schwarz/weiß-Modus aus, wodurch alle Farben in Graustufen erscheinen. Wenn du die Seite fotokopierst, die Kopie wieder fotokopierst und dies zwei oder drei Mal wiederholst, wirst du schnell erkennen, welche Bereiche auf deinen Seiten gut lesbar sind und welche Bereiche einer Kontrastverbesserung bedürfen.

Möchtest du die Farben einer Grafik oder gar deiner gesamtem Website prüfen, bietet **vischeck.com** kostenlose online- und herunterladbare Tools dafür an.

Mehrsprachige Seiten

Die meisten Websites werden in einer und derselben Sprache verfasst. Setzt du jedoch mehr als eine Sprache auf deinen Seiten ein, z.B. auf einer primär deutschsprachigen Seite Zitate oder sonstige Abschnitte in englischer Sprache, musst du dich im Hinblick auf Screen-Reader-Anwender unbedingt an die nachfolgenden Ausführungen zur Mehrsprachigkeit halten.

Screen-Reader arbeiten mit Sprachtabellen, d.h. sie erfassen einen Text und geben ihn entsprechend der voreingestellten Aussprache an die Soundkarte weiter. Liest ein blinder Surfer also vor allem deutschsprachige WWW-Seiten und ruft dann eine englischsprachige Seite auf, muss er die Aussprache seines Screen-Readers umstellen, damit die Seite verständlich vorgelesen wird.

Dieses Problem wird meist hingenommen. Die Kennzeichnung der Hauptsprache einer Seite hat die WAI lediglich mit der Priorität 3 belegt. Wird aber innerhalb einer Seite die Sprache gewechselt, wird das Lesen problematisch, da der Screen-Reader-Anwender wiederholt die Konfiguration des Screen-Readers ändern muss. Das wird ihm erspart, wenn die Sprachänderung auf der Seite angegeben ist und also vom Screen-Reader automatisch erkannt wird. Die Kennzeichnung von Sprachwechsel hat die Priorität 1.

Kennzeichne die Hauptsprache deiner HTMLDateien

Die Sprache einer Webseite wird mit dem Universalattribut **lang** (*language* = Sprache) zugeordnet. Dieses Attribut kannst du für fast jedes Element einsetzen. Allerdings reicht es bei einer einsprachigeb Seite völlig, wenn das HTML-Element dieses Attribut erhält. Das **lang**-Attribut wird an alle anderen Elemente vererbt.

```
<HTML lang="de">
   ... Rest eines HTML-Dokuments in
deutscher Sprache ...
</HTML>
```

Die Abkürzungen für die einzelnen Sprachen entsprechen den im Internet üblichen Länderkürzeln, etwa **en** = englisch, **fr** = französisch oder **dk** = dänisch. Weiteres findest du hier:

```
www.w3.org/TR/html4/references.
html#ref-RFC1766
```

Wechsel der Sprache innerhalb eines Dokuments

Kennzeichnest du einen Wechsel der Sprache auf deinen Seiten mit dem **lang**-Attribut, sind Screen-Reader in der Lage, die Aussprache anzupassen und somit die Seite für den mehrsprachigen Benutzer verständlich zu machen. Es reicht nicht, die Hauptsprache einer Seite zu kennzeichnen – jeder fremdsprachiger Text sollte ebenfalls gekennzeichnet werden.

```
<HTML lang="de">
  <HEAD> ... </HEAD>
  <BODY>
    ...
    Und es wurde still im Saal. Dann
sagte er: <SPAN lang="en">&quot;I
want to make money, not
music!&quot;. Damit war klar, dass
der Musiksparte des Unternehmens
keine Zukunft mehr eingeräumt wird
... </SPAN>
  </BODY>
</HTML>
```

Die Sprachzuweisung hilft übrigens nicht nur den Nutzern von Screen-Readern, sondern besonders auch Suchmaschinen bei der Indexierung von Schlüsselwörtern. Außerdem hilft die Angabe der Sprache bei vielen weiteren Aspekten der Darstellung, etwa bei der länderspezifischen Anzeige von Anführungs- und anderen Zeichen oder bei Rechtschreibprüfungen und maschinellen Übersetzungen.

Als ich seinerzeit das erste Mal einen Screen-Reader testete, hatte ich die Ausgabe-Sprache von amerikanisch auf deutsch umgestellt. Allerdings hatte ich keine Spracheinstellung für die Funktionen gefunden. Beim Vorlesen von Webseiten las die Software dann fleißig Grafiken mit ihren Alternativtexten vor usw. Immer wieder sprach sie auch "Wie sie Tee trinkt" ... jedenfalls verstand ich das so. Nach einer Weile bemerkte ich, dass die Sprachausgabe den Teetrinkerin-Hinweis eher mit chinesischem Einschlag sprach, also "Wie sie Tee tlinkt". Später erst stellte ich fest, dass die Sprachausgabe bereits besuchte Links ankündigte und die englische Bezeichnung "visited link" in deutsch vorlas.

Grundsätzliches zu Style-Sheets

Die Verwendung von Cascading Style-Sheets (CSS) zur Formatierung von Text gehört zum Handwerk des Webdesign. HTML wurde nicht etwa entwickelt, um Texte zu formatieren und gestalten, sondern vielmehr um sie zu organisieren. Mit CSS, vergleichbar mit den Formatvorlagen eines Textverarbeitungsprogramms, wurde eine Formatierungs- oder Präsentations-Ebene für Web-Auftritte eingeführt, die das Prinzip der Trennung von Inhalt und Layout unterstützt, im übrigen auch das Grundprinzip der Zukunftsweisenden *eXtended Markup Language* (XML). Letzten Endes soll diese Trennung bezwecken, dass die angebotenen Informationen, z.B. auf einer Website, in der vom Benutzer gewünschten Form präsentiert werden. Aus Sicht der Barrierefreiheit, insbesondere für Sehbehinderte, bedeutet die Möglichkeit der individuellen Darstellungsform eine deutliche Erleichterung der Zugänglichkeit. In diesem Abschnitt stelle ich das Prinzip der Trennung von Inhalt (HTML) und Layout (CSS) vor. Auf Seite 51 findest du ausführliche Informationen und einige Techniken zu CSS.

Die derzeitige Praxis

Viele Webgestalter verwenden zur Formatierung von Texten die Elemente **BASEFONT** und **FONT**, die Textgröße und -farbe sowie Schriftart definieren. Seit 1997 gehören diese beiden Elemente nicht mehr zum HTML-Standard – und das W3C empfiehlt ausdrücklich, sie nicht zu verwenden, auch wenn HTML-Editoren und Browser sie weiterhin unterstützen. HTML4.0 enthält weitere Elemente, die zur Textformatierung eingesetzt werden, wie etwa die **B**-, **I**- und **H1**- bis **H6**-Elemente. Auch die Elemente **BIG** und **SMALL** bleiben weiterhin ein Teil der W3C-Standards. Gerade die **H1**- bis **H6**-Überschriften sollten aber ausschließlich dazu verwendet werden, deinen Dokumenten eine logische Struktur zu geben – manche Browser erkennen die Überschriften nämlich als Marken und lassen den Benutzer somit schneller durch längere Texte navigieren.

Anstatt des **FONT**-Elements kannst du genauso gut die vielseitigen CSS-Eigenschaften benutzen: **font**, **font-family**, **font-size**, **font-size-adjust**, **font-stretch**, **font-style**, **font-variant** und **font-weight**.

Vorteile von CSS

Definierst du eine CSS-Formatvorlage, kannst du Elemente von vorneherein gestalten (z.B. Schriftart); du kannst aber auch sogenannte Klassen und IDs definieren, die mit deinen speziellen Formatierungen bei Bedarf einzelnen Elementen auf deiner Seite zugewiesen werden können. CSS lässt sich zwar überall in deinen HTML-Seiten definieren, letztendlich ist es nun aber ebenso wirkungsvoll, wenn sie im Kopfbereich der HTML-Seiten mit dem **STYLE**-Element oder aus einer externen Datei mit dem **LINK**-Element eingebunden werden. Bindest du eine externe Datei in allen Seiten ein, kannst du später mit einem Handgriff das Layout aller Seiten ändern. Dabei ist CSS deutlich leistungsfähiger als HTML. So kannst du z.B. mit CSS den Einzug der ersten Zeile eines Absatzes vordefinieren (wie in einem Buch) oder Abstände vor und nach Überschriften verändern, was in HTML nicht möglich ist. Die Einbindung einer externen CSS-Datei hilft dir, WAI-14.3 zu erfüllen, und erfolgt über folgende Zeile im **HEAD**-Bereich einer Web-Seite:

```
<LINK rel="stylesheet"
href="dateiname.css"
type="text/css">
```

Aus Sicht der Barrierefreiheit sind die beiden wesentlichen Vorteile von Style-Sheets die Textorientierung deiner Inhalte und die Browser-übergreifende, einheitliche Darstellung der Inhalte. Dies erfordert natürlich die Trennung von Inhalten, die mit strukturellen Elementen – also HTML – gekennzeichnet werden, und Gestaltung, die mit CSS für die visuelle Darbietung vorgegeben werden. Gestaltest du Inhalte mit CSS, erhält der Browser und jede andere Zugangssoftware wie z.B. Suchmaschinen zunächst nur den strukturierten Text. Handelt es sich bei der Zugangssoftware um ein CSS-fähigen Browser, werden dann diese CSS-Angaben geladen und zugewiesen. Solche Mühen werden übrigens von Suchmaschinen "belohnt", die Texte in Strukturelemente höher bewerten. Ganz abgesehen davon, kannst du die Dateigrößen auf deiner Website auf bis zu einem Drittel verkleinern, wie du auf folgender Adresse nachlesen kannst:

www.w3.org/Protocols/HTTP/Performance/Pipeline

Moderne Browser, die CSS und HTML4 korrekt interpretieren, ergeben Pixel-genaue, identische Ergebnisse. Allerdings gehören Microsoft's Internet Explorer 5.0x (für Windows-PCs) sowie alle 4er Browser von Microsoft und Netscape sowie die 3.5x-Versionen von Opera nicht dazu. Immer noch mit kleineren Fehlern und daher der Notwendigkeit, für die einzelnen Browser kleinere Browser-abhängige Anpassungen vorzunehmen, sind Microsoft Internet Explorer ab der Version 5.5 (für Windows-PCs), Netscape 6 und Opera 5 durchaus CSS-fähig:

- Für den PC bietet Microsoft's Internet Explorer ab der Version 5.5 ordentliche Unterstützung für HTML4 und CSS1 sowie weitere wichtige W3C-Standards. Der MSIE6 bietet sogar erweiterte Unterstützung. Interessanterweise ist der "perfekte" Browser der Microsoft Internet Explorer 5.0 für den Mac – er hält sich unmittelbar an alle W3C-Richtlinien.
- Netscape 6 erfüllt fünf der wichtigen W3C-Standards, einschließlich XML und dem Document Object Model (DOM), wodurch auch CSS unterstützt wird. Die Netscape-Browser laufen auf AIX, Linux, Windows, MacOS, OpenVMS, HPUX und FreeBSD.
- Opera 5 unterstützt viele der wichtigen WWW-Standards. Der Chef-Entwickler dieser Zugangssoftware war auch der Chef-Autor des CSS1-Standards der W3C. Opera unterstützt Windows, Linux, MacOS, OS/2, EPOC und BeOS.
- Konqueror ist einer der modernsten Browser für UNIX, Linux und IBM-Systeme mit hervorragender Unterstützung für viele W3C-Standards wie HTML4, CSS1, ECMAScript und den DOM sowie zum Teil auch XML und CSS2.

Neben der flexiblen Formatierung von Text kann CSS auch für die alternative Ausgabe von WWW-Seiten an verschiedenen Geräten (Bildschirm, Sound, Braille-Zeile, ...) oder zum Positionieren von Elementen eingesetzt werden.

Alle Größen- und Positionierungsangaben können Pixel-genau gemacht werden; die Verwendung relativer Werte ist aber eine sehr benutzerfreundliche Webgestaltung, die das Layout an die persönlichen Bildschirmeinstellungen deines Besuchers anpasst. Anstatt mit Punkten (pt) oder Millimeter (mm) kannst du Werte in Prozent (z.B. für Spaltenbreiten in Tabellen) und in em-Einheiten (z.B. für Schriftgrößen) angeben. So können deine Websites genauso gut von Sehbehinderten gelesen werden, die vielleicht auf eine sehr geringe Bildschirmauflösung und größere Schrift angewiesen sind.

Ist deine Seite ohne das Style-Sheet lesbar?

Wie immer du deine Seite mit CSS gestaltest, du wirst oft erleben, dass Browser die Style-Sheets gar nicht oder nicht richtig unterstützen bzw. dass der Benutzer sie ausschaltet. Hier kommt es auf die Organisation deines Quellcodes an! Unterstützt ein Browser z.B. Textpositionierungen mit dem **`DIV`**-Element nicht, werden die CSS-Angaben zur Positionierung ignoriert, und Textblöcke werden unter- statt z.B. nebeneinander dargestellt. Achtest du also im Quelltext deiner Seiten darauf, dass Informationen, die am Bildschirm zusammengehörig dargestellt werden, auch im Quelltext gruppiert sind, kannst du in der Regel davon ausgehen, dass deine Website auch ohne Style-Sheet-Unterstützung nachvollzogen werden kann (WAI-6.1). Testen kannst du die Organisation deines Quellcodes mit älteren Browsern, etwa Netscape Navigator 4.7, der Positionierungsangaben nur bedingt unterstützt.

Mit modernen Browsern wie Microsoft Internet Explorer 6, Netscape 6 oder Opera 5 haben User die Möglichkeit, eigene Style-Sheet-Dateien zu benutzen und gleichzeitig CSS-Angaben auf der Seite abzuschalten. Vor allem stark sehbehinderte Surfer nutzen diese Möglichkeit. Ein ansprechendes Design, das du mittels CSS erzeugt hast, kann also jederzeit ausgeschaltet werden – weswegen die Struktur des Quellcodes große Bedeutung erhält!

Warum Dynamik eine Barriere ist

Mit "Dynamik" sind vor allem Client-seitige Programme oder programmähnliche Abläufe gemeint. Server-seitige Programmierung ist generell kein Problem, solange die Ergebnisse im Browser reines HTML sind.

Im folgenden gehe ich hauptsächlich auf JavaScript ein, da dies häufig im WWW eingesetzt wird. Andere Programmierungen wie Java-Applets sollten nur über das **OBJECT**-Element dargestellt werden, das ich zu Anfang dieses Kapitels ausführlich beschrieb. Damit sind solche Programmierungen noch lange nicht barrierefrei! Am Abschnittsende findest du weitere Links.

Musst du unbedingt das **APPLET**-Element anstatt des **OBJECT**-Elements zur Einbindung deiner Programmierungen verwenden, solltest du einen Alternativtext und einer Beschreibung wie für das **OBJECT**-Element beifügen (Priorität 1):

```
<APPLET code="tanzenderText.class"
width="500" height="40" alt="Java-
Applet: tanzender Text.">
Mit einem Java-fähigen Browser würde
der Text "Das war die schönste Zeit
meines Lebens ..." mit animierten
und tanzenden Buchstaben statt
dieses Absatzes zu sehen sein.
</APPLET>
```

Auch wenn du ein guter Programmierer bist und gewisse Ansprüche an deiner Website stellst, sollten Mittel und Zweck einander entsprechen. Für die Darstellung von reinen Informationen sind aufwendige Programmierungen womöglich eine unnötige Barriere.

JavaScript und das NOSCRIPT-Element

Auch die WAI-Richtlinie 6.3 hat die Priorität 1. Sie besagt, dass beim Einsatz von Scripts, Applets und anderen programmierten Objekten die relevanten Seiten auch für ältere Browser oder Browser mit ausgeschalteter Interpretation der Programme lesbar sein sollen. Nicht jeder Browser kann alle Programmierungen interpretieren – und lässt sich die Programmierung nicht zugänglich machen, muss eine alternative Nur-Text-Seite angeboten werden. Insbesondere JavaScript wird in der Web-Programmierung sehr häufig eingesetzt, weil damit sehr viele optische Effekte bis hin zu dynamischen Daten-Konstrukten erzielt werden können.

Beim Einsatz von JavaScript solltest du immer überlegen: wird mit dem JavaScript eine Information ausgegeben? Oder wird beispielsweise mit dem **onClick**-Attribut eine bestimmte Funktion nur für Mausbenutzer ermöglicht? Was ist, wenn dein Besucher keine JavaScript-Funktionalität in seinem Browser hat?

Für alle mit JavaScript erzeugten Informationen und Funktionen solltest du eine gleichwertige Alternative in einem unmittelbar anschließendem **NOSCRIPT**-Element einfügen. Dieses Element sollte reines HTML enthalten! Im folgenden Beispiel gehe ich davon aus, dass deine Seiten über eine Fußleiste verfügen, die über eine vordefinierte Funktion **Fussleiste(x);** aufgerufen wird. In dieser Fußleiste werden mehrere Links ausgegeben, und je nach dem übergebenen Parameter **x** kann die Anzeige der Fußleisten-Links unterschiedlich gestaltet werden; so könnte z.B. auf der Kontaktseite mit dem Parameter **x="kontakt"** eine Fußleiste ohne einen Link zu sich selbst ausgegeben werden, wohl aber auf allen anderen Seiten. Für Browser ohne JavaScript-Funktionalität erschiene hier nichts. Nur wenn ein nachfolgendes **NOSCRIPT**-Element die wichtigsten Elemente enthält, findet der Besucher der Seite unten weiter führende Navigationslinks:

```
<SCRIPT language="JavaScript"
type="text/javascript"><!--
    Fussleiste("kontaktseite");
//--></SCRIPT>
<NOSCRIPT>| <A
href="index.htm">Startseite</A> | <A
href="news.htm">brand aktuell</A> |
<A href="katalog.htm">online
einkaufen</A> |</NOSCRIPT>
```

Zum Verstecken des JavaScript-Quellcodes vor älteren Browsern muss das Code als Kommentar mit **<!--** und **//-->** umschlossen werden. Im Gegensatz zu HTML sind in JavaScript die beiden Schrägstriche am Ende des Kommentars für manche Browser zwingend!

Im **NOSCRIPT**-Bereich des Beispiels sind vielleicht nicht alle Links der JavaScript-Funktion enthalten. Je größer deine Website, desto schwieriger wird die Pflege der **NOSCRIPT**-Elemente. Ein Mindestmaß an Angleichung ist ratsam.

Event-Handler

Viele Websites haben JavaScripts eingebaut, die von der Mausposition und anderen Maus-"Ereignissen" (*events*) abhängig sind, die für bestimmte Behinderungen ungeeignet sind. Ein sehr häufig eingesetzter Effekt für Grafik bzw. Links ist die Veränderung der Darstellung, wenn der User den Mauszeiger über das entsprechende Element hält. Dies wird mit dem Attributenpaar **onMouseOver** und **onMouseOut** erzielt. Einfach gesagt, wer keine Maus benutzt, hat natürlich nichts davon! Solange es hier um optische Effekte geht, d.h. keine zusätzlichen Informationen mit solchen Effekten vermittelt werden, und die jeweiligen Elemente als solche barrierefrei sind, ist nichts gegen diese Effekte einzuwenden. Solche Effekte sind sogar für viele förderlich, wenn durch die optische Hervorhebung die Navigation visuell unterstützt wird.

Aber Events werden auch dafür benutzt, eine Information zu vermitteln, Berechnungen vorzunehmen oder Formulare weiterzuleiten. Hier kannst du für Browser ohne JavaScript-Interpretation nicht immer das **NOSCRIPT**-Element zu Hilfe nehmen. Du solltest daher in jedem Fall darauf achten, dass die Event-Handler logischer Natur und nicht geräteabhängig (Mauszeiger) sind. In HTML4 sind die logischen Event-Handler-Attribute **onFocus**, **onBlur** und **onSelect**.

Kannst du nicht auf Geräte-abhängige Event-Handler-Attribute wie **onMouseOver** verzichten, solltest du laut WAI-9.3 HTML-Elementen einen zweiten redundanten Event-Handler beifügen:

- mit **onMouseDown**: **onKeyDown**
- mit **onMouseUp**: **onKeyUp**
- mit **onClick**: **onKeyPress**

> Für den Doppelklick-Handler **onDblClick** gibt es in HTML4 keine relevante Tastatureingabe.

Das folgende Beispiel zeigt ein **BUTTON**-Element, das über eine JavaScript-Funktion eine Nachricht in einem Pop-Up-Fenster ausgibt, wenn einer von zwei Events geschieht. In diesem Falle kann das entweder über die Tab-Taste erfolgen, die den Fokus auf das **BUTTON**-Element bringt, oder der Benutzer bewegt die Maus über das Element.

```
<BUTTON onFocus="hase()"
onMouseOver="hase()">Was du schon
immer wissen wolltest!</BUTTON>

<NOSCRIPT>Pop-Up: Mein Name ist
Hase!</NOSCRIPT>

...

<SCRIPT language="JavaScript"
type="text/javascript"><!--

    function hase() {alert("Mein
Name ist Hase!");}

//--></SCRIPT>
```

> Das **onFocus**-Attribut ist Geräte-unabhängig und reagiert zusätzlich zur Tastaturbedienung auf die Maus, wenn der Benutzer damit auf das **BUTTON**-Element klickt.

On-the-Fly

Links, die z.B. mit **javascript:** statt **http://** beginnen, sind problematisch. Hat ein Browser keine JavaScript-Funktionalität, führt dieser Link ins Nichts, was sehr verwirren kann. Das heißt nicht, dass man keine solchen Links benutzen darf – sie müssen aber zunächst vor Browsern ohne JavaScript-Funktionalität versteckt werden, und dann muss durch ein **NOSCRIPT**-Element eine Alternative angeboten werden. Also nicht so:

```
<A href="javascript:
location.href='start.htm'">
Startseite</A>
```

sondern so:

```
<SCRIPT language="JavaScript"
type="text/javascript"><!--

document.write("<A href='javascript:
location.href=\'start.htm\''>
Startseite</A>");

//--></SCRIPT>

<NOSCRIPT><A href="start.htm">
Startseite</A></NOSCRIPT>
```

Dieses verhältnismäßig triviale Beispiel zeigt, wie aufwändig es sein könnte, nicht nur ein einziges Element, sondern eine ganze Website anzupassen und zu pflegen. Gleichzeitig zeigt das Beispiel, dass manche gestalterische Maßnahmen wie etwa das Einbinden von JavaScript bisweilen überflüssig sind. Hier lautet die Faustregel: "Wann immer möglich, verwende HTML."

Das Beispiel enthält ein Client-seitiges, d.h. vom Browser zu interpretierendes Script. Solltest du in der Lage sein, Server-seitige Programmierungen zu erstellen, sollte es keine Schwierigkeiten für den Benutzer geben.

Weitere Programmierungen

Wie problematisch Programmierungen für User ohne Standard-Zugangssoftware sind, hat die WAI mit einer weiteren Richtlinie unterstrichen. In WAI-8.1 wird neben der eben erwähnten alternativen Darstellung von Programmierungen auch die Kompatibilität der Programmierungen mit der oft notwendigen Spezialsoftware als Anforderung der Priorität 1 formuliert. Alle Programmierungen, die interaktiven Austausch zwischen Benutzer und Anwendung ermöglichen, ob im **OBJECT**- oder **APPLET**-Element eingebunden, müssen ohne Maus bedienbar sein.

Die Zugänglichkeit von Objekten mit eigener Benutzeroberfläche ist unabhängig vom Browser und von sonstiger Zugangssoftware. Daher muss die Barrierefreiheit in der Programmierung selbst gewährleistet werden. Bist du ein Programmierer, solltest du dir folgende Websites mit weiteren Informationen zu barrierefreiem Design in Programmierungen anschauen:

- Java Accessibility (Trace R&D Center)
  trace.wisc.edu/world/java/java.htm
- IBM Guidelines for Writing Accessible Applications Using 100% Pure Java (IBM Special Needs Systems)
  www.austin.ibm.com/sns/access.html
- Active Accessibility (Microsoft Corporation)
  www.microsoft.com/enable/msaa/default.htm
- Advanced Microsoft Visual Basic 6.0, 2nd Edition, für Active Accessibility (Microsoft Corporation)
  www.microsoft.com/enable/dev/vbexcerpt.htm

Flackern, Blinken und sonstige Bewegungen

Wie bereits im ersten Abschnitt dieses Kapitels ausgeführt können flackernde Bildschirmelemente wie etwa animierte GIFs für Menschen mit fotosensitiver Epilepsie gesundheitsgefährdend sein. Die WAI hat hierzu zwei Richtlinien aufgestellt:

- WAI-7.1 (Flackernder Bildschirm, Priorität 1)
- WAI-7.2 (Flackernde und blinkende Elemente, Priorität 2).

Auf das Flackern deines Bildschirms solltest du auf jeden Fall verzichten. Das dürfte die meisten Besuchern stören – für Menschen, die an Epilepsie leiden, ist es aber geradezu gefährlich, und z.B. für Menschen mit schweren Erkrankungen des Nervensystems wie Multipler Sklerose und daraus folgenden Konzentrationsschwächen ist die Ablenkung oft zu groß, um lesen zu können.

Viel häufiger als ein flackernder Bildschirm sind jedoch blinkende Texte und Laufschrifte. Immer noch haben Screen-Reader, die ja statische Bilder erfassen, große Schwierigkeiten mit dieser Art von Elementen, da sie eine derartige Dynamik nicht kontrollieren können. In vielen Fällen führt dies sogar zu einem kompletten Systemabsturz. Auch für Sehbehinderte sind diese Darstellungselemente problematisch.

Setzt du blinkenden Text auf deine Seiten, um die Aufmerksamkeit deiner Besucher auf einen Inhalt zu lenken, benutzt du sicher das **BLINK**-Element. Die Richtlinie WAI-7.2 fordert eine Möglichkeit, das Blinken abzustellen. Das kannst du zwar mit einem kleinen Programm erreichen – viel einfacher ist es aber, wenn du für das Blinken – wenn du denn nicht darauf verzichten kannst – dem entsprechenden Element das CSS-Attribut **style="text-decoration:blink"** zuweist. Solche Style-Angaben kann der Besucher notfalls in seinem Browser abschalten.

Weder das **BLINK**- noch das **MARQUEE**-Element (Laufschrift) sind Standard-Elemente – sie tauchen in keinem HTML-Standard des W3C auf, sondern wurden willkürlich von Browser- und Editorenanbietern eingeführt.

Für die Zukunft bietet CSS Level 2 (CSS2) viele Möglichkeiten, Webseiten nach eigenem Gusto zu programmieren und dennoch dem Benutzer über seinen Browser zu erlauben, bestimmte Effekte selbst ein- oder auszuschalten, was vor allem das Blinken einschließt. Folgende CSS2-Angaben sind sehr benutzerfreundlich: **text-transform** (Groß-/Kleinschreibung). **text-shadow** (Schattierungen), **text-decoration**, (Unter-/Überstreichung, Blinken; der Benutzer kann bei **text-decoration:blink** entweder Style-Sheets ausschalten oder in seinem Benutzer-Style-Sheet das Blinken überschreiben).

Datentabellen

Tabellen jeder Art, werden sie nun für die Aufbereitung von Zahlen und anderen Daten eingesetzt (Datentabelle) oder für das Layout von Text (Layout-Tabellen), lassen sich am besten von Screen-Readern lesen, wenn die einzelnen Tabellenzellen linearisiert, d.h. Zeile für Zeile von links nach rechts lesbar sind.

In diesem Kapitel gehe ich nur auf Datentabellen ein. Im nächsten Kapitel findest du einen Abschnitt zu Layout-Tabellen.

Spalten- und Reihenbezeichnungen

Aus Sicht des Screen-Reader-Benutzers, aber auch für jeden anderen Surfer, sind korrekte und sinnvolle Überschriften für Spalten und Reihen wichtig, um eine Tabelle besser zu verstehen. Für den Durchschnitts-Besucher deiner Seite ist es in aller Regel kein Beinbruch, wenn diese Überschriften fehlen – für blinde Surfer, die mit der Braille-Zeile arbeiten, bedeuten fehlende Zellenüberschriften aber einen sehr hohen Konzentrationsaufwand.

Komplexere Tabellen sind schwer lesbar, wenn Spalten- und Reihenüberschrift nicht jederzeit abrufbar sind. Es ist nun einmal so, dass auf der Braille-Zeile maximal eine Bildschirm-Zeile angezeigt werden kann – daher auch der Name "Zeile". Die Tabellenzeilen lassen sich also immer nur nacheinander lesen. Befindet sich der Screen-Reader-Benutzer "mitten in einer Tabelle", ist es umständlich, an den Anfang der Tabelle zurückzukehren und dann wieder zu der Stelle, wo man eben war, um die Spaltenüberschrift nochmal zu lesen. Der Bildschirmleser hat es da wesentlich einfacher!

Nach WAI-5.1 müssen tabellarische Darstellungen von Daten mit dem **TABLE**-Element zusammen mit den unterstützenden Elementen aufbereitet werden, etwa **TR** (Tabellenreihe), **TD** (Tabellenzelle), **TH** (Tabellenkopf) und **CAPTION** (Tabellenüberschrift). Alternative Darstellungsformen für Daten wie das **PRE**-Element sollten auf gar keinen Fall eingesetzt werden, weil sie für die Benutzer von Screen-Readern das Lesen noch mehr erschweren (vgl. Beispiel im Abschnitt "ASCII-Zeichnungen" auf S. 71).

Die einfachste Struktur einer Tabelle sieht wie folgt aus:

```
<TABLE border="1">
  <CAPTION>Beispiel einer einfachen Tabelle in HTML.</CAPTION>
  <TR>
    <TD></TD>
    <TH>Überschrift für erste Spalte</TH>
    <TH>Überschrift für zweite Spalte</TH>
  </TR>
  <TR>
    <TH>Überschrift für erste Reihe</TH>
    <TD> Reihe 1,Spalte 1</TD>
    <TD>Reihe 1, Spalte 2</TD>
  </TR>
  <TR>
    <TH>Überschrift für zweite Reihe</TH>
    <TD>Reihe 2, Spalte 1</TD>
    <TD>Reihe 2, Spalte 2</TD>
  </TR>
</TABLE>
```

Datentabellen sollten nach diem Beispiel gebaut werden, nicht nur weil Screen-Reader solche Tabellen besser verarbeiten, sondern auch weil zukünftige Browser logische Muster in Tabellen aufzubereiten können, z.B. für Filterfunktionen.

> Benutzt du für das Seitenlayout Tabellen statt des **DIV**-Elements mit CSS (vgl. "Layout ohne Tabellen" ab S. 54), solltest du für Datentabellen reservierte Elemente wie **TH** und **COLGROUP** sowie Attribute wie **headers** und **scope** nicht nutzen. Sie werden von Browsern "manipuliert", um eine effektivere Wiedergabe von Daten zu sichern.

Verknüpfungen in komplexen Tabellen

Hat eine Datentabelle zwei oder mehr logische Ebenen von Zeilen- und/oder Spaltenüberschriften, solltest du Elemente wie **THEAD**, **TFOOT** und **TBODY** sowie **COL** und **COLGROUP** verwenden, um Datenzellen mit entsprechenden Spalten- und Reihenüberschriften zu verknüpfen. Auch die korrekte Verwendung der **axis**-, **scope**- und **headers**-Attribute ist notwendig, um komplexere Zusammenhänge zwischen Daten zu beschreiben.

In HTML 4.0 wurde das **headers**-Attribut für Tabellen eingeführt, das eine interne unsichtbare Spalten- und Reihenbeziehung erlaubt. Das Beispiel zeigt, wie du einzelne Tabellenzellen, **TD**-Elemente, mit Hilfe des **headers**-Attributs mit einer Spaltenüberschrift verknüpfst; das gilt auch für Reihenüberschriften. Das **headers**-Attribut sucht nach relevanten Überschriften in den **TH**-Elementen, denen mit dem **id**-Attribut.

```
<TABLE border="1" summary="Diese
Tabelle zeigt den Kaffekonsum
unserer Mitarbeiter in Tassen pro
Tag sowie welche Art Kaffee sie
trinken und ob sie Zucker nehmen.">
  <CAPTION>Kaffeekonsum der
Mitarbeiter</CAPTION>
  <TR>
    <TH id="sp1">Name</TH>
    <TH id="sp2">Tassen</TH>
    <TH id="sp3" abbr="Art">Art des
Kaffees</TH>
    <TH id="sp4" abbr="Zucker">mit
Zucker?</TH>
  </TR>
  <TR>
    <TD headers="sp1">Hans</TD>
    <TD headers="sp2">10</TD>
    <TD headers="sp3">Espresso</TD>
    <TD headers="sp4">nein</TD>
  </TR>
  <TR>
    <TD headers="sp1">Petra</TD>
    <TD headers="sp2">5</TD>
    <TD
headers="sp3">entcoffeiniert</TD>
    <TD headers="sp4">ja</TD>
  </TR>
</TABLE>
```

Mit dem **headers**-Attribut liest ein Screen-Reader die Tabelle wie folgt:

"Überschrift: Kaffeekonsum der Mitarbeiter

Zusammenfassung: Diese Tabelle zeigt den Kaffekonsum unserer Mitarbeiter in Tassen pro Tag sowie welche Art Kaffee sie trinken und ob sie Zucker nehmen.

Name: Hans, Tassen: 10, Art: Espresso, Zucker: nein

Name: Petra, Tassen: 5, Art: entcoffeiniert, Zucker: ja"

Die Daten, die **TD**-Elemente, sind mit den Überschriften, den **TH**-Elementen, über das **headers**- resp. das **id**-Attribut verknüpft. Das **abbr**-Attribut (*abbreviation* = Abkürzung) ermöglicht bei einer längeren Überschrift eine Beschreibung für den Screen-Reader. Mit dem **summary**-Attribut lassen sich komplexe Tabellen zusammenfassen.

Das **headers**-Attribut hilft, in zwei- und mehrdimensionalen Tabellen Verknüpfungen einzurichten. In einer einfachen Datentabelle kannst du statt des **headers**-Attributs auch das **scope**-Attribut zur Erstellung von Verknüpfungen zwischen Tabellenzellen und ihren Spalten- und Reihenüberschriften benutzen. Das folgende Beispiel ist mit dem obigen vergleichbar, nur verwenden wir hier das **scope**-Attribut, das einen der Werte **row**, **col**, **rowgroup** oder **colgroup** haben muss – auf die **COLGROUP**- und **ROWGROUP**-Elemente gehe ich hier nicht ein. Damit werden mehrere Tabellenzellen einer und derselben Überschrift zugewiesen. Dieses Beispiel

ist für den Screen-Reader identisch zum vorigen:

```
<TABLE border="1" summary="Diese
Tabelle zeigt den Kaffekonsum
unserer Mitarbeiter in Tassen pro
Tag sowie welche Art Kaffee sie
trinken und ob sie Zucker nehmen.">
  <CAPTION>Kaffeekonsum der
Mitarbeiter</CAPTION>
  <TR>
    <TH scope="col">Name</TH>
    <TH scope="col">Tassen</TH>
    <TH scope="col" abbr="Art">Art
des Kaffees</TH>
    <TH scope="col"
abbr="Zucker">mit Zucker?</TH>
  </TR>
  <TR>
    <TD>Hans</TD>
    <TD>10</TD>
    <TD>Espresso</TD>
    <TD>nein</TD>
  </TR>
  <TR>
    <TD headers="sp1">Petra</TD>
    <TD>5</TD>
    <TD>entcoffeiniert</TD>
    <TD>ja</TD>
  </TR>
</TABLE>
```

Solltest du als Tabellenspezialist komplexe Tabellen mit mehreren Dimensionen erstellen, studiere das folgende Beispiel. Es ist schwieriger, da die Datenzellen mit drei logischen Ebenen verknüpft sind.

```
<TABLE border="1">
  <CAPTION>Reisekosten-Übersicht</CAPTION>
  <TR>
    <TD></TD>
    <TH id="ueberschrift2" axis="ausgaben">Verpflegung</TH>
    <TH id="ueberschrift3" axis="ausgaben">Übernachtungen</TH>
    <TH id="ueberschrift4" axis="ausgaben">Transport</TH>
    <TD>Zwischensummen:</TD>
  </TR>
  <TR>
    <TH id="ueberschrift6" axis="ort">Frankfurt a.M.
    <TH></TH> <TH></TH> <TH></TH> <TD></TD>
  </TR>
  <TR>
    <TD id="ueberschrift7" axis="datum">15.10.2001</TD>
    <TD headers="ueberschrift6 ueberschrift7 ueberschrift2">120,00</TD>
    <TD headers="ueberschrift6 ueberschrift7 ueberschrift3">235,00</TD>
    <TD headers="ueberschrift6 ueberschrift7 ueberschrift4">92,00</TD>
    <TD></TD>
  </TR>
  <TR>
    <TD id="ueberschrift8" axis="datum">16.10.2001</TD>
    <TD headers=" ueberschrift6 ueberschrift8 ueberschrift2">54,50</TD>
    <TD headers=" ueberschrift6 ueberschrift8 ueberschrift3">225,00</TD>
    <TD headers=" ueberschrift6 ueberschrift8 ueberschrift4">92.00 </TD>
    <TD></TD>
```

```
  </TR>
  <TR>
    <TD>Zwischensummen:</TD>
    <TD>174,50</TD>
    <TD>460,00</TD>
    <TD>184,00</TD>
    <TD>918,50</TD>
  </TR>
  <TR>
    <TH id="ueberschrift10" axis="ort">Berlin</TH>
    <TH></TH> <TH></TH> <TH></TH> <TD></TD>
  </TR>
  <TR>
    <TD id="ueberschrift11" axis="datum">22.10.2001</TD>
    <TD headers="ueberschrift10 ueberschrift11 ueberschrift2">189,00</TD>
    <TD headers="ueberschrift10 ueberschrift11 ueberschrift3">220,00</TD>
    <TD headers="ueberschrift10 ueberschrift11 ueberschrift4">75,00</TD>
    <TD></TD>
  </TR>
  <TR>
    <TD id="ueberschrift12" axis="datum">23.10.2001</TD>
    <TD headers="ueberschrift10 ueberschrift12 ueberschrift2">72,50</TD>
    <TD headers="ueberschrift10 ueberschrift12 ueberschrift3">214,00</TD>
    <TD headers="ueberschrift10 ueberschrift12 ueberschrift4">74,00</TD>
    <TD></TD>
  </TR>
  <TR>
    <TD>Zwischensummen:</TD>
    <TD>261,50</TD>
    <TD>434,00</TD>
    <TD>149,00</TD>
    <TD>844,50</TD>
  </TR>
  <TR>
    <TH>Gesamtsummen:</TH>
    <TD>436,00</TD>
    <TD>894,00</TD>
    <TD>333.00</TD>

    <TD>1663,00</TD>
  </TR>
</TABLE>
```

Reisekosten-Übersicht

	Verpflegung	Übernachtungen	Transport	Zwischensummen:
Frankfurt a.M.				
15.10.2001	120,00	235,00	92,00	
16.10.2001	54,50	225,00	92.00	
Zwischensummen:	174,50	460,00	184,00	918,50
Berlin				
22.10.2001	189,00	220,00	75,00	
23.10.2001	72,50	214,00	74,00	
Zwischensummen:	261,50	434,00	149,00	844,50
Gesamtsummen:	436,00	894,00	333.00	1663,00

Diese Tabelle hat mehrere Verknüpfungen zwischen Datenzellen und Spalten- bzw. Reihenüberschriften. Ein Screen-Reader könnte die Tabelle wie untenstehend auslesen:[1]

Tabellenüberschrift: Reisekosten-Übersicht
Überschrift: Verpflegung; Überschrift: Übernachtungen; Überschrift: Transport; Zwischensummen
Überschrift: Frankfurt a.M.
15.10.2001; Ort Frankfurt Datum 15.10.2001 Ausgaben Verpflegung 120,00; Ort Frankfurt Datum 15.10.2001 Ausgaben Übernachtungen 235,00; Ort Frankfurt Datum 15.10.2001 Ausgaben Transport 92,00
16.10.2001; Ort Frankfurt Datum 16.10.2001 Ausgaben Verpflegung 54,50; Ort Frankfurt Datum 16.10.2001 Ausgaben Übernachtungen 225,00; Ort Frankfurt Datum 16.10.2001 Ausgaben Transport 92.00
Zwischensummen; 174,50; 460,00; 184,00; 918,50
Überschrift: Berlin
22.10.2001; Ort Berlin Datum 22.10.2001 Ausgaben Verpflegung 189,00; Ort Berlin Datum 22.10.2001 Ausgaben Übernachtungen 220,00; Ort Berlin Datum 22.10.2001 Ausgaben Transport 75,00
23.10.2001; Ort Berlin Datum 23.10.2001 Ausgaben Verpflegung 72,50; Ort Berlin Datum 23.10.2001 Ausgaben Übernachtungen 214,00; Ort Berlin Datum 23.10.2001 Ausgaben Transport 74,00
Zwischensummen; 261,50; 434,00; 149,00; 844,50
Überschrift: Gesamtsummen; 436,00; 894,00; 333.00; 1663,00

Weitere
Informationen zu Tabellenverknüpfungen findest Du bei:
http://www.webreview.com/2001/04_27/webauthors/index04.shtml.

Auch diesen Screenshot findest du auf der Webseite des Hefts.

Ohne die Verknüpfungen würde diese Tabelle von einem Screen-Reader wie folgt gelesen:

Reisekosten-Übersicht Verpflegung Übernachtungen Transport Zwischensummen: Frankfurt a.M. 15.10.2001 120,00 235,00 92,00 16.10.2001 54,50 225,00 92.00 Zwischensummen: 174,50 460,00 184,00 918,50 Berlin 22.10.2001 189,00 220,00 75,00 23.10.2001 72,50 214,00 74,00 Zwischensummen: 261,50 434,00 149,00 844,50 Gesamtsummen: 436,00 894,00 333.00 1663,00

Braille-Zeilen können bis maximal 80 Zeichen auf dem Braille-Display ausgeben. Danach erst werden die nächsten (maximal 80) Zeichen dargestellt. Enthält eine Tabelle mehr als 80 Zeichen in einer Zeile, erzeugt die Braille-Zeile einen zusätzlichen, am Bildschirm nicht existierenden Zeilenumbruch. Das bedeutet für den sehgeschädigten Leser dann natürlich eine noch größere Unübersichtlichkeit.

Der Einsatz von Framesets

Die Verwendung von Framesets in der Web-Gestaltung war für blinde Personen schon immer problematisch , da sie sich – ähnlich wie bei den Zellen einer Tabellen – den "Gesamtüberblick" nur auf etwas umständliche Art durch das einzelne Auslesen der Frames verschaffen können. Grundsätzlich unmöglich ist der Zugriff bei Benutzern, die ältere Versionen des Text-Browsers "Lynx" einsetzen. Mittlerweile können viele Blinde, die sich mit entsprechender Anpassung der Screen-Reader-Versionen auf Windows-Systeme umgestellt haben, mit Standard-Browsern und anderer Web-Zugangssoftware relativ unproblematisch mit Framesets umgehen, auch wenn es weiterhin umständlich bleibt. Voraussetzung ist korrekte Einbindung der Frames: so ist z.B. die sinnvolle Benennung der einzelnen Frames eine wichtige Orientierungshilfe. Für ältere Text-Browser ist das **NOFRAMES**-Element besonders wichtig – wie es das übrigens auch für Suchmaschinen ist!

Dennoch steigert der Einsatz von Frames die Unübersichtlichkeit eines WWW-Angebotes in mehr oder weniger hohem Maße. Während ein Frameset mit zwei oder drei Frames mit neueren Screen-Readern gut erfassbar ist, sind übermäßig verschachtelte Seiten oder Seiten ohne identifizierende Bezeichnung kaum navigierbar.

Bevor du dich zur Gestaltung deiner Website mit üblichen, dafür vorgesehenen HTML-Elementen **FRAMESET**, **FRAME** und **IFRAME** entscheidest, solltest du folgende Nachteile von Framesets bedenken:

- ohne Script-Programmierung wird die "Zurück"-Funktion in einigen Browsern gestört
- es ist dem Besucher nicht möglich, einen Bookmark auf den Inhalt deiner Seiten zu setzen (außer auf das Frameset selbst)
- wenn ein Besucher ein zweites Browserfenster öffnet und sein Browser wieder die Startseite oder einen anderen unerwarteten Inhalt öffnet, ist das für ihn schlicht ärgerlich

In den folgenden Abschnitten erfährst du, wie du Framesets zugänglich machst und wie du mit HTML und CSS Alternativen gestalten kannst, um die eingeschränkten Möglichkeiten, die du beim Verwenden von Frames hast, zu umgehen.

Sinnvolle Frame-Benennung

In der Richtlinie WAI-12.1 wird eine sinnvolle Bezeichnung von Frames mit der Priorität 1 gefordert. Sinnvoll bedeutet hier, dass der Titel des einzelnen Frames auf seinen Inhalt schließen lässt. Setzt du ein "klassisches" Frame-Muster ein, nämlich einen schmaleren Frame auf der linken Seite des Browser-Fensters mit Navigationselementen, Menüpunkten, Inhaltsverzeichnissen u.dgl. und rechts einen breiteren mit den Inhalten deiner Site, sind Frame-Titel wie "Navigation" oder "Menü" für den linken Frame und Titel wie "Inhalt" oder "Text" für den rechten sinnvolle Bezeichnungen. Die Bezeichnung der Frames wird den **FRAME**-, **IFRAME**- und **FRAMESET**-Elementen über das **title**-Attribut zugeordnet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
<HTML>
<HEAD> ... </HEAD>
<FRAMESET cols="10%, 90%" title="die elektronische Bibliothek">
  <FRAME src="menue.htm" title="Navigation">
  <FRAME src="start.htm" title="dokumente">
  <NOFRAMES>... </NOFRAMES>
</FRAMESET>
</HTML>
```

Das **DOCTYPE**-Element gibt an, um welche Art Dokument es geht – in diesem Fall ein Frameset; auf Seite 72 erfährst du mehr dazu.

Beschreibung komplexer Framesets

Je mehr Frames in einem Frameset verschachtelt werden, desto schwieriger wird die Handhabung der Seite für Screen-Reader-Anwender. Benutzt du zum Beispiel ein und denselben Frame für verschiedene Inhalte, die du mit einem einzigen Titel nicht abdecken kannst, solltest du dir überlegen, über das **longdesc**-Attribut eine weitere Kurzbeschreibung deines Framesets anzubieten. Das gibt Screen-Reader-Nutzern die Chance, die Organisation deiner Seiten schneller aufzufassen.

Das folgende Beispiel ist bei aller Bemühung, eine zugängliche Gestaltung zu erreichen, eher eine Demonstration, wie du ein Frameset *nicht* gestalten solltest – derart komplexe Framesets sind für Screen-Reader schwer zu navigieren.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
<HTML>
<HEAD>
  <TITLE>Ein Tutorial zum effektiveren Webdesign</TITLE>
</HEAD>
<FRAMESET cols="20%,*,15%">
  <FRAMESET rows="40%,*">
    <FRAME src="sponsoren.htm" name="sponsoren" title="Sponsoren">
    <FRAME src="nav-site.htm" name="site" title="angebotsübergreifende
Navigation" longdesc="frameset-beschr.htm#site">
  </FRAMESET>
  <FRAME src="einleitung.htm" name="inhalt" title="Inhalte"
longdesc="frameset-beschr.htm#inhalt">
  <FRAMESET rows="*,30%">
    <FRAME src="themen.htm" name="index" title="andere relevante Themen"
longdesc="frameset-beschr.htm#themen">
    <FRAME src="werbung.asp?filter=17" name="werbung" title="Werbung">
  </FRAMESET>
  <NOFRAMES>
    <P><A href="nur_text.htm">Nur-Text-Startseite</A>.</P>
    <P><A href="frameset-beschr.htm">Beschreibung des Framesets</A>.</P>
  </NOFRAMES>
</FRAMESET>
</HTML>
```

In der Datei **frameset-beschr.htm** könnten bei Angabe der entsprechenden Marken folgende Erläuterungen stehen:

#site	Dieser Frame bietet weitere Links zu anderen wichtigen Bereichen unserer Website: Foren und Artikel, unsere Multimedia-Produktionen mit Referenzliste, Nachrichten, Download-Bereich
#inhalt	In diesem Frame werden die Seiten des Tutorials angezeigt
#themen	In diesem Frame finden Sie Links zu aktuellen Themen und Angeboten ...

Das Beispiel geht zurück auf die Richtlinie WAI-1.1, die Alternativtexte für alle Elemente fordert, die selbst kein Text sind. Änderte sich nun der Inhalt eines Frames, stimmten die Beschreibungen nicht mehr. Umso wichtiger ist es also, solche Erläuterungen auch im **NOFRAMES**-Element zu integrieren!

Das NOFRAMES-Element

Auch das **NOFRAMES**-Element geht auf die Richtlinie WAI-1.1 zurück, wie auch des weiteren auf WAI-6.5, wo für jegliche Form der Dynamik alternative Inhalte gefordert werden. Nun sind Frames als solche nicht dynamisch – aber die Organisation einer Seite in 3, 5 oder 10 einzelnen Frames ist ein dynamischer Faktor, den ein Frames-unfähiger Browser nicht wiedergeben kann. Auch die häufige zwangsläufige Kombination von komplexen Framesets mit Scripts und anderen Programmierungen, um als Webmaster die Seiten im Griff zu halten, führt gerade bei älteren Browsern immer wieder zu Problemen. Daher ist das **NOFRAMES**-Element eine notwendige Ergänzung für diese Browser – also Angaben ohne Frames und ohne Script-Programmierung!

> Gestaltest du deine Website mit Frames und Scripts zu ihrer Steuerung, ist die Interpretation von Script-Programmierung beim Testen unbedingt auszuschalten!

Es empfiehlt sich auch, bei Framesets mit zwei Frames das **NOFRAMES**-Element mit den wichtigen Links zur Navigation auf der Site einzufügen, weil manche Suchmaschinen nur einzelne Seiten durchsuchen können, nicht aber einzelne Frames eines Framesets.

Ruft ein User das folgende Frameset auf:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Frameset//EN">
<HTML>
<HEAD>
<TITLE>Meine Startseite mit einem
Frameset</TITLE>
</HEAD>
<FRAMESET cols="50%, 50%">
  <FRAME src="start.htm"
title="Texte">
    <FRAME src="menue.htm"
title="Inhaltsangabe">
  <NOFRAMES>
    <A
href="menue.htm">Inhaltsangabe</A>.
     ... (weitere
Navigationselemente aus start.htm
sollten hier ebenfalls zugänglich
gemacht werden) ...
    </NOFRAMES>
</FRAMESET>
</HTML>
```

und kann der Browser keine Frames darstellen, wird der Inhalt des **NOFRAMES**-Elements dargeboten, und in diesem Fall die Navigationsseite sowie eventuelle weitere Navigationselemente aus der Startseite des Web-Angebots,.

> Änderst Du Navigationselemente auf deiner Website, müssen sie auch im **NOFRAMES**-Element geändert werden.

Weitere Aspekte bei der Gestaltung mit Framesets

In diesem Abschnitt habe ich einige Aspekte von alternativen Texten in Verbindung mit Framesets angesprochen, etwa dass Änderungen in der Navigation auch im **NOFRAMES**-Element vollzogen werden müssen. Ein Spezialfall ist die Angabe einer Grafik an Stelle einer Textdatei als Frame-Quelle , etwa so:

```
<FRAME src="bild.gif" name="f1"
title="Frame enthält nur ein Bild
mit folgendem Motiv: ...">
```

Kann sich diese Frame-Quelle dynamisch ändern, wird es unmöglich, eine neue Alternativtext-Belegung zu vergeben. Daher sollte in diesem Fall statt einer Grafik grundsätzlich eine HTML-Seite eingebunden werden, die z.B. nur eine Grafik enthält. In der HTML-Datei kann dann die Grafik selbst einen Alternativtext enthalten.

Im Frameset steht in diesem Fall folgendes:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Frameset//EN">
<HTML>
<HEAD> ... </HEAD>
<FRAMESET cols="100%"
title="Frameset mit wechselndem
Inhalt">
  <FRAME name="f1" src="bild.htm"
title="Verschiedene Bilder">
<NOFRAMES> ... </NOFRAMES>
</FRAMESET>
</HTML>
```

In der Frame-Quelle **bild.htm** wird angegeben:

```
... <IMG src="bild.gif" alt="Das
schönste Bild von mir"> ...
```

Mit dem **target**-Attribut oder dem **location.href**-Befehl in JavaScript wird festgelegt, in welchem Frame ein ausgewählter Link angezeigt werden soll. Mit dem reservierten Wert **target="_blank"** oder einem Wert des **target**-Attributs, der keinem der derzeit offenen Browser-Fenster und/oder –Frames entspricht, wird das Ziel eines Links in einem neuen Fenster dargestellt. Diese Funktionalität in den meisten grafischen Browsern kann sehr nützlich sein, um beispielsweise Detailinformationen anzuzeigen oder um Bilder vergrößert darzustellen. Allerdings bilden diese Zusatzfenster für Benutzer von Screen-Readern Probleme, die nicht ohne weiteres umgegangen werden können. Der Screen-Reader ist manchmal nicht in der Lage, das neue Fenster zu erfassen, und zeigt dem Benutzer weiterhin lediglich die Herkunftsseite des Links an. Daher solltest du diese Frames-Technik vermeiden, indem du neue Informationen immer in ein und dem selben Frameset darstellen lässt.

Eine Alternative zu Frames

Der "klassische" Gebrauch von Frames ist die Unterteilung einer Seite in einem linken Navigations- und einem rechten Inhalts-Frame. Anstatt eines Framesets kannst du mit dem **OBJECT**-Element arbeiten. Zukünftige Browser werden dieses Element noch besser unterstützen, gerade im Hinblick auf Grafik und Multimedia. Das **OBJECT**-Element kann allerdings auch weitere HTML-Dokumente im aktuellen Dokument aufnehmen. Gehe wie folgt vor:

1. Erstelle eine HTML-Datei MENUE.HTM, in der mehrere Links zeilenweise angeordnet sind.
2. Füge am Ende eines neuen (oder anderen) Dokuments folgende Zeilen ein:

```
<DIV class="menue">
  <OBJECT data="menue.htm">
Zur <A
href="menue.htm">Menüleiste</A>.
</OBJECT>
</DIV>
```

3. Füge folgende Angabe in dein Style-Sheet ein:
   DIV.menue { float:left; width:25%; }
   (erzeugt eine "Zelle" auf der linken Seite der Seite, die 25% der Bildschirmbreite hat)
   oder
   DIV.menue { position:fixed; left:0; bottom:0; }
   (bewirkt, dass das **OBJECT**-Element in der unteren linken Ecke des Bildschirms positioniert wird und auch dann stehen bleibt, wenn die restliche Seite nach unten oder oben gescrollt wird.)

Solche Objekte werden nicht von allen Browsern unterstützt. Diese Navigationsmechanismen lassen sich auch per Server-Side-Include in die Seite integrieren, um Browser-unabhängige Darstellungen zu gewährleisten. Mehr dazu findest du hier: **www.wilsonweb.com/articles/ssi.htm**

Audio: Hilfe oder Hürde?

Aus Sicht der Barrierefreiheit haben Audio-Clips gleich für mehrere Personengruppen Bedeutung:

- Für schwerhörige oder taube Menschen ist Audio nicht zugänglich, wenn kein alternativer Text bereit gestellt wird. Dies gilt ganz besonders für Clips, die wichtige Informationen beinhalten, aber auch für Begrüßungen und eventuelle akustische Hinweise. Bei Audio-Clips, die nur Musik oder Geräusche wiedergeben und keine inhaltliche Bedeutung haben, solltest du zumindest einen Hinweis geben; so wird vermieden, dass der Leser glaubt, er habe etwas wichtiges versäumt. Bei Stand-alone-Audio-Clips, d.h. Clips, die als Link auf einer Seite eingebaut sind, kannst du ein Link zu einer Textbeschreibung oder das **title**-Attribut verwenden. Darüber freuen sich dann auch diejenigen, die keinen Sound von ihren Rechnern erhalten (WAI-1.1).

- Für sehbehinderte oder blinde Surfer können Audio-Clips andererseits den Zugang zu den Seiten erleichtern, z.B. durch gesprochene Hinweise zu Inhalt oder Navigation. Auch können Audio-Clips eingesetzt werden, um längere Texte direkt aus dem Web vorzulesen, wobei die bloße Wiedergabe von Textinhalten auf einer Seite normalerweise nicht notwendig ist, weil Screen-Reader-Benutzer über eine eigene Sprachausgabe verfügen – Ausnahmen bilden hier etwa mit Macromedia's Flash vektorisierte Texte. Es ist beispielsweise möglich, diese Broschüre über barrierefreies Webdesign von jemandem vorlesen zu lassen und Kapitel-weise als Audio-Clips bereit zu stellen (WAI-1.3).

- Auch für Menschen, die Lernschwierigkeiten oder Konzentrationsschwächen haben, können Audio-Clips als Unterstützung eingesetzt werden – wenn du z.B. aufwendige Formulareingaben über mehrere Seiten anbietest, wenn Hinweise über verschiedene Möglichkeiten in der Navigation erscheinen oder wenn du den Besucher auf eine bevorstehende Aktion aufmerksam machen willst, kann ein kurzer gesprochener Text hilfreich sein (WAI-14.2).

Textalternativen für Sound

Vielfach wird Sound auf WWW-Seiten eingebaut, wobei der Besucher zunächst keinen Einfluss hat, ob die Klänge vom Browser heruntergeladen und dann abgespielt werden – etwa bei Hintergrund-Musik, die mit dem **BGSOUND**-Element erzeugt wird. Das ist auch völlig in Ordnung – nur muss der Text einen Hinweis enthalten, etwa einen Link "Informationen zur Hintergrundmusik", der eine entsprechende Erläuterung anbietet.

Problematisch wird Sound dann, wenn du deine Besucher allein damit auf Funktionen und Inhalte hinweist. Das ist nicht nur für den eine Barriere, der nichts oder wenig hört, sondern auch für alle, deren Computer Sound nicht wiedergeben kann.

Stell dir vor, du hast eine Mailing-Liste, in der sich jeder Besucher deiner Website über ein Formular mit seiner Emailadresse eintragen kann. Um Fehler durch Angabe einer ungültigen Emailadresse zu vermeiden, hast du ein JavaScript geschrieben – oder aus dem WWW geladen –, das alle möglichen Gültigkeitsregeln für Emailadressen anwendet. Im Formular wird nach dem Absenden der Formularinhalte dieses JavaScript aufgerufen; und nur wenn die eingegebene Adresse allen Gültigkeitsregeln entspricht, wird sie zum automatischen Eintrag in deine Mailing-Liste an eine weitere Datei weiter gegeben. Entspricht sie nicht den Gültigkeitsregeln , wird in Folge der Programmierung des JavaScripts ein als Fehlermeldung erkennbarer Ton ausgegeben. Am Bildschirm passiert weiter nichts.

Diese Fehlermeldung ist natürlich tückisch. Am Bildschirm tut sich einfach nichts, und der User bemerkt vielleicht gar nicht, dass er sich bei der Eingabe seiner Emailadresse vertippt hat.

Das Formular für das Beispiel mag so aussehen:

```
<FORM name="eingabe" method="post" action="verarbeiten.asp"
onSubmit="return ueberpruefen()">
  <INPUT name="email" type="text" width="30">
  <INPUT type="submit" value="Absenden">
</FORM>
```

Dieses Formular erfordert die Eingabe einer Emailadresse im Eingabefeld **email**. Wählt der Besucher den Absende-Button über Tastatur oder Maus, wird zunächst über das **onSubmit**-Attribut die JavaScript-Funktion **ueberpruefen()** aufgerufen, die als Rückgabewert entweder **true** liefern soll – also wahr, die Emailadresse entspricht den Gültigkeitsregeln – oder **false** – also falsch, die Emailadresse entspricht ihnen nicht. Ist der Wert **true**, werden die Formularinhalte, also die eingegebene Emailadresse, an eine Datei **verarbeiten.asp** geschickt. Die Funktion **ueberpruefen()** könnte etwa so aussehen:

```
<SCRIPT language="JavaScript" type="text/javascript"><!--
function ueberpruefen() {
  if (document.eingabe.email.value == "") {
    location.href="sounds/fehler.wav";
    return false; }
    /* weitere Befehle zur Überprüfung der Gültigkeit */
  else return true; }
//--></SCRIPT>
```

Im Beispiel untersucht die Gültigkeitsprüfung nur, ob das Formularfeld **email**, also das JavaScript-Objekt **document.eingabe.email**, leer ist. Du kannst sie natürlich ausbauen, z.B. um zu prüfen, ob das "@" sowie ein Domain- und ein Länderteil vorkommt, ob es keine Sonder- oder Leerzeichen gibt usw.

Hat der Benutzer in diesem Beispiel nichts in das Formularfeld eingegeben, wird die Sound-Datei **fehler.wav** aufgerufen.

Du solltest entspr. WAI-1.1 deinem JavaScript die Fähigkeit geben, Zeichen am Bildschirm anzuzeigen. Nachfolgend wird das Script um eine **alert()**-Funktion ergänzt, die eine zusätzliche Fehlermeldung in einem Pop-Up-Fenster erzeugt:

```
<SCRIPT language="JavaScript"
type="text/javascript"><!--
function ueberpruefen() {
  if (document.eingabe.email.value
== "" ) {location.href="sounds/
fehler.wav";
    alert("Bitte geben Sie eine
gültige Email-Adresse ein! ");
    return false; }
  else return true; }
//--></SCRIPT>
```

Hast du verschiedene Blockelemente wie **P**, **DIV**, usw. auf deiner Seite mit dem **name**-Attribut benannt, kannst du diese Elemente natürlich über das **document**-Objekt in JavaScript unmittelbar ansprechen und mit neuem Inhalt füllen, anstatt die **alert()**-Funktion zu verwenden!

Textalternativen für Stand-alone-Audio-Dateien

Stellst du Audio-Clips zum Download bereit, ist es wie bei jedem anderen Download sinnvoll, den wesentlichen Inhalt zu beschreiben. Enthalten die Audio-Dateien gesprochene Sprache, solltest du das Gesprochene auch in einer alternativen Textfassung anbieten. Bei kürzeren Texten lässt sich das im **title**-Attribut des Links unterbringen:

```
<A href="aufruf.wav" title="Audio-
Clip: Lass uns gemeinsam an
barrierefreiem Webdesign
arbeiten!">Der Autor spricht ...</A>
```

Alternativ kannst du den Hinweis über eine "begleitende" Grafik mit Alternativtext zur Verfügung stellen:

```
<A href="aufruf.wav"><IMG
src="grafik/audio.jpg" alt="Audio-
Clip: Lass uns gemeinsam an
barrierefreiem Webdesign arbeiten!">
Der Autor spricht ...</A>
```

Im Gegensatz zu den "beigefügten" 1-Pixel-großen transparenten Grafiken für Screen-Reader-Nnutzer sollten Beschreibungen für Audio-Dateien klar erkennbar sein!

Ist der gesprochene Text länger, sollte eine weitere Textversion des Sound-Clips verfügbar sein, z.B. als Link auf eine zusätzliche Datei:

```
<A href="thesen.au"><IMG src="grafik/
audio.jpg" alt="Audio-Clip: Thesen
zu ..."> Thesen (.AU, 590KB)</A>, <A
href="thesen.htm">Thesen als HTML</A>
```

Mit der zusätzlichen Textversion kann übrigens auch eine Suchmaschine etwas anfangen!

Bedenke auch, dass große Audio-Dateien lange Übertragungszeiten haben und daher die Dateigröße für alle downloadbare Dateien angegeben werden sollte. Auch das Format ist eine nützliche Information für deinen Besucher, der vielleicht eine andere Software-Ausstattung hat als du!

Sound mit CSS

Entsprechend WAI-11.3 (Priorität 3) kannst du dir auch überlegen, ob du deine Seiten auch mit Style-Sheets akustisch aufbereitest. Die Audio-Eigenschaften in CSS2 bieten Nicht-Sehenden sowie Benutzern von Audio-Browsern ähnliche Informationen wie Schriftangaben für visuell-orientierte Surfer. Das folgende Beispiel zeigt, wie du bestimmte Audio-Eigenschaften in deine Style-Sheets integrieren kannst, einschließlich der **voice-family**-Eigenschaft, die quasi der Schriftart im Visuellen entspricht:

```
H1 {
  voice-family:paul;
  stress:20;
  richness:90;
  cue-before: url("sounds/dong.wav")
}
```

In CSS2 berücksichtigte Audio-Eigenschaften:

- **volume** (Lautstärke)
- **speak** (bestimmt, ob ein Wort gesprochen wird und wenn ja, ob es als Wort ausgesprochen oder buchstabiert wird)
- **pause**, **pause-before** und **pause-after** (Bestimmung von Sprechpausen)
- **cue**, **cue-before** und **cue-after** (Bestimmung unterstützender Sounds – wie ein Ikon im Visuellen)
- **play-during** (Hintergrund-Sound)
- **azimuth** und **elevation** (Sound-Dimensionierung, um z.B. Stimmen zu unterscheiden)
- **speech-rate**, **voice-family**, **pitch**, **pitch-range**, **stress** und **richness** (Qualitätseigenschaften der gesprochenen Sprache)
- **speak-punctuation** und **speak-numeral** (Bestimmung des gesprochenen bei Satzzeichen und Zahlen)
- **speak-header** (Bestimmung der Aussprache bei Tabellen-Kopfzeilen)

Sind Multimedia barrierefrei gestaltbar?

Die Antwort lautet "Ja!" Mit Multimedia sind Video, Animationen, Dia-Shows oder andere Zeitbasierte Präsentation gemeint. Solche Objekte, die mit dem **OBJECT**-Element in HTML eingebunden werden, bereiten für taube, schwerhörige, blinde und sehbehinderte Surfer Probleme, wenn sie nur mit einer Bild- und Sound-Spur dargeboten werden. Video-Clips lassen sich aber auch mit zusätzlichen Spuren für ergänzende textliche Beschreibungen (Untertitel) oder gesprochenem Text (Audio-Deskription) ausstatten. Gerade bei Videos liegen natürlich auch Regie-Anweisungen vor, die ohne viel Aufwand zur barrierefreien Gestaltung von Video-Clips anpassbar sind.

Im wesentlichen solltest du dir entsprechend der Richtlinie WAI-1.1 beim Einsatz von Multimedia folgendes überlegen:

1. Enthalten deine Multimedia-Dateien viel gesprochenen Text, solltest du entsprechende Textalternativen anbieten.
2. Gibt es nur wenig oder keinen Sound, solltest du entsprechende Sprachalternativen bieten.
3. Verfügen die Dateien über Sprache wie auch über "für sich sprechende" Bilder, solltest du die textlichen und auditiven Synchronisationsmöglichkeiten in deinen Multimedia ausschöpfen.

Textalternativen für Audio-Spuren eines Video-Clips

Eine Textalternative für die Audio-Spur eines Video-Clips entspricht der Textalternative für eine gesprochene Audio-Datei. Bei Multimedia kann aber die Textalternative als Untertitel synchronisiert werden, wie ich das unten im Abschnitt "Synchronisationen" beschreibe.

Auch wenn die Synchronisation die eindeutig beste Methode ist, die Audio-Spur eines Video-Clips für Gehörlose zugänglich zu machen, kannst du einen Link zu einem gleichwertigen Text vor allem dann anbieten, wenn du z.B. Clips in Form einer Nachrichtensendung oder eines Interviews auf deiner Seite hast.

Alternative für eine Video-Spur

Auch für Screen-Reader-Anwender sind Multimedia meist erst dann vollständig verständlich, wenn zusätzlich bestimmte optische Elemente erläutert werden, etwa Szenerie, Bewegungen, Körpersprache, aber auch Tabellen, Grafiken und andere komplexere, bildhaft dargestellte Zusammenhänge.

Multimedia sollte immer eine Textalternative enthalten, wenn deine Seite so besser verstanden wird. Stelle dir z.B. eine Animation vor, die einen Wolkenbruch mit Blitzeinschlägen zeigt. Gibt deine Seite Informationen zum aktuellen Wetter, dürfte ein Alternativtext der Art **alt="Animation: Wolkenbruch mit Blitzen"** völlig ausreichen – die Informationen über das Wetter gehen hoffentlich aus dem Text hervor. Wird jedoch die Animation auf einer Seite verwendet, wo Schüler in pädagogischem Rahmen über Wolkenbildung und ihre Beziehung zu Landmassen informiert werden, solltest du eine ausführlichere Beschreibung beifügen, damit auch diejenigen, die die Animation nicht sehen können, etwas davon haben. Wie das mit dem **OBJECT**-Element geht, erfuhrst du bereits im ersten Abschnitt dieses Kapitels.

Generell – insbesondere bei längeren Clips – ist es bedeutend besser, wenn du diese Beschreibung als Audio-Deskription auf eine zusätzliche Spur deines Videos legst. Die Richtlinie WAI-1.3 verlangt hier in Konkurrenz mit WAI-1.1 eine auditive Beschreibung von Video-Spuren, so lange Zugangssoftware die Textalternativen nicht automatisch lesen können.

Eine Textalternative ist für dich wahrscheinlich einfacher anzubieten als eine gesprochene, denn Sprachaufnahmen erfordern Aufnahmegeräte und -bearbeitung. Ein weiterer Vorteil der Textalternative gegenüber einer Sprachalternative ist, dass diese Seite indexiert und durchsucht werden kann. Die Grundstruktur sieht wie folgt aus:

```
<OBJECT ... >
    <IMG src="standbild.gif"
alt="Standbild aus ... ">
    ... Textbeschreibung des Videos
...
</OBJECT>
```

Für nähere Informationen zur Verwendung des **OBJECT**-Elements siehe Seite 11.

Die 66 Gebote (und zum Teil auch Verbote) auf einem Blick

Die WAI bietet keine offizielle deutsche Übersetzung der Zugänglichkeitsrichtlinien für die Gestaltung von Web-Inhalten an. Die Bedeutung der auf **www.w3.org/TR/WAI-WEBCONTENT** aufgeführten Anforderungen kannst du der folgenden Übersicht entnehmen.

WAI	Bedeutung	Priorität
1	**Biete Inhalte an, die im Wesentlichen dieselben auditiven und visuellen Funktionen bereitstellen**	
1.1	Biete Textalternativen für alle Elemente an, die selbst kein Text sind.	1
	i. für Grafiken und grafische Buttons ii. für grafische Textdarstellung und Symbole iii. für Image-Map-Bereiche iv. für animierte GIFs v. für Programmierungen (Applets) vi. für ASCII-Zeichnungen vii. für Frames viii. für Script-Programmierungen ix. für grafische Aufzählungszeichen x. für grafische Abstandshalter xi. für Audio (mit oder ohne Aufruf des Benutzers) xii. für Audio-Dateien (stand-alone) xiii. für die Audio-Spur eines Videos xiv. für Video	
1.2	Biete redundante Textlinks für jeden aktive Bereich eines Server-seitigen Image-Maps an.	1
1.3	Biete Audio-Deskription für die wichtigen Informationen einer visuellen Darbietung in Multimedia-Präsentationen an.	1
1.4	Biete Synchronisationen (Untertitel oder Audio-Deskription) für Video-Spuren in Zeit-basierten Multimedia-Präsentationen an.	1
1.5	Biete redundante Textlinks für jede aktive Region eines Client-seitigen Image-Maps an.	3
2	**Stelle sicher, dass Text und Grafik auch verstanden werden, wenn sie ohne Farbe betrachtet werden.**	
2.1	Stelle sicher, dass alle durch Farbe gekennzeichneten Informationen auch ohne Farbe wahrgenommen werden können.	1
2.2	Stelle sicher, dass Kombinationen aus Vorder- und Hintergrundfarben genügend Kontrast bieten für Menschen mit Farbwahrnehmungsdefiziten oder für die Betrachtung auf Monochrom-Bildschirmen.	2 (Grafik) 3 (Text)
3	**Gestalte Seiten mit den korrekten Elementen. Verwende Style-Sheets für die Präsentation statt Präsentations-Elementen mit Attributen.**	
3.1	Verwende verfügbare Elemente statt Grafik, um Informationen zu vermitteln.	2
3.2	Erstelle Dokumente, die mit den veröffentlichten Standards übereinstimmen.	2
3.3	Verwende Style-Sheets, um Layout und Präsentation zu definieren.	2

3.4	Verwende relative statt absoluter Einheiten in Attributen und Style-Eigenschaften.	2
3.5	Verwende Überschriften-Elemente, um Dokumentstrukturen zu vermitteln, und verwende sie entsprechend den Spezifikationen.	2
3.6	Verwende die vorgesehenen Elemente für Listen und Listenelemente.	2
3.7	Verwende die für Zitate vorgesehenen Elemente für diese und nicht etwa zum Formatieren, z.B. Einrücken.	2
4	**Verwende die vorgesehenen Elemente, um Aussprache oder Interpretation von Abkürzungen oder fremdsprachige Texte zu ermöglichen.**	
4.1	Kennzeichne deutlich jegliche Änderung der Sprache in einem Text oder einer Textalternative.	1
4.2	Gib die Bedeutung einer Abkürzung an, wenn sie das erste Mal in einem Dokument erscheint.	3
4.3	Kennzeichne die Hauptsprache eines Dokuments.	3
5	**Stelle sicher, dass Tabellen mit entsprechenden Elementen gestaltet sind, damit sie in verfügbare Browser und anderer Zugangssoftware transformiert werden.**	
5.1	Kennzeichne Spalten- und Reihenüberschriften in Datentabellen.	1
5.2	Verknüpfe Tabellenzellen und Tabellenüberschriften mit den entsprechenden Attributen, wenn Datentabelle zwei oder mehrere logische Ebenen haben.	1
5.3	Verwende Tabellen nicht zum Layout, es sei denn, sie können linearisiert werden. Biete eine alternative Version an, wenn die Tabelle nicht linearisiert werden kann.	2
5.4	Verwende keine strukturelle Elemente innerhalb von Layout-Tabellen.	2
5.5	Biete Zusammenfassungen für Tabellen an.	3
6	**Stelle sicher, dass Seiten zugänglich sind, wenn neuere Technologien von der Zugangssoftware nicht unterstützt werden.**	
6.1	Organisiere Dokumente so, dass sie ohne das angegebene Style-Sheet gelesen werden können.	1
6.2	Stelle sicher, dass Alternativen für dynamische Inhalte aktualisiert werden, wenn der dynamische Inhalt aktualisiert wird.	1
6.3	Stelle sicher, dass Seiten angezeigt werden können, wenn Script- und andere Programmierungen ausgeschaltet sind oder nicht unterstützt werden.	1
6.4	Stelle sicher, dass Event-Handler für Scripts und Applets Geräte-unabhängig sind.	2
6.5	Stelle sicher, dass dynamische Inhalte zugänglich sind, oder biete eine alternative Version an.	2
7	**Stelle sicher, dass Bewegung, Blinken, Scrollen oder Auto-Aktualisierungen angehalten oder gestoppt werden können.**	
7.1	Vermeide Bildschirmflackern.	1
7.2	Vermeide blinkende Inhalte.	2
7.3	Vermeide bewegende Inhalte auf Seiten.	2
7.4	Vermeide es, Seiten mit Auto-Aktualisierung (auto-refresh) zu erstellen.	2
7.5	Verwende Server statt Client-seitige Möglichkeiten zu Weiterleitungen.	2

8	**Stelle sicher, dass in Benutzer-Interfaces die Prinzipien der Zugänglichkeit bewahrt werden: Zugang zu Funktionen, Tastaturbedienbarkeit, Spracheingabe und –ausgabe, usw.**	
8.1	Gestalte Programmierungen wie Scripts und Applets so, dass sie zugänglich und mit den technischen Hilfsmitteln kompatibel sind.	1
9	**Verwende Features, die die Bedienung von Seitenelementen mit verschiedenen Geräten ermöglichen.**	
9.1	Biete Client- statt Server-seitige Image-Maps an, außer wenn die aktiven Bereiche nicht mit den geometrischen Formen Client-seitiger Image-Maps abgebildet werden können.	1
9.2	Stelle sicher, dass alle Elemente mit eigenem Interface Geräte-unabhängig bedient werden können.	2
9.3	Biete logische statt Geräte-abhängiger Event-Handler für Scripts an.	2
9.4	Biete eine logische Tabulatorenreihenfolge für Links, Formularfelder und Objekte an.	3
9.5	Biete Tastaturkürzel für wichtige Links (auch innerhalb Client-seitiger Image-Maps) und Formulare an.	3
10	**Verwende vorläufige Zugänglichkeitslösungen, um die Funktionalität technischer Hilfsmittel sowie älterer Browser zu gewährleisten.**	
10.1	Vermeide es, zusätzliche Pop-Up-Fenster zu erzeugen oder Fensternamen zu verändern, ohne den Benutzer vorher zu informieren.	2
10.2	Stelle sicher, dass in Formularen die Bezeichnungen, die implizit mit Formularfeldern verknüpft sind, korrekt positioniert sind.	2
10.3	Biete eine linearisierte (einspaltige) Alternative für alle Texte an, die mit Tabellen in zwei- oder mehrspaltigem Layout gestaltet wurden.	3
10.4	Biete einen textlichen Platzhalter in editierbaren Formularfeldern an.	3
10.5	Stelle sicher, dass nebeneinander positionierte Links durch nicht-verlinkte, ausdruckbare Zeichen getrennt werden.	3
11	**Verwende W3C-Spezifikationen und beachte Zugänglichkeitsrichtlinien. Wenn es nicht möglich ist, W3C-Standards einzuhalten oder diese unerwünschte Ergebnisse erzeugen, biete eine zugängliche Alternative an.**	
11.1	Verwende W3C-Technologien, wenn sie verfügbar sind und die Aufgabe lösen können, und verwende die aktuelle Version, sobald sie unterstützt wird.	2
11.2	Vermeide veraltete W3C-Technologien.	2
11.3	Biete Informationen an, die dem Benutzer den Dokumentenempfang nach seinen Präferenzen ermöglicht.	3
11.4	Biete einen Link zu einer alternativen Seite an, die W3C-Technologien verwendet zugänglich ist gleichwertige Informationen und Funktionen enthält gleichermaßen aktualisiert wird wenn nach besten Wissen und Gewissen die Barrierefreiheit nicht gewährt werden kann.	1

12	**Biete Informationen zum Inhalt und zur Orientierung an, die dem Benutzer helfen, komplexe Seiten und Elemente zu verstehen.**	
12.1	Betitele jeden Frame, damit es identifiziert und navigiert werden kann.	1
12.2	Beschreibe den Zweck eines Frames und die Beziehung zwischen Frames, wenn die Betitelung der Frames alleine dies nicht hergeben.	2
12.3	Teile große Informationsblöcke in kleinere, besser handhabbare Gruppen, wenn es natürlich und angemessen ist.	2
12.4	Verknüpfe Labels explizit mit ihren Formularfeldern.	2
13	**Biete eine klare und konsistente Navigation an – Informationen zur Orientierung, Navigationsleisten, ein Site-Map usw. –, um den Benutzer eher zu seinem gewünschten Ergebnis zu führen.**	
13.1	Benenne deutlich das Ziel eines jeden Links.	2
13.2	Biete Meta-Informationen an, um höherwertige semantische Informationen zur Site zu geben (z.B. Site-Map, Inhaltsverzeichnis).	2
13.3	Biete Informationen zum allgemeinen Layout einer Site an (z.B. Site-Map, Inhaltsverzeichnis).	2
13.4	Biete eine konsistente Navigation an.	2
13.5	Biete Navigationsleisten an, die sowohl hervorheben als auch Zugang zur Navigation geben.	3
13.6	Gruppiere themenverwandte Links, benenne die Gruppe (für die Zugangssoftware) und biete eine Möglichkeit, sie zu überspringen.	3
13.7	Biete verschieden Suchmöglichkeiten für unterschiedliche Fähigkeiten und Präferenzen an, wenn eine Suchfunktion angeboten wird.	3
13.8	Biete abhebende Informationen vor Überschriften, Absätzen, Listen usw.	3
13.9	Biete Informationen über Zusammenstellungen von Dokumenten an (z.B. Dokument bestehend aus mehreren Seiten).	3
13.10	Biete eine Möglichkeit an, mehrzeiliges ASCII-Art zu überspringen.	3
14	**Stelle sicher, dass Dokumente klar und einfach sind, um sie verständlicher werden zu lassen.**	
14.1	Verwende die klarste und einfachste Sprache, die für den Site-Inhalt angemessen ist.	1
14.2	Ergänze Text mit Grafik und Audio-Präsentationen, wenn damit das Verständnis ermöglicht wird.	3
14.3	Erzeuge einen Präsentationsstil, der Seiten-übergreifend konsistent ist.	3

Und da keine Aufgabe als abgeschlossen gelten kann, ohne das Ergebnis zu kontrollieren, hat die WAI die Überprüfung in einem Anhang formuliert. Die Hinweise hierzu sind auch deswegen besonders wichtig, weil in den meisten Standard-Browsern die Umsetzung der Barrierefreiheit nur zum Teil überprüft werden kann.

Anhang	Überprüfe die Barrierefreiheit mit automatischen Werkzeugen und persönlicher Durchsicht. Die automatischen Werkzeuge sind meist schnell und bequem, können allerdings nicht alle Aspekte der Zugänglichkeit aufdecken. Persönliche Durchsicht hilft, die Klarheit der Sprache und der Navigation festzustellen.

Synchronisationen

Enthalten Multimedia Sound und/oder gesprochenen Text, benötigen taube und schwerhörige Menschen gleichwertige Untertitel; sehbehinderte Menschen, die Bilder nicht oder nur schlecht verfolgen können, können via Audio-Deskription die Bildabfolgen nachvollziehen. Insbesondere bei längeren Multimedia-Abfolgen ist Synchronisation mit der Abfolge im Clip sehr wichtig.

Sind die Multimedia, d.h. Video und/oder Audio, mit Untertiteln und Audio-Deskription synchronisiert, erhalten fast alle Besucher den gesamten Informationswert einer Seite. Zu bedenken ist nur, dass für Surfer ohne Möglichkeit der Multimedia-Wiedergabe und für Hör- und Sehgeschädigte eine vollständige Textalternative die beste Form der Informationsvermittlung ist.

Die gängigen Standards für Multimedia

Mittlerweile lassen sich über die drei gängigen Standards für Multimedia, *QuickTime* (Apple), *Synchronized Multimedia Integration Language* (SMIL, W3C-Standard u.a. von RealNetworks unterstützt) und *Synchronized Accessible Media Interchange* (SAMI, Microsoft) Multimedia für taube, schwerhörige, blinde und sehbehinderte Menschen zugänglich machen.

Viele Video-Clips im WWW basieren auf dem QuickTime-Standard von Apple. Bereits der QuickTime3.0-Standard erlaubte sowohl Untertitel als auch Audio-Deskription in QuickTime-Movies. Ausführliche Informationen zur Gestaltung zugänglicher QuickTime-Movies findest du hier:

- The Caption Center:
  http://www.wgbh.org/caption/
  für Untertitel
- Descriptive Video Service:
  http://www.wgbh.org/dvs/
  für Audio-Deskription
- NCAM:
  http://ncam.wgbh.org/accessncam.html
  für Beispiele)

Der RealPlayer G2 von RealNetworks bietet auf SMIL-Basis ebenfalls die Möglichkeit barrierefreier Gestaltung von Multimedia. Untertitel werden z.B. mit RealText realisiert. Zur Zugänglichkeit im SMIL-Standard siehe auch:

- WAI-Richtlinien zur Zugänglichkeit im SMIL-Standard
  www.w3.org/TR/SMIL-access/ ()
- Beschreibung von RealText
  service.real.com/help/library/guides/realtext/realtext.htm)
- Werkzeuge zur Gestaltung von Multimedia nach dem SMIL-Standard
  www.justsmil.com/tools/
- Beispiele zum Herunterladen
  www.realnetworks.com/products/servers/showcase/av.html

Mit dem Windows Media Player und dem SAMI-Standard hat Microsoft einen eigenen Standard für Multimedia entwickelt, der Untertitel und Audio-Deskription ermöglicht. Unter der Adresse **www.microsoft.com/enable/sami/** findest du ausführliche Informationen über Zweck und Verwendung von Untertiteln sowie Beispiele und Demo-Versionen zum Herunterladen.

Wie im Kapitel zur Programmierung, in dem ich auf weiterführende Informationen für Java, Visual Basic usw. verweisen musste, kann ich auch hier nicht weiter ins Detail gehen. Diese Broschüre nimmt vor allem HTML unter die barrierefreie Lupe. Umfang und Vielzahl der Multimedia-Standards würde auch viele Seiten erfordern, die hier nicht zur Verfügung stehen!

Untertitel

Zugestanden, dass die Bereitstellung einer Textalternative für Multimedia – auch wenn es für manche die einzige Möglichkeit ist, Zugang zu Multimedia zu finden – nicht unbedingt die beste Methode ist, Menschen mit Behinderungen vergleichbare Erlebnisse zu ermöglichen. Es ist schon lange akzeptiert, dass gehörlose Menschen Fernseh-, Film- und Multimedia-Produktionen erst mit Untertiteln genießen können – wobei Deutschland zu den wenigen westlichen Ländern gehört, die sich bei der gesetzlichen Verankerung einer konsequenten Synchronisation von Filmen mit Untertiteln schwer tun.

Multimedia-Präsentationen wie Videos oder Animationen müssen daher nach WAI-1.4 mit entsprechenden Alternativen synchronisiert werden. Die Möglichkeiten hierzu bieten heute QuickTime,

RealPlayer G2 und Windows Media Player. Informationen über Methoden und Gestaltung habe ich im vorherigen Abschnitt angegeben.

Die kostenlose Software **MAGpie**, die du unter **ncam.wgbh.org/webaccess/magpie/** findest, verwaltet Untertitel in deinem Multimedia gleichermaßen für QuickTime, SMIL (RealPlayer) und SAMI (Windows Media Player).

Audio-Deskription

In WAI-1.3 wird die Audio-Deskription für Multimedia gefordert, solange Zugangssoftware die Textalternativen nicht automatisch lesen kann. Audio-Deskription heißt, dass das Video mit zusätzlichen akustischen Bildbeschreibungen ergänzt wird, die in Dialogpausen knappe Erläuterungen der visuellen Elemente einer Szene wie Szenerie, Bewegung, Körpersprache usw. bieten, damit der Video-Clip auch für Nicht-Sehende zugänglich wird.

Diese Richtlinie ist wohl als Übergangslösung zu betrachten, da zukünftige Zugangssoftware für das WWW mit Sicherheit in der Lage sein wird, Textalternativen in Sprache umzuwandeln – sofern sie denn bereitgestellt sind; damit entfiele die Notwendigkeit der Audio-Deskription. Da es derzeit aber keinen Standard-Browser gibt, der derartige Texte automatisch in Sprache wandelt, muss die Video-Spur von Multimedia mit einer synchronisierten Audio-Deskription ergänzt werden.

Auch diese Funktionalität findest du in den Apple-, RealNetworks- und Microsoft-Produkten resp. auf deren Websites.

In Deutschland produziert die Deutsche Hörfilm gGmbH (**www.hoerfilm.de**) Audio-Deskription für Filme im Fernsehen, für das Kino und das Theater auf Kooperationsbasis.

Ein Wort zu vektorbasierten Animationen (Flash)

Flash erlaubt den Einbau von vektorbasierten Animationen in eine Homepage. Dies erfordert, dass im Browser deines Besuchers ein entsprechendes Plug-In installiert ist – für Flash wird es in modernen Standard-Browsern automatisch installiert. Es gibt aber keine Schnittstelle für Screen-Reader.

Dadurch, dass Flash nicht nur Bilder, sonder auch Text grafisch darstellt, wird der Zugang für Blinde und Sehbehinderte versperrt. Die Internet-Seiten haben einen rein grafischen Inhalt. Dieser Umstand stellt Screen-Reader-Nutzer vor eine unlösbare Aufgabe: Sprachausgaben schweigen, und Braille-Zeilen liefern nicht einmal einen einzigen Punkt.

Flash bietet aber auch die Möglichkeit, Sound darzubieten. Benutzt du Flash, solltest du diese Möglichkeit einsetzen und deine Präsentation mit Hörbarem ergänzen. Du könntest auch soweit gehen, Flash nur zu akustischen Zwecken einzusetzen, um eine kurze Inhaltsangabe einer Seite zu geben. Grundsätzlich ist Flash aber nicht zugänglich und sollte daher ausschließlich für Animationen, keinesfalls aber für textliche Informationen benutzt werden. Auch Bilder ohne Bewegung sind – aus Sicht der Barrierefreiheit – mit "herkömmlichen Mitteln zu gestalten."

Anmerkung zur Spracheingabe

Eine interaktive, rein akustische Seite ist mir nicht bekannt, aber die Sprache ist für Menschen wohl ein intuitiveres Kommunikationsmittel als Tastatur, Maus und Bildschirm. Sicherlich wird in der Zukunft interaktive Sprache im Web Fuß fassen. Es sollte daher immer darauf geachtet werden, auch hier alternative Ausgabe- und Eingabemöglichkeiten anzubieten.

Wie bei der auditiv bedingten Barriere besteht auch für die Spracheingabe bisher kaum eine Anwendung im Netz, wo die Spracheingabe für Menschen mit mehr oder weniger gewichtigen Sprachstörungen als Barriere gelten könnte. Die Spracheingabe ist sicherlich auf dem Vormarsch, aber es wird noch Jahre dauern, bis interaktive Sprachein- und -ausgabe auf breiter Basis als Mittel der Kommunikation zwischen Mensch und Maschine eingeführt werden kann. Das WWW als multimediales Kommunikationsmittel wird aber zwangsläufig Sprache als Medium einsetzen, denn das ist unsere eigene natürliche Form der Kommunikation. Für bestimmte Anwendungen ist es durchaus vorstellbar, dass die Sprache dem geschriebenem Text vorgezogen wird, beides sollte aber möglich sein.

Und wenn das alles nicht hilft ...

Solltest du die Richtlinien der WAI trotz aller Bemühungen nicht umsetzen können, und sind deine Seiten weiterhin nicht zugänglich, bleibt eigentlich nur noch die Möglichkeit, eine parallele Nur-Text-Version der Website anzubieten,

- die den Standards des W3C entspricht,
- zugänglich ist
- den gleichen informativen und funktionalen Wert wie das "Original" hat und
- im selben Umfang aktualisiert wird wie das "Original"

Das klingt nach viel Arbeit. Ist es im Prinzip auch, wenn deine Seiten nicht mit einem Werkzeug generiert werden. Eine Website wird zu zwei parallelen Websites! Überlege daher genau, ob du nicht an anderer Stelle ein Paar Rädchen drehst, um deine Site zugänglich zu machen, und ob du nicht eventuell auf das eine oder andere Gestaltungselement verzichten kannst.

Die WAI resp. W3C schlägt vor, dass du mit dem **LINK**-Element arbeitest, um die Richtlinien 11.4 sowie 6.5 einzuhalten. Das **LINK**-Element wird bislang kaum gebraucht und wurde von Standard-Browsern wie Microsoft Internet Explorer und Netscape Navigator lange nur für die Einbindung von Style-Sheets unterstützt. Richtig eingesetzt kann dieses Element aber für Suchmaschinen, Content-Management-Systeme und weitere Indexierungssoftware sehr nützlich sein, um Websites im Aufbau und wechselseitigen Abhängigkeiten darzustellen, z.B. in einer Site-Map. So kannst du mit **rel="prev"** die vorherige Seite, mit **rel="next"** die nachfolgende Seite, mit **rel="chapter"** oder **rel="section"** Beziehungen zu übergeordneten Seiten angeben und somit logische Zusammenhänge erstellen, die theoretisch von Browsern in der Symbolleiste abgebildet werden (in der Praxis werden diese Verknüpfungen aber leider noch nicht angezeigt). Mit **rel="alternate"** kannst du auch eine Beziehung zu einer oder mehreren alternativen Dateien anbieten, die den Zugänglichkeitsrichtlinien entsprechen. Mit dem **media**-Attribut kannst du sogar ergänzend bestimmen, für welche Ausgabegeräte deine alternativen Seiten am besten geeignet sind, wobei auch diese Einstellungsmöglichkeit in Standard-Browsern fehlt. Das W3C hat im HTML4.01-Standard folgende Werte für das **media**-Attribut festgelegt:

screen	Bildschirm
tty	Bildschirme mit festgelegter Textgröße
tv	Fernseh-Bildschirm
projection	Projektor
handheld	Handheld-Geräte
print	Druckvorschau
braille	taktile Ausgabe (Braille-Zeile)
aural	akustische Wiedergabe

Das Beispiel zeigt, wie du eine Anweisung für den Browser deines Besuchers angibst:

```
<HEAD>
  <TITLE>Virtuelle Welten</TITLE>
  <LINK rel="alternate"
href="index_nurtext.htm" title="Nur-
Text-Version" media="aural, braille,
tty">
</HEAD>
<BODY> ... </BODY>
```

Natürlich hast du damit nur die Einbindung einer alternativen Seite erzielt! Bei der Erstellung dieser Seiten solltest du lediglich die Inhalte sowie Navigationsmechanismen berücksichtigen.

Selbstverständlich kannst du auch weitere Elemente einfügen, wenn diese barrierefrei sind. Wichtig ist dabei, dass die Nur-Text-Version im gleichen Maße gepflegt wird wie das ja nicht zugängliche Original. Dies gilt insbesondere für dynamische Inhalte (WAI-6.2).

Erstellst du Inhalte und Funktionen dynamisch mit Server-seitigen Werkzeugen, ist es relativ leicht, eine zweite (parallele Site anzubieten. Hast du aber viele statische Seiten erstellt, die dynamische Elemente mit wichtigen Inhalten und Funktionen enthalten, ist es sehr aufwändig, diese Elemente zugänglich zu machen. Eine parallele Alternative muss eigentlich von vornherein geplant sein.

Wie du z.B. ein Frameset zugänglich machst, habe ich bereits beschrieben. Auf **www.dvbs-online.de** siehst du ein einfaches Beispiel, wie du deinen Besuchern eine Frames-Version und eine Version ohne Frames anbietest, ohne dass das doppelte Arbeit bedeutet. Die Inhalte enthalten hier in jedem Navigationsbereich u.a. einen Link zur Frames-Startseite und zur Startseite ohne Frames. Nur von der Navigationsleiste im Frameset wird ein **target** aufs Inhaltsfenster gesetzt, ansonsten findet jegliche Navigation (innerhalb des rechten Frames oder ohne den Frameset) unabhängig vom Frameset statt. Nur die Links zu den beiden Startseiten benötigen einen **target="_top"**, um Verschachtelungen zu vermeiden.

Was du noch für Barrierefreiheit berücksichtigen solltest

Bisher hast du im Wesentlichen über Barrieren gelesen, auf die du unbedingt achten musst, wenn du deine WWW-Seiten zugänglich machen willst. Es gibt aber weitere Anforderungen der WAI, die du berücksichtigen solltest, um auch kleinere Barrieren zu umgehen. In diesem Kapitel befasse ich mich zunächst mit Style-Sheets, die für den Webzugang ohne Standard-Bildschirmeinstellungen eine besondere Bedeutung haben. Dann geht es um einige weitere Anforderungen an HTML, etwa zum korrekten Aufbau von Formularen oder wie du deine Webseite gestaltest, ohne dass der Benutzer eine Maus benutzen muss.

Layout mit Style-Sheets

In den Anfangstagen des Web war die Gestaltung von Seiten durchschaubar: Die Hypertext Mark-Up Language wurde zur simplen Strukturierung von (meist wissenschaftlichen) Dokumenten benutzt, die der Browser entsprechend darstellte. Überschrift vom Typ **H1**, **H2**, ... wurden (und werden immer noch) mit abgestuften Größen dargestellt, Absätze (**P** = paragraph) werden als Kasten erzeugt, und der Browser fügt automatisch Abstände zum vorherigen und nachfolgenden Element ein; zitierte Textpassagen wurden statt mit dem **P**-Element mit **BLOCKQUOTE** gekennzeichnet, damit der Browser sie erkennen und kursiv oder in anderer Weise abgesetzt darstellen konnte. Über die Jahre wurden viele der strukturellen Elemente zu Layout-Zwecken missbraucht. Das liegt an den Browsern, die sehr freizügig mit HTML umgehen, aber natürlich liegt es auch an Webdesignern und Grafikern, die das Grundprinzip von HTML, nämlich die Strukturierung von Informationen, immer noch nicht ganz verstehen.

Für die Gestaltung sind Style-Sheets vorgesehen. Weil aber bisher

- gängige Browser elementare Fehler bei der Interpretation von Style-Sheets machten
- die Ergebnisse in verschiedenen Browsern teilweise erschreckend unterschiedlich waren
- die Gestaltungs-Werkzeuge Style-Sheets nur begrenzt unterstützen
- das Verstehen von CSS und somit auch deren Einsatz zumindest eine gewisse Neigung zum Programmieren vom Gestalter erfordert

wird CSS bis heute wenig und/oder falsch eingesetzt. Ironie der Geschichte ist, dass bereits der erste am CERN "erfundene" Browser (Mosaic) benutzerdefinierte Formatvorlagen - eben Style-Sheets – beherrschte.

Immer häufiger erfolgt der Zugang zum WWW ohne den "traditionellen" Bildschirm – denke an die vielen mobilen Geräten mit Internetzugang, die heute auf dem Markt sind. Eine Organisation von Seiteninhalten, z.B. mit Layout-Tabellen, ist am Bildschirm zunächst einmal in Ordnung – aber sie macht nicht nur Benutzern mit kleinen Displays das Leben schwer, sondern auch den Benutzern von Screen-Readern und anderen, die mit geringen Bildschirmauflösungen arbeiten. Mit der Trennung von Inhalt und Layout bietest du deine Inhalte allen Besuchern in strukturierter Form an und überlässt es der Zugangssoftware, ob das Layout angezeigt wird oder nicht. Die konsequente Anwendung von HTML4.0 und CSS bietet hier alle Möglichkeiten.

Über die Vorteile von Style-Sheets konntest du bereits auf Seite 24 lesen.

Mal mit und mal ohne CSS

Die Trennung von Inhalt und Layout habe ich nun oft genug betont. Diese Forderung an das Webdesign geht auf WAI-3.3 zurück und ist – wenn deine Website bereits erstellt worden ist – ehrlich gesagt nicht ohne weiteres umzusetzen,. Dennoch solltest du überlegen, wie du deine Vorlagen anpassen kannst, damit Layout und Präsentation zukünftig mit CSS bestimmt werden können. Sollten deine Seiten ohne Style-Sheets gestaltet worden sein oder sind Style-Angaben in jeder HTML-Datei statt in einer externen Datei enthalten, kommt die Anpassung an barrierefreie Style-Sheets einer kompletten Überarbeitung gleich. Du solltest dich also auf die Vorlagen konzentrieren und Anpassungen deiner Seite erstmal für die wichtigen Seiten deiner Website vornehmen, wie die Startseite und andere viel besuchte Seiten.

Beim Einsatz von Style-Sheets musst du jedoch zwei Punkte beachten:

- Die Techniken, die du einsetzt, sollten auf den neuesten Stand sein
- Deine Seiten müssen auch ohne Unterstützung von Style-Sheets nachvollziehbar sein.

Das in WAI-11.2 formulierte Gebot, aktuelle W3C-Standards zu verwenden gilt natürlich nicht nur für HTML, sondern eben auch für CSS und andere Standards. Bei CSS gibt es z.B. bereits die "CSS Level 2"-Standardisierung (CSS2). Im Gegensatz zu der früheren "Level 1"-Norm, in der der Webgestalter immer das letzte Wort hatte, wird in "Level 2" diese Gestaltungsmöglichkeit dem Besucher übertragen – gibt dieser also seine eigenen Style-Angaben in seinem Browser an, haben diese den Vorrang gegenüber Vorgaben auf einer WWW-Seite. Diese Umkehrung der Layout-Vorgabe ist wichtig für sehbehinderte Surfer, die etwa bei bestimmten Kombinationen von Text- und Hintergrundfarben große Leseschwierigkeiten haben und deshalb eigene Text- und Hintergrundfarben festsetzen. Gleiches gilt für Schriftgröße und Schriftart. So wird dann dein erstelltes Layout mit wenigen Klicks zu Nichte gemacht. Setzt ein Besucher z.B. eine größere Schriftart ein, könnte das Style-Sheet deines Besuchers folgende Zeilen enthalten:

```
P { font-size:18pt ! important }
```

wobei hier alle als Absätze gekennzeichneten Texte mit der vom Benutzer vorgezogenen Größe 18pt dargestellt werden. Ausschlaggebende Stelle ist dabei der **!important**-Operator, da er dem Browser die höhere Priorität gegenüber Style-Angaben der Seite mitteilt.

Viele Erneuerungen sind zwar eher für den User bzw. die Zugangssoftware gedacht, können aber auch dir in deiner Arbeit helfen. Was bei der Gestaltung zugänglicher Seiten in CSS2 nützlich ist, ist der Wert **inherit**, der jeder CSS-Angabe zugewiesen werden kann. Damit gibst du an, dass ein bestimmtes Element, z.B. das **P**-Element, den Wert eunes übergeordneten Elements "erben" soll, was z.B. das **BODY**-Element, aber auch jedes andere Element sein könnte. Das folgende Beispiel zwingt alle Elemente, die Textfarbe weiß und Hintergrundfarbe blau anzunehmen:

```
/* Erzeugt weisse Vordergrundfarbe
(Text) und blaue Hintergrundfarbe
für das BODY-Element */
BODY { color:#fff; background:#00d;
}
/* Bewirkt, dass 'color' und
'background' von jedem anderen
Element ('*' = alle Elemente)
innerhalb des BODY-Elements geerbt
werden */
* { color:inherit;
background:inherit; }
```

Vielleicht kennst du Farben-Codes nur als Kombination aus 6 hexadezimaler Ziffern; mit HTML4 sind auch Zahlentripel zulässig, wobei jede Zahl doppelt bewertet wird. **#00d** entspricht dann **#0000dd**.

Weitere CSS2-Features sind:

- Angabe von Systemfarben (für **color**, **background-color**, **border-color** und **outline-color**) und Systemschriftarten (für **font**-Angabe in CSS – nicht mit dem **FONT**-Elemnt in HTML zu verwechseln); damit kannst du die vorgezogene Schriftfarbe oder Hintergrundfarbe des Benutzers immer berücksichtigen.
- Dynamische Rahmen mit der **outline**-Eigenschaft, die für den sehbehinderten Benutzer das Erkennen beispielsweise von Links durch eine Umrahmung erleichtern. Das folgende Beispiel zeigt, wie ausgewählte Element einen schwarzen Rahmen und aktive Elemente einen roten Rahmen erhalten:

```
:focus  { outline: thick solid
#000 }
:active { outline: thick solid
#f00 }
```

Worauf du aber bei der Gestaltung deiner Seiten mit Style-Sheets auf jeden Fall achten solltest, ist, dass die Seiteninhalte auch dann lesbar sind, wenn der Benutzer ganz ohne Unterstützung von Style-Sheets durch das WWW surft. Mehr hierzu auf Seite 25. Ähnlich wie bei der Vorgabe, Informationen nicht nur durch Farbe zu vermitteln (S. 16), können auch Textumrahmungen oder andere Linien unter vom Benutzer abhängigen Umständen nicht angezeigt werden. Du könntest z.B. zur Hervorhebung bestimmter Absätze folgendes einsetzen:

```
...
<STYLE type="text/css"><!--
  .hervorhebung { border-bottom:2px
solid #f00; border-top:2px solid
#f00 }
--></STYLE>
...
<P class="hervorhebung">Ein
Absatz</P>
```

um optisch eine bestimmte Bedeutung zu vermitteln. Ohne Style-Sheet-Unterstützung erscheint dieser Absatz aber so wie jeder andere. In diesem speziellen Fall lohnt es sich, das strukturelle **HR**-Element (horizontal rule = horizontale Begrenzung) zu Hilfe zu nehmen.

Relative statt absoluter Maßeinheiten

Für Sehbehinderte sind manche Seiten nicht zu lesen, nicht nur weil die Schrift zu klein ist, sondern auch weil die Schriftgröße nicht geändert werden kann. Dies ist immer dann der Fall, wenn Schriftgrößen mit Punkten (pt) angegeben und Seiten mit älteren Browsern angezeigt werden. Das W3C hat sowohl mit HTML als auch mit CSS Möglichkeiten für Angaben geschaffen, die zu den benutzereigenen Browser-Einstellungen relativ definiert sind. Auf dieser Weise können die persönlichen Präferenzen bei der Bestimmung der Schriftgröße immer berücksichtigt werden. In HTML ist die Angabe nur über das nicht mehr empfohlene **FONT**-Element möglich. In CSS gibt es gleich mehrere Möglichkeiten, allen voran die Größe **em**, die der benutzereigenen Schriftgröße entspricht. Neben **em** ist auch eine Angabe in Prozent möglich. Was diesen Größenangaben gemein ist, ist der Bezug auf die im Browser angegebene Standard-Schriftgröße. Bei der Mehrzahl der Surfer wird diese Größe in etwa 10pt sein. Entsprechend würde 100% als Schriftgröße 10pt ergeben. Die Schriftgröße **1em** (1em = die Breite des Buchstaben "m" und meist identisch mit der Schriftgröße einer Schriftart) ergäbe ebenfalls 10pt beim Durchschnitts-Surfer. Für Sehbehinderte, die eine größere Standard-Schriftgröße in ihrem Browsereinstellungen eingegeben haben, z.B. 16pt, bedeutet die Schriftgröße 100% bzw. 1em, dass die Webinhalte nach den von ihnen vorgezogenen Einstellungen angezeigt werden.

Gerade bei älteren Browsern wie dem Microsoft Internet Explorer 5.0x und Netscape 4.x ist die Unterstützung von CSS2-Funktionen zur Unterdrückung von solchen Gestaltungsvorgaben nicht implementiert. Die Benutzung relativer Schriftgrößen löst auch das Problem, ob ein Besucher mit einer extrem hohen Bildschirmauflösung und damit auch deutlich verkleinerten Schriftgrößen Texte ebenfalls gut lesen kann. Benutze also skalierbare CSS-Angaben wie:

```
P { font-size:0.8em }
```

statt fester Werte wie

```
P { font-size:8pt }
```

Die Verwendung relativer Maßeinheiten wird in WAI-3.4 verlangt und bezieht sich auf HTML und CSS. CSS erlaubt auch, bei der absoluten Positionierung von **DIV**-Elementen relative Breiten und Abstände einzusetzen. Der einzige Fall, wo feste Werte ruhig verwendet werden können, ist eine genaue Definition des Ausgabemodus (siehe Seite 49). Stellst du z.B. Visitenkarten zum Ausdruck bereit, kannst du für die Schriftgröße eine Klasse **visitenkarte** wie folgt definieren:

```
.visitenkarte { font-size:8pt }
```

Zur sehbehindertengerechten Gestaltung von WWW-Seiten gehört auch das Verwenden von relativen Größenangaben für Tabellen und Frames. Dies sind aber keine CSS-spezifischen Angaben, sondern Teil des HTMLs. Die relativen Angaben in Prozent dienen der Berücksichtigung geringer Bildschirmauflösungen. Statt z.B. Tabellenzellen mit **width="200"** zu formatieren, solltest du lieber **width="25%"** angeben. Das gleiche gilt für Frames. Die Gesamtbreite für Tabellen und Frames sollte jeweils 100% ergeben. Bei Frames sollte die Gesamthöhe ebenfalls 100% ergeben.

Wenn die Breite der Tabelle ausschließlich über Zellen (**TD**-Element) statt der gesamten Tabelle definiert wird, kann der Browser sowohl bei geringen Auflösungen als auch für den Ausdruck auf einem Drucker die Tabellenbreite anpassen. Ist die Tabelle aber fest vorgegeben und dadurch nicht für das Ausgabemedium geeignet, wird die rechte Seite des Inhalts nicht angezeigt bzw. abgeschnitten.

Du wirst feststellen, dass Netscape 4.x bei relativen Schriftgrößen in bestimmten Fällen etwa in Tabellen aussteigt und sonderbare Ergebnisse liefert. Dies ist ein "Known Bug" und kann nur mit Netscape's eigenen "Javascript Style-Sheets" (JSS) behoben werden. JSS ist eine Kombination aus JavaScript und CSS und wird nur von den Netscape Browsern ab der Version 4.0 interpretiert. Gibst du z.B. folgende Angaben in einem Style-Sheet vor:

```
TD { font-size:0.8em }
```

musst du für das gleiche Ergebnis in Netscape-4.xx-Browsern in einem JavaScript im **HEAD**-Bereich oder einer externen JavaScript-Datei folgende Zeilen einfügen:

```
if ((navigator.appName ==
"Netscape") &&
(parseInt(navigator.appVersion) <
5)) {
 document.write("<STYLE
TYPE='text/javascript'><!--
tags.TD.fontSize='10pt'; --
></STYLE>");}
```

Mehr Informationen zu JSS findest du hier:
developer.netscape.com/docs/manuals/index.html?content=javascript.html

wo du das Kapitel über JSS aus dem Buch "The JavaScript Bible" als PDF-Datei downloaden kannst.

Allgemeine Textformatierung und -positionierung

Die Richtlinie in WAI-3.3 (Layout mit CSS statt HTML) bedeutet auch, dass du auf kleine Tricks in HTML zur Erzeugung von optischen Wirkungen verzichten solltest. So wird z.B. oft das Non-breaking Space (** **) für Abstände benutzt. Für Screen-Reader können diese Leerzeichen problematisch werden, da sie – im Gegensatz zu echtem Leerraum – verarbeitet werden, worauf dann mehrmals das Wort "Leerraum" erklingt. Benutzt du ein oder zwei solcher Leerzeichen hintereinander, ist das sicher nicht schlimm. Es gibt nun aber mit CSS sehr elegante Lösungen, Abstände zentral in einer externen CSS-Datei zu steuern, etwa mit **text-indent** (Einzug links und rechts – verwende dafür keinesfalls das **BLOCKQUOTE**- oder **UL**-Element), **letter-spacing** (Zeichenabstand) oder **word-spacing** (Wortabstand).

So solltest du z.B. eine Begrüßung nicht mit **H A L L O** definieren, was auf dem Bildschirm als das Wort "H A L L O" mit vergrößertem Zeichenabstand erscheint, sondern die entsprechende CSS-Eigenschaft auf das Wort anwenden.

Auch zwei Pseudo-Eigenschaften in CSS, die zwar kaum unterstützt werden, sind interessant als Alternative zu HTML, nämlich **:first-letter** (erster Buchstabe einer Textfolge anders formatieren) und **:first-line** (erste Zeile eines Blockelements anders darstellen)

Layout ohne Tabellen

In diesem Abschnitt steht CSS ebenfalls im Zentrum, wenn auch die Verwendung von Tabellen für das Layout mit HTML erfolgt. Da Tabellen problematisch sind, ist es gut, dass es sehr gute Möglichkeiten gibt, das Layout völlig ohne sie zu gestalten und statt dessen CSS in Verbindung mit dem **DIV**-Element einzusetzen. Aber vorab: die CSS-Techniken, die hier angesprochen werden, betreffen grundlegendes Webdesign. Die Idee, das Layout einer Seite nicht mit Tabellen, sondern mit dem **DIV**-Element zu gestalten, gehört eindeutig in die Planung der Website. Die Überarbeitung der Website, um sie an die in diesem Abschnitt angesprochenen Anforderungen der WAI anzupassen, wäre ein komplettes Redesign. Daher gehe ich zunächst darauf ein, wie Layout-Tabellen barrierefrei gestaltet werden könnten; und dann stelle ich einige Möglichkeiten vor, wie du mit CSS ähnliche Effekte erzielst.

Lassen sich deine Tabellen linearisieren?

Benutzt du Tabellen für das Layout, könnte das Grundgerüst für deine Seite wie folgt aussehen:

```
<TABLE>
  <TR>
    <TD colspan="2">Kopfbereich der
Seite (Überschrift)</TD>
  </TR>
  <TR>
    <TD
width="30%">Navigationselemente</TD>
    <TD width="70%">Inhalte</TD>
  </TR>
</TABLE>
```

Womöglich hast du innerhalb der einzelnen Tabellenzellen weitere Tabellen verschachtelt. Prinzipiell solltest du solche Layouts vermeiden oder zumindest sicherstellen, dass diese Tabellen linearisiert gelesen werden können. Das Problem bei Tabellen ist, dass Screen-Reader bestimmte Algorithmen befolgen, um die Inhalte einzelner Zellen auszugeben. Du kannst dir das etwa so vorstellen: Der Screen-Reader bearbeitet Layout-Tabellen in der Regel Zeile für Zeile und gibt ihre Zellen von links nach rechts als einzelne Absätze aus. Eine sinnvolle und systematische Anordnung der Zelleninhalte erleichtert dem Screen-Reader-Benutzer das Lesen. Es ist auch wichtig, dass du die Inhalte der Tabellenzellen in entsprechenden Überschriften, Absätze, Listen usw. formatierst, damit sie nach der Linearisierung Sinn machen. Auch wenn Screen-Reader Tabellen immer besser lesen können, bleiben sie nach wie vor problematisch. Auch ältere grafische Browser und Textbrowser haben nicht die Funktionalität, alle Tabelleninhalte richtig darzustellen, d.h. auch die Rückwärts-Kompatibilität für Browser, die nach HTML3.2 interpretieren, sollte sicher sein.

Eine Tabelle, die folgendermaßen aussieht:

Pressemeldungen	**Ausstellung in**
Informationen für	**unsere Münchener**
Medien-Vertreter	**Filiale** – jetzt auch virtuell. Werke des Künstlers Karl Antz

könnte von einem Screen-Reader wie folgt ausgegeben werden:

"Pressemedungen Ausstellung in Informationen für unsere Münchener Medien-Vertreter Filiale – jetzt auch virtuell Werke des Künstlers Karl Antz"

Die Linearisierung von Tabellen ist sehr simpel nachzustellen. Lese die Inhalte im Quellcode ohne die Tabellen-Elemente einfach von oben nach unten. Es gibt aber auch einige Werkzeuge, die dir dabei helfen können, etwa TabLin: **`www.w3.org/WAI/Resources/Tablin/`**

Verwende keine logischen Datenstrukturen für Layout-Tabellen

Auf Seite 29 ging ich ausführlich auf die logischen Strukturen ein. Außerdem gab ich Beispiele für HTML-Elemente wie **`CAPTION`** und **`TH`**. Diese Elemente sind zwar wichtig für Screen-Reader, damit Tabellen vom Benutzer verstanden werden können, aber für Layout-Zwecke machen sie keinen Sinn, da z.B. das **`CAPTION`**-Element Tabellen ebenfalls eine optisch sichtbare Überschrift gibt. Abgesehen davon verbietet WAI-5.4 die Anwendung solcher struktureller Elemente für Layout-Tabellen, weil sie den Benutzer verwirren können.

Lösungen mit CSS

Mit Style-Sheets kannst du eine gleichermaßen ansprechende wie auch barrierefreie Alternative für Layout-Tabellen anbieten. Style-Sheets dienen dem Layout, HTML der Strukturierung! Auch wenn dieses grundlegende Prinzip seit der Erfindung des World Wide Web nur wenig beachtet wird, hat sich daran nichts geändert. Benutzt du Tabellen für das Layout, ist das so, als schriebst du deine Briefe mit Excel. Benutzt du aufwändige Tabellen für Layout-Zwecke, sind Entwicklung und Pflege der Seiten ebenfalls aufwändig. Mit dem DIV-Element kannst du diese Arbeiten und auch die Dateigröße reduzieren. Das W3C gibt zum Aspekt der Dateigröße an, dass sich die Ladezeiten von HTML-Dateien um den Faktor 3 verkürze.

In den folgenden Beispielen erfolgt die Zuweisung der Style-Angaben über **`id`**. Benutzt du jedoch Formatierungen mehrmals auf einer Seite, solltest du **`class`** statt **`id`** verwenden. Eine Klasse wird mit einem Punkt (**`.`**) statt mit einer Raute (**`#`**) im Style-Sheet eingeleitet.

Die Quelldateien für die Screenshots dieses Heftes findest du im Internet unter der Seite des Heftes.

Es empfiehlt sich, diese Dateien downzuloaden, da sie wichtige zusätzliche Informationen enthalten.

Beispiel mit drei Spalten (1)

Eine sehr elegante Lösung: drei Spalten mit einer weichen (d.h. sich an die Browsereinstellungen des Benutzers anpassenden) Spalte. Das Layout ist leicht zu verstehen und sehr flexibel anzuwenden – allerdings musst du aufpassen, dass die Zeilen in den fixen Spalten gut umbrechen, da bei langen Wörtern und vergrößerter Schrift diese Spalten verbreitert werden und die mittlere Spalte überlagern.

Layout ohne Tabellen: Beispiel mit drei Spalten (1)

die linke Spalte

```
#spaltelinks {
position:absolute;
left:0px;
top:69px;
width:200px;
border:1px solid;
        }
```

Diese Spalte wird mit `<div id="spaltelinks">` erzeugt, nachdem `spaltelinks` im Style-Sheet definiert wurde. Es handelt sich um

die mittlere Spalte

```
#spaltemitte {
margin-top:-1px; /* fummel */
margin-left:199px;
margin-right:199px;
border:1px solid;
voice-family: "\"}\"";
voice-family:inherit;
margin-left:201px;
margin-right:201px;
        }
html>body #spaltemitte {
margin-left:201px;
margin-right:201px;
border:1px solid;
        }
```

Im Gegensatz zu den äusseren beiden Spalten, die feste Abstände und Breiten haben, ist diese Spalte

die rechte Spalte

```
#spalterechts {
position:absolute;
right:-1px;
top:69px;
width:200px;
border:1px solid;
voice-family: "\"}\"";
voice-family:inherit;
        }
html>body
    #spalterechts {
right:0px;
        }
```

Weitere Beispiele:

Beispiel mit zwei Spalten (ALA)

Dieses Beispiel ist wohl die am einfachsten anzuwendende Lösung, die im Prinzip lediglich ein **`float:left`** erfordert.

Layout mit CSS: Beispiel mit zwei Spalten (ALA)

"freier" Text

Diese Spalte wird mit `<div id="spalterechts">` eingeleitet, wobei `spalterechts` keine Formatierungen enthält.

Weitere Beispiele:

- Beispiel mit drei Spalten (1).
- Beispiel mit zwei Spalten (ALA).
- Beispiel mit vier Spalten.
- Beispiel mit drei Spalten (2).
- statisches Beispiel.
- Beispiel mit `float`.

Entspricht die Seite den W3C-Normen?

W3C HTML 4.01

Übersicht aller Beispiele aus dem KnowWare-Heft.

Positionsrahmen mit `float`

```
#spaltelinks {
float:right;
width:67%;
border-left:1px solid;
border-bottom:1px solid;
margin-right:15px;
padding-bottom:20px;
        }
```

Auch wenn diese Spalte `spaltelinks` heisst, steht sie rechts. Mit der Änderung der Positionsangabe von `float:left` auf `float:right` kann man das auch sehr schnell ändern und das Layout einer Website in Sekundenschnelle anders aussehen lassen.

Diese sehr einfache Technik, Spalten mit CSS zu erzeugen, wird von Jeffrey Zeldman in seinem ALA-Artikel "A Web Designer's Journey" auf http://www.alistapart.com/stories/journey/ ausführlich dokumentiert.

Im Prinzip erfordert diese Technik nur die Angabe `float: ...` für ein `DIV`-Element sowie eine Breitenangabe. Der rechte Text wird dann in einem Kasten der angegebenen Breite (67% der Fensterbreite) dargestellt und nachfolgender Text (im Quellcode nach dem `DIV`-Element) fließt um diesen Kasten herum. Je länger der Text im `DIV`-Element ist, desto länger die Spalten. Sobald der freie Text jedoch länger wird als der Text in dem Kasten, wird der freie Text auch die volle Bildschirmbreite ausfüllen.

Beispiel mit vier Spalten

Diese Technik bietet sich für eine beliebige Anzahl Spalten an, wobei du die Breite eines Bildschirms mit geringer Auflösung bedenken solltest, bevor du vier und mehr Spalten auf einer Seite positionierst. Die Technik hat den "Nachteil", dass feste Spaltenbreiten sehr schnell zu Problemen führen. Da sie auf Prozente setzt, die in jedem Browser anders berechnet werden, ist Pixel-genaue Gestaltung unmöglich.

die linke Spalte

```
#spaltelinks {
position:absolute;
left:1%;
width:20%;
top:69px;
border:1px solid;
        }
```

Diese Technik kann theoretisch für eine beliebige Anzahl Spalten angewandt werden.

Solange keine Rahmen und/oder verschiedene Hintergrundfarben für die Spalten gewünscht werden, ist diese Technik sehr praktisch.

mitte links

```
#spaltemittelinks {
position:absolute;
left:22%;
width:28%;
top:69px;
border:1px solid;
        }
```

Ein Nachteil dieser Methode ist, dass die Style-Sheets sehr schnell kompliziert werden, sobald eine Spalte mit fester Breite angegeben werden soll.

Diese Methode zur Erzeugung mehrerer Spalten ist stark abhängig von %-Angaben. Die verschiedenen Browser berechnen die %-Angaben ganz unterschiedlich, und es ist deswegen sehr schwierig, Pixel-genaue Darstellungen zu erreichen.

mitte rechts

```
#spaltemitterechts {
position:absolute;
left:51%;
width:28%;
top:69px;
border:1px solid;
        }
```

Weitere Beispiele:

- Beispiel mit drei Spalten (1).
- Beispiel mit zwei Spalten (ALA).
- Beispiel mit vier Spalten.
- Beispiel mit drei Spalten (2).
- statisches Beispiel.
- Beispiel mit `float`.

Entspricht die Seite den **W3C-Normen**?

die rechte Spalte

```
#spalterechts {
position: absolute;
left:80%;
width:19%;
top:69px;
border:1px solid;
        }
```

Mehr Beispiele bei Stytik Majik: **www.statikmajik.**

Beispiel mit drei Spalten (2)

Eine einfachere Technik als die 4-spaltige Methode. Sie verwendet **`float:left`** statt der **`position`**-Eigenschaft und benötigt Prozentangaben für jede Spalte. Bei schmalen Fenstern brechen Spalten um.

die linke Spalte

```
#spaltelinks {
width:33%;
float:left;
padding-bottom:10px;
        }
```

Übersicht aller Beispiele aus dem KnowWare-Heft.

die mittlere Spalte

```
#spaltemitte {
width:34%;
float:left;
padding-bottom:10px;
        }
```

Diese Technik ist etwas einfacher und wahrscheinlich auch praktischer als das **Beispiel mit 4 Spalten**. Anstatt mit `position:absolute` werden die Spalten mit `float:left` erzeugt.

Die Methode ist stark abhängig

die rechte Spalte

```
#spalterechts {
width:33%;
float:left;
padding-bottom:10px;
        }
```

Weitere Beispiele:

- **Beispiel mit drei Spalten (1).**
- **Beispiel mit zwei Spalten (ALA).**
- **Beispiel mit vier Spalten.**
- **Beispiel mit drei Spalten (2).**

Statisches Beispiel

Aus Sicht der Barrierefreiheit nicht das beste Beispiel, gehört das Layout mit fixen Spalten jedoch sicher auch dazu! Die drei statischen Spalten werden mit der Kopfzeile zusammen in einem weiteren **DIV**-Element eingebettet und dieses zentriert.

die linke Spalte

```
#spaltelinks {
width:175px;
padding:0px;
float:left;
        }
```

Das umschließende DIV-Element wurde wie folgt definiert:

```
BODY {
text-align:center;
        }
    ...
#rahmen {
width:650px;
margin-right:auto;
margin-left:auto;
margin-top:10px;
padding:0px;
text-align:left;
}       }
```

die mittlere Spalte

```
#spaltemitte {
width:300px;
padding:0px;
float:left;
        }
```

Ein weiteres Beispiel mit drei Spalten, jedoch diesmal mit starren Spaltenbreiten. Die vier DIV-Kästen (Kopfzeile und drei Spalten) sind alle innerhalb eines 5., zentrierten DIV-Elements eingebunden.

Ein wesentlicher Nachteil dieser Methode ist, dass Bilder oder Texte in einer der drei Spalten nicht breiter als die Spalte selbst sein dürfen. Sollte dies passieren, dann wird das Layout völlig zerstört - abhängig vom Browser!

Übersicht aller Beispiele aus dem KnowWare-Heft.

die rechte Spalte

```
#spalterechts {
width:175px;
padding:0px;
float:left;
        }
```

Weitere Beispiele:

- **Beispiel mit drei Spalten (1).**
- **Beispiel mit zwei Spalten (ALA).**
- **Beispiel mit vier Spalten.**
- **Beispiel mit drei Spalten (2).**
- **statisches Beispiel.**
- **Beispiel mit `float`.**

Entspricht die Seite den W3C-Normen?

Ein sehr einfaches Layout mit einer eingebetteter Navigation.

die linke Spalte

```
#spaltelinks {
border:1px solid;
        }
```

Dies ist eine sehr einfache Spaltentechnik, die wiederum lediglich die `float`-Angabe für die rechte Spalte benötigt.

Es gibt zwei Dinge, die beachtet werden müssen:

1. Die rechte Spalte muss im Wuelltext vor der linken Spalte erscheinen, damut die linke Spalte um die rechte Spalte fließen kann.
2. Der Rexr in der linken Spalte muss länger sein, als der im rechten Kasten, wenn die linke Spalte umrahmt ist.

Zweitere ist bis hierhin nicht erfüllt und deswegen füge ich noch ein wenig

sinnloser Text

ein, um die untere Kante dieser Spalte nach unten zu drük-
ken.

die rechte Spalte

```
#spaltemitterechts {
border-left:1px solid;
border-bottom:1px solid;
float:right;
width:230px;
margin:0px 0px 10px 10px;
        }
```

Weitere Beispiele:

- Beispiel mit drei Spalten (1).
- Beispiel mit zwei Spalten (ALA).
- Beispiel mit vier Spalten.
- Beispiel mit drei Spalten (2).
- statisches Beispiel.
- Beispiel mit `float`.

Entspricht die Seite den **W3C-Normen**?

Anregungen zu Navigationselementen

Navigation bzw. Gestaltung und Design von Navigationselementen auf deiner Website sind auch ein wichtiger Aspekt der Barrierefreiheit und sollten insbesondere in der Planungsphase bzw. bei der Erstellung von Vorlagen beachtet werden, denn eine nachträgliche Anpassung der Navigationselemente greift in das grundsätzliche Design ein, und Anpassungen können sehr viel Zeit beanspruchen. Der wichtigste Aspekt ist, dass die Navigationselemente überhaupt bedient werden können. Es gibt z.B. Navigationselemente, die ausschließlich mit der Maus bedient werden können: ein Button, der mit einer JavaScript-Funktion über das **onMouseOver**-Attribut zum Anzeigen eines Menüs dient, kann nicht mit der Tastatur bedient werden, wodurch die Navigation für Blinde und für Menschen mit Bewegungseinschränkung unzugänglich wird. Navigationselemente sind also Geräte-unabhängig zu gestalten!

Für Leute, die Schwierigkeiten bei der Orientierung haben – ob durch kognitive Störungen, weil sie deine Seiten mit anderer Zugangssoftware als dem Standard-Browser lesen oder bedingt durch Einschränkungen in der Bewegung – sind einheitliche und aussagekräftige Navigationselemente viel wert. Zum einen sollten die Aussagekraft der Navigationselemente wie Menü- oder Fußleisten eindeutige Informationen enthalten, zum anderen sollten die Navigationselemente einheitlich oder zumindest schematisch im Design sein.

Mit Navigationselementen sind vor allem eine Menüleiste, eine Fußleiste, eine Seitenüberschrift und ein Inhaltsverzeichnis oder eine Site-Map gemeint. Beispiele für Überschriften und Menüleisten siehst du auf den Screen-Shots ab Seite 55. Fußleisten sind besonders dann sinnvoll, wenn du auch Seiten mit längeren Texten auf deiner Site hast und die eigentliche Navigationsleiste beim Herunterscrollen verschwindet. Auch bei umfangreichen Sites mit stark verzweigter Navigation ist die Fußleiste sinnvoll, um jederzeit zu den Übersichtsseiten wieder zurückzukehren. Wie du letzten Endes Navigationselemente einbindest, ist eine Frage deines Stils – und nicht unbedingt Gegenstand des barrierefreien Webdesigns.

Ein Inhaltsverzeichnis oder Site-Map ist dagegen keine Frage des Designs, sondern ein zusätzliches Informationsangebot, das umso wichtiger wird, je größer deine Website ist. Hier bietet sich auch die Möglichkeit an, eine Suchfunktionalität anzubieten, auf die ich in nächsten Abschnitt auf Seite 63 zurückkomme.

Zum Inhalt der Navigationselemente

Als Navigationselemente sind vor allem eine Menüleiste sowie eine redundante Fußleiste zu sehen. In der Menüleiste, die meist am linken oder rechten Rand bzw., wenn sie horizontal gestaltet wird, mit der Überschrift im oberen Bereich des Browser-Fensters per CSS, Frames oder Tabellen platziert wird, sollten Links zu allen wichtigen Bereichen deiner Site enthalten sein mit eventuellen weiteren Links zu Unter-Rubriken. Insbesondere wenn du mit CSS oder Tabellen die Trennung von der Menüleiste und dem eigentlichen Inhalt vornimmst, wird es bei Seiten mit längeren Texten notwendig, zumindest einige der Links aus der Menüleiste zu wiederholen. Damit erleichterst du insbesondere denen den Zugang zu deinen Seiten, die keine Maus zum schnellen Scrollen benutzen können. Auch Screen-Reader-Benutzern und vielen anderen hilfst du so. Setzt du Frames ein, ist die Fußleiste eine nützliche Ergänzung, da z.B. Screen-Reader-Benutzer nicht das Menü-Frame suchen müssen (bei mehr als zwei Frames).

Du kannst natürlich sehr viele Informationen in einer Menüleiste anbieten, die deinen Besuchern Orientierungshilfen geben und damit eine höhere Transparenz des Site-Aufbaus schaffen. Es ist nicht leicht, das optimale Maß an Informationen zu bestimmen. Aus Sicht der Barrierefreiheit sind die inhaltlichen Aspekte für Navigationselemente im wesentlichen die Aussagekraft der Link-Texte (einschließlich Alternativtexte, wenn Navigationselemente grafisch dargestellt werden) und die Berücksichtigung aller gefragten Bereiche deiner Homepage.

Link-Texte sollten aussagekräftig sein, um z.B. einen Sinn zu ergeben, wenn sie außerhalb des Kontexts gelesen werden. Screen-Reader haben zur schnelleren Navigation auf WWW-Seiten die Funktionalität, nur Link-Texte ohne restlichen Text, aber mit den Alternativtexten der als Links eingesetzten Grafiken vorzulesen. Was nützt es dann, wenn du lauter Links hast wie

```
<A href="Seite_Freddy.htm">Hier
klicken</A>, um mehr über Freddy zu
erfahren<BR />
```

oder

```
<A href="Seite_Helmut.htm">Hier
klicken</A>, um mehr über Helmut zu
erfahren
```

die vom Screen-Reader als "Hier klicken Hier klicken" gelesen werden. In diesem Fall wäre jeweils der gesamte Text als Link zu setzen:

```
<A href="Seite_Freddy.htm">Hier
klicken, um mehr über Freddy zu
erfahren</A><BR />

<A href="Seite_Helmut.htm">Hier
klicken, um mehr über Helmut zu
erfahren</A>
```

damit das Ziel des Links verstanden wird. Natürlich bleibt dir offen, das "Hier klicken" mit einem **title**-Attribut zu versehen, etwa:

```
<A href="Seite_Freddy.htm"
title="Infos über Freddy">Hier
klicken</A> ...
```

oder das "Subjekt" als Link-Text zu formatieren:

```
Informationen über <A
href="Seite_Freddy.htm" title="Infos
über Freddy">Freddy</A> ...
```

was dem Screen-Reader-Benutzer genauere Informationen gibt. Ganz abgesehen davon wird mit dem "Hier klicken" die Benutzung einer Maus impliziert und damit auch eine Barriere!

Was du nun alles in den Navigationselementen berücksichtigst, ist stark abhängig vom Umfang und Inhalt deiner Site. In jedem Fall sollten in einem Navigationselement Links zur Startseite, Kontaktseite und Übersichtsseiten der verschiedenen Informationsangebote auf deiner Site. Hier geht es vor allem darum, sich in die Lage des Besuchers zu versetzen, der

- mit oder ohne Maus
- mit oder ohne Vorkenntnisse der spezifischen Themen auf den Seiten
- mit oder ohne technisches Verständnis in der Bedienung der Zugangssoftware

nicht lange nach bestimmten Informationen suchen möchte. Welche Informationen suche ich, wenn ich deine Seite besuche? Dabei sind allgemeine Begriffe Fachbegriffen vorzuziehen. Eine Vielzahl der Besucher wird deine Seiten gezielt suchen und mit Sicherheit auch die fachlichen Terme verstehen. Aber es werden eine ganze Menge Besucher ohne Vorkenntnisse deine Seite aufrufen, einfach weil sie das Thema interessiert. Auch für Menschen, die kognitive Störungen haben, seien es Konzentrationsschwächen oder schwere Nervenerkrankungen, kannst du die Navigation auf deiner Site erheblich erleichtern, wenn du gängige und eindeutige Begriffe gut lesbar verwendest. Als die "Steuerung" durch deine Seiten sollten die Navigationselemente natürlich auch besonders gut von Sehbehinderten gelesen werden können (s. ab Seite 16).

Eine Site-Map oder ein Inhaltsverzeichnis ist ebenfalls eine sinnvolle Navigationshilfe, weil damit alle Inhalte deiner Site "auf einem Blick" gelesen werden können. Je dezentraler eine Site gepflegt wird und je mehr individuelle Gestaltungselemente für einzelne Angebote sie dadurch enthält, desto notwendiger ist es, Informationen in einem Überblick zusammenzufassen. Verfügst du allerdings über keine Werkzeuge, z.B. ein Content-Management-System, benasprucht die Pflege einer solchen Übersichtsseite viel Zeit.

Einheitliche Navigationsgestaltung

Für Menschen, die in ihrer Bewegungsfreiheit eingeschränkt sind, ist es besonders wichtig, dass Navigationselemente auf verschiedenen Seiten entweder an gleicher Stelle sind (Navigation mit der Maus) oder im Quelltext derart organisiert werden, dass sie schnell über die Tastatur erreichbar sind. D.h. die Navigationselemente sollten entweder am Anfang oder am Ende des Quelltextes eingebaut werden.

Wendest du eine konsistente Navigation auf deine Seiten an, wie in WAI-13.4 gefordert, und bietest du Informationen, die den Zusammenhang der jeweils aktuellen Seite innerhalb der Site-Struktur deutlich machen, hilft das auch Screen-Reader-Nutzern und vielen anderen, sich auf deiner Seite zu orientieren. Abgesehen von durchdachten Menü- oder Fußleisten kannst du dir überlegen, welche weiteren Informationen zur Orientierung dem (nicht-sehenden) Besucher helfen können, deine Seite schnell und unkompliziert zu navigieren. Gerade dann solltest du die Gelegenheit nutzen, das Layout zu dokumentieren. Liest jemand deine Seiten nicht am Bildschirm, können Navigationselemente sehr umständlich zu finden sein, z.B. einfach weil sie im Quelltext mitten in einer Tabelle enthalten sind. Wer mit

Sound, Braille oder kleinen Displays arbeitet, wird die Navigationselemente auf jeder einzelnen Seite suchen müssen. Daher ist die eine ergänzende Beschreibung sehr hilfreich (WAI-13.3).

Wie auch immer du deine Navigationselemente gestaltest, ist der Zugang mit Screen-Readern auch insofern problematisch, als die Links immer vorgelesen werden. Hast du z.B. die Navigationselemente zu Anfang des Quellcodes eingebaut, wird der Screen-Reader zuerst immer diese Links vorlesen, was auf Dauer zeitraubend werden kann. Daher wird in WAI-13.6 die Gruppierung verwandter Links gefordert sowie die Möglichkeit, diese Gruppen bei Bedarf zu überspringen. Du hast folgende Möglichkeiten:

- Füge vor der Link-Gruppe einen nur für Screen-Reader erfassbaren Link ein
- Verwende das **MAP**-Element in Verbindung mit dem **title**-Attribut

Das könnte so aussehen:

```
<MAP title="Navigationsleiste">
  <P>| <A
href="#sprungmarke0"><IMG
src="grafik/transparent.gif"
width="1" height="1"
alt="Navigationsleiste
überspringen"></A>
  <A
href="index.htm">Startseite</A> |
  <A href="apfel.htm">Über
Äpfel</A> |
  <A href="birne.htm">Über
Birnen</A> |
  <A href="sitemap.htm">Site-
Map</A> |</P>
</MAP>
<H1><A
name="sprungmarke0">Willkommen auf
der Obst-Site</A></H1>
...
```

Hier wird die Navigationsleiste – ein simpler Absatz – in das **MAP**-Element eingebettet, die vom Screen-Reader als Navigationselement erkannt wird und über das **title**-Attribut eine sinnvolle Bezeichnung erhält. Der erste Link gibt den Screen-Reader-Benutzer die Möglichkeit, sofort das Navigationselement zu überspringen. Der Durchschnitts-Surfer wird die Sprungmarke gar nicht wahrnehmen.

> Links, die du hintereinander z.B. in einer Fußleiste platzierst, sollten immer mit einem ausdruckbaren Zeichen getrennt werden wie **|**, **[**, **]**. Ältere Screen-Reader haben Probleme, Links einzeln auszugeben, wenn sie nur durch Leerraum getrennt werden.

Eine weitere Möglichkeit, Menüleisten zu verstecken, bieten CSS und die **display**-Eigenschaft. Mit **display:none** kannst du z.B. den Link zum Überspringen der Menüleiste im letzten Beispiel vor CSS-fähigen Browsern verstecken. Ein Beispiel für das Verstecken von Elementen, die insbesondere von Text-orientierten Browsern ausgegeben werden, findest du im Beispiel "rote Liste" ab Seite 18.

Erreichbarkeit der Navigationselemente

Browser ermöglichen den Zugang zu den Links auf einer Seite auch ohne Maus: du springst mit der TAB-Taste von Link zu Link, genau wie in vielen anderen Programmen auch. Bist du am Ziel deiner Wünsche, drückst du die ENTER- oder die RETURN-Taste, worauf sich der Link im Browserfenster öffnet.

Bei mehrmaligem Druck auf die TAB-Taste werden die Links normalerweise in der Reihenfolge angesprungen, in der sie im Quelltext erscheinen. Im HTML-Text kannst du die Reihenfolge ändern, wenn dies einen logischen Sinn macht, z.B. wenn du viele Links auf deiner Seite hast und wichtige Navigations-Links in der Mitte oder zu aller Letzt erscheinen. Ein Beispiel:

```
| <A href="datei0.htm"
tabindex="3">Link1</A> |
<A href="datei1.htm"
tabindex="1">Link2</A> |
<A href="datei2.htm"
tabindex="0">Link3</A> |
<A href="datei3.htm"
tabindex="2">Link4</A> |
```

Ruft jemand eine Seite mit obigen Quellcode auf und betätigt die TAB-Taste, erreicht er zuerst den "Link3" und dann in dieser Reihenfolge "Link2", "Link4" und "Link1". Bei erneutem Drücken der TAB-Taste wird die Symbol- und Adressleiste aktiviert, dann wieder "Link3". Enthält deine Homepage außer diesen Links auch weitere Links, Formulare, Image-Maps und Objekte,

werden diese ebenso mit der TAB-Taste angesprungen – und sollten also bei der Vergabe von Tabindizes berücksichtigt werden! Für das Attribut sind Zahlen zwischen 0 und 32767 erlaubt.

Wie für Screen-Reader-Anwender ist die Tastatur auch für Menschen mit motorisch Behinderungen wichtiger als eine Maus. Alle Funktionen müssen sich daher auch per Tastatur ausführen lassen. Z.B. sollten Image-Maps immer Client-seitig sein, um die Bedienung mit der Tastatur zu erlauben.

In Verbindung mit dem Zugang zu wichtigen Links auf deinen Webseiten mit der Tastatur ist natürlich auch das **accesskey**-Attribut zu nennen, das einem Link beigefügt werden kann und somit den direkten Aufruf über eine Tastaturkombination (Shortcut) ermöglicht. Leider kannst du hiermit nicht alle Browser auf gleiche Weise bedienen, denn die Browser und auch die Betriebssysteme deines Besuchers haben ja selbst eine ganze Reihe Shortcuts, die den von dir mit **accesskey** vergebenen Shortcuts vorgezogen werden. Beispielsweise könnte ALT+C auf einem Windows-PC wunderbar für deine Zwecke genutzt werden, aber auf einem Mac mit CMD+C nicht. In Netscape ist ALT+H für die Hilfe reserviert – und wenn du bedenkst, wie viele Browser und Betriebssysteme es gibt, wird es kaum Buchstaben geben, die in jedem Browser als **accesskey** funktionieren. Die wahrscheinlich einzigen Tastenkürzel, die überall funktionieren dürften, sind die Zahlen 0 ... 9.

Ein Beispiel für das Verwenden des **accesskey**-Attributs ist das Intranet eines Unternehmens, wo oft alle User die selbe Zugangssoftware benutzen. So sieht es dann aus: Mit dem C auf der Tastatur würde der angegebene Link aufgerufen werden:

```
<A accesskey="C"
href="http://www.xyz.mond"
title="Startseite der Mondfirma
XYZ">XYZ</A>
```

Arbeitest du mit Shortcuts, solltest du sie dokumentieren und deine Besucher darauf hinweisen, dass diese Möglichkeiten bestehen. Auch solltest du angeben, welche Shortcuts in welchem Browser auf welchen Betriebssystemen funktionieren.

Kognitiv bedingte Barrieren

In den WAI-Richtlinien sind im wesentlichen nur solche Kriterien verankert, die physische Behinderungen betreffen. Psychisch-bedingte Barrieren müssen jedoch ebenso berücksichtigt werden, da sie durchaus – wenn auch in subtilere Weise – jeden betreffen können.

Verständlich wie eine Grundschul-Lektüre

In den Anfangszeiten des WWW waren es begabte Programmierer und Gelehrte an Hochschulen, die das Medium "Internet" benutzten. Seit Anfang der 90er finden die Teilbereiche "WWW" und "Email" des Internets ein immer breiteres Publikum. Es sind nicht mehr alleine die Hochintelligenten, Analytiker und Computer-Freaks, die das Web benutzen, sondern mehr und mehr auch "durchschnittliche" Personen und solche, die eine unterdurchschnittliche kognitive Wahrnehmungsfähigkeit haben.

Es ist wichtig, dass Informationen auf einer Website von möglichst vielen verstanden werden. Versuche bei der inhaltlichen Gestaltung deiner Webseiten möglichst auf komplexe Sätze und zu vielen Fremdwörtern zu verzichten.

Einfache Struktur/simple Navigation

Auch der Aufbau der Website, insbesondere der Navigation, sollte möglichst einfach gehalten werden. Die Navigation mit Menüs ist eng verknüpft mit einer visuellen Vorstellung des Aufbaus samt der dahinter liegenden Inhalte. Bei fehlenden Visualisierungsfähigkeiten kommt es also darauf an, ein bestimmtes "Gerüst" vor sich zu haben, an dem Wiedererkennung und klare verständliche Sprache vermittelt werden. Gerade die Navigationselemente auf deiner Site sollten von jedem verstanden werden.

Eine andere kognitive Störung liegt im Bereich des Gedächtnisses. Ist ein Surfer nicht in der Lage, ungewöhnliche Namen oder andere Texte für mehr als ein Paar Sekunden zu behalten, kann das Navigieren im Web schnell in eine Sackgasse führen. Daher solltest du Namen für Links wählen, die einen Bezug zum Ziel haben. Die Bedeutung der Link-Texte für Screen-Reader-Benutzer ist so wichtig wie für Bilder (Seite 11).

Wird der Besucher deiner Webseiten durch mehrere Seiten "gelotst", um beispielsweise mehrere Formulare auszufüllen, gib Hilfestellung bei jeder Entscheidung statt einer ausführlichen Anleitung zu Beginn. Die begleitende Unterstützung kann textlich, grafisch und/oder akustisch vermittelt werden (WAI-14.2).

Reduziere die Notwendigkeit für Rechtschreibung

Auch Menschen mit einer Lese- und Schreibschwächen solltest du berücksichtigen. In WAU-14.1 wird die Verwendung der klarsten und einfachsten Sprache gefordert, die für den Site-Inhalt angemessen ist. Dabei solltest du natürlich daran denken, dass das WWW international ist und nicht jeder, der deutschsprachige Seiten liest, auch die Sprache besonders gut versteht.

Erfordert eine Seiten Eingaben in einem Formular oder Suchfunktion, solltest du auch bedenken, dass nicht jeder die deutsche Rechtschreibung beherrscht – das gilt sowohl für Deutsche als auch für internationale Besucher. Daher ist Vorsorge zu tragen bei allen Eingabefeldern, die eine präzise Eingabe erfordern.

Hast du z.B. eine Volltext-Suche auf deiner Site eingebaut, überlege dir, ob du die Möglichkeiten hast, phonetisch vorzugehen. Auch automatische Rechtschreibprüfung bei der Eingabe kann dem Benutzer helfen, schneller die Informationen zu finden, die er sucht. Beispielsweise könnte bei einem 0-Treffer-Ergebnis einer Suchanfrage die Rechtschreibprüfung und/oder phonetische Suche automatisch eingreifen und Verbesserungen vorschlagen.

Suchmaschinen bieten auch die Möglichkeit der Einbindung eines Thesaurus, der vor allem – aber nicht nur – mit Fachbegriffen arbeitet. WAI-13.7, das die Berücksichtigung verschiedener Fähigkeiten bei der Bedienung der Suchfunktionalität fordert, bedeutet dann, wenn du beispielsweise viele verschiedene Spielzeuge anbietest, und der Besucher in deiner Suchmaschine das Wort "Lokomotive" eingibt, dass auch Ergebnisseiten mit dem Wort "Modelleisenbahn" angezeigt werden, auch wenn das Wort "Lokomotive" dort nicht auftaucht.

Erläutere Abkürzungen und Akronyme

Die Erläuterung von Abkürzungen jeder Art ist eigentlich selbstverständlich. Erscheint eine Abkürzung zum ersten Mal, wird sie in der Regel ausgeschrieben. In WAI-4.2 ist dies explizit gefordert, damit Benutzer die Möglichkeit haben, in ihrem Browser wahlweise die Abkürzung oder

die ausgeschriebene Form darstellen zu lassen. Dafür gibt es zwei verschiedene Elemente: **ABBR** und **ACRONYM**, die beide das **title**-Attribut benötigen, um die ausgeschriebene Bedeutung angeben zu lassen. Der Unterschied zwischen dem **ABBR**-Element (abbreviation = Abkürzung) und dem **ACRONYM**-Element ist einfach, dass Akronyme eher eine Wiedererkennung im Sinne einer bekannten Begrifflichkeit oder Institution bedeuten wie

```
Das World Wide Web, abgekürzt
<ACRONYM title="World Wide
Web">WWW</ACRONYM>, ...
```

während das ABBR-Element abgekürzten Worten entspricht, etwa:

```
Hier noch ein <ABBR
title="Beispiel">Bsp.</ABBR>:
```

Auch in Datentabellen, wo oft Spalten- und Zeilenüberschriften abgekürzt werden, sollten diese beiden Elemente benutzt werden!

```
<TH><ACRONYM title="Laufende
Nummer">LN</ACRONYM></TH>

<TH><ABBR
title="Seite">S.</ABBR></TH>
```

Wenn du einfach bedenkst, dass ein Screen-Reader das "S" aussprechen müsste, um einen Sinn für den Benutzer zu geben, wird die Notwendigkeit des Ausschreibens schnell klar.

Was bei Formularen zu beachten ist

Eine Form der Kommunikation im Internet ist per Email. Möchtest du aber die Informationen in einer bestimmten Form von deinem Besucher erhalten möchtest, etwa weil sie unmittelbar mit einer Datenbank interagieren sollen oder weil du nur bestimmte Informationen erhalten willst, erfolgt die Kommunikation zwischen Benutzer und Server sinnvollerweise mit Formularen. Formulare kennzeichnen sich insbesondere dadurch aus, dass sie von anderen Programmen auf deinem Server weiterverarbeitet werden können. Im Formular selbst werden verschiedene Eingabefelder (Texteingabe, Check-Boxen, Drop-Down-Menüs, ...) eingebunden und per Button an einer im **FORM**-Element bestimmte Datei oder Funktion weitergeleitet. Die Datei bzw. Funktion ist i.d.R. eine Server-seitige Programmierung (CGI-Script, ASP, PHP3, ...), die die Ergebnisse aus dem Formular verarbeitet und – je nach Aufgabe – die Informationen mit einer Datenbank abgleicht, um dynamische Inhalte im Browser zu erzeugen, oder in einer Datenbank speichert, wenn Informationen vom Besucher "eingereicht" wurden, oder eine Email mit Server-seitigen Mail-Objekten erzeugt und versendet und so weiter. Client-seitige Verarbeitung ist beschränkt möglich, z.B. um eine Email mit dem Mail-Client des Besuchers zu versenden, und kann mit JavaScript programmiert werden. Auf die Möglichkeiten der Weiterverarbeitung von Formularen will ich nicht eingehen, da sie in keinster Weise die Barrierefreiheit einer Website beeinflussen. Lediglich die Client-seitigen JavaScripts können eine Barriere bedeuten, z.B. wenn JavaScript von der Zugangssoftware nicht unterstützt wird, weswegen sie durch Server-seitige Anweisungen ersetzt werden sollten.

Geräte-unabhängiger Zugang

Als "Spezialfall" der Tastatursteuerung von WWW-Seiten gilt die Bedienung von Formularen. Bei beschränkten motorischen Bewegungsmöglichkeiten sind Radio-Buttons und Check-Boxen, aber auch viele andere Formularfelder eine besonders kniffelige Angelegenheit, wenn sie nicht oder nur sehr umständlich mit der Tastatur bedient werden können.

Auch für Screen-Reader-Benutzer sind Formulare schwierig zu bedienen, wenn die Organisation innerhalb des **FORM**-Elements sich eher an der visuellen Anordnung am Bildschirm statt an logische Strukturierungen anlehnt. Die Formularfelder sollten also in einer vernünftigen Reihenfolge im Quelltext angeordnet werden. Wenn dies nicht möglich ist, kannst du entsprechen WAI-9.4 mit dem **tabindex**-Attribut, genauso wie auf Seite 61 für das **A**-Elemente gezeigt, eine Tabulatorenreihenfolge für die Formularfelder vergeben:

```
Betriebssystem: <INPUT type="text"
name="feld1" value="Win32"
tabindex="5"><BR />

Browser: <INPUT type="text"
name="feld2" value="Opera 5"
tabindex="8">
```

Für motorisch Behinderte sollten soweit möglich bereits alle Felder in Formularfeldern eingetragen sein, damit eine möglichst zügige Eingabe gewährleistet werden kann. Im obigen Beispiel sind die beiden Informationen z.B. auch per JavaScript zu ermitteln und könnten dynamisch als **value** festgelegt werden.

Ein Mehrfachtastendruck, wie z.B. für das "@" erforderlich, kann ebenfalls Probleme bereiten, weil Menschen mit körperlichen Behinderungen möglicherweise Schwierigkeiten haben, zwei Tasten gleichzeitig zu drücken. Zeichen, die nur mit Tastenkombinationen zu erzeugen sind, sollten nach Möglichkeit Server-seitig erstellt und nicht vom Benutzer eingegeben werden müssen. Das kann sich auch auf Groß-/Kleinschreibung beziehen, aber es sind vor allem die Zeichen gemeint, die z.B. auf der PC-Tastatur mit ALT-, STRG- und ALTGR-Tasten erreichbar sind. D.h. statt eine komplette Email-Adresse einzugeben, sollte die Eingabe in zwei getrennten Feldern erfolgen können:

```
<FORM method="post" name="formular"
action="/config/mail_konfig.php3">
  <TABLE><TR>
    <TD colspan="2">Email-Adresse
eingeben</TD>
  </TR><TR>
    <TD><STRONG>Email-
Adresse:</STRONG></TD>
    <TD><INPUT type="text"
name="email1" value="">@<INPUT
type="text" name="email2"
value=""></TD>
  </TR></TABLE>
...</FORM>
```

Im verarbeitenden Script können die beiden Teile der Email-Adresse mit dem "@" an der relevanten Stelle wieder zusammengeführt werden.

> Die Tabellenformatierung ist für das **FORM**-Element meistens notwendig, um ein optisch sauberes Formular zu erstellen. Die Verwendung von CSS erweist sich in manchen Browsern als schwierig.

Für Sehbehinderte kann diese Lösung das Gegenteil bewirken, weil der kleine Ausschnitt, den sie mit dem Screen-Magnifier sehen, eventuell die Fortsetzung der Eingabe nicht anzeigt. Das kann zu falschen Eingaben führen. Ein sinnvoller Mittelweg, um diese Barrieren gleichzeitig aufzuheben, ist, alle Formularelemente direkt untereinander aufzuführen, so dass jedes Feld linkerhand in einer vertikalen Linie steht. Damit könnte obiges Beispiel wie folgt aussehen:

```
<FORM method="post" name="formular"
action="/config/mail_konfig.php3">
  <TABLE><TR>
    <TD>Email-Adresse <STRONG>1.
Teil</STRONG></TD>
<INPUT type="text" name="email1"
value="">@
  </TR><TR>
    <TD>Email-Adresse <STRONG>2.
Teil</STRONG></TD>
    <TD><INPUT type="text"
name="email2" value=""></TD>
  </TR></TABLE>
...
</FORM>
```

Bezeichnungen für Formularfelder

Für Formularfelder wie Eingabefelder oder Auswahllisten gab es vor HTML 4 keine logische Beschriftungsmöglichkeit. So war (und ist) es für Blinde schwierig, Eingabefelder in Formularen zu identifizieren – gerade wenn sie mit Tabellen formatiert sind und der logische oder sequentielle Zusammenhang schwer nachzuvollziehen ist. Was optisch wie "Name:" mit folgendem, intuitiv auszufüllenden Eingabefeld am Bildschirm erscheint, kann im Quellcode weit auseinander liegen. Deswegen ist es für Screen-Reader-Benutzer wichtig, dass alle Eingabefelder eine direkt vorangestellte Bezeichnung erhalten, etwa:

```
Name: <INPUT type="text"
name="feld1">
```

Der bezeichnende Text sollte unmittelbar vor dem Eingabefeld positioniert werden, damit die Eingabefelder vom Screen-Reader angekündigt werden können. Wird die Bezeichnung hinter dem Feld angezeigt, muss der Benutzer den Screen-Reader anhalten und zum Eingabefeld zurückspringen lassen – ein unnötiger Aufwand!

Der Nachteil solcher HTML-Texte ist, dass es keine zwingende Verknüpfung zwischen Text und Eingabefeld gibt. In HTML 4 – und entsprechend auch in WAI 12.4 festgehalten – wurde deswegen das **LABEL**-Element mit dem **for**-Attribut eingeführt, das die direkte Bezeichnung eines Formularfeldes erlaubt:

```
<FORM>
  <TABLE><TR>
    <TD><LABEL
for="vorname">Vorname:</LABEL></TD>
    <TD><INPUT type="text"
id="vorname"></TD>
  </TR><TR>
    <TD><LABEL
for="nachname">Nachname:</LABEL></TD
>
    <TD><INPUT type=text
id="nachname"></TD>
  </TR>
</TABLE>
</FORM>
```

In diesem Beispiel werden die beiden Eingabefelder mit dem **id**-Attribut gekennzeichnet, wobei der Wert des Attributs laut W3C-Spezifikationen

- einen eindeutigen Namen haben muss
- in Anführungszeichen gesetzt sein muss
- nur (internationale) Buchstaben, Ziffern und Unterstrich enthalten darf, wobei das erste Zeichen ein Buchstabe sein sollte.

Existiert ein **LABEL**-Element mit einem **for**-Attribut mit genau demselben Namen, gibt der Screen-Reader diese Bezeichnung aus und hilft dem Nutzer, das Eingabefeld zu identifizieren.

> Setzt du das **accesskey**-Attribut in Formularen einsetzt, dann solltest du diese über die **LABEL**-Elemente vergeben. Diese werden über das **id**-Attribut mit der Bezeichnung auf die Eingabefelder herangezogen.

Das **LABEL**-Element hat keine visuelle Darbietung und dient ausschließlich dem nicht-visuellen Zugang zu Formularfeldern. Es kann aber auf alle Eingabefelder und Auswahllisten innerhalb eines **FORM**-Elements angewandt werden.

Ein Nachteil des **LABEL**-Elements – das geht aus der Einführung beim Release von HTML 4 hervor – ist die Interpretation nur durch moderne Browser. Da aber immer mehr Blinde Standard-Software einsetzen, wird der Anteil derjenigen, die einen Nutzen vom **LABEL**-Element haben, immer größer. In WAI-10.2 wird auch mit Priorität 2 Rücksicht darauf genommen, und für Browser, die die **for**- und **id**-Attribute nichts kennen, empfiehlt die WAI Feldbezeichnungen wie folgt anzubieten:

```
<LABEL for="adresse">
  Adresse: <TEXTAREA id="adresse"
tabindex="4" rows="5" cols="60">
  Bitte Postleitzahl nicht
vergessen! </TEXTAREA>
</LABEL>
```

Der Text vor dem Eingabefeld sichert, dass auch ältere Browser eine Bezeichnung für Eingabefelder liefern.

Grafische Buttons in Formulare

Normalerweise wird mit

```
<INPUT type="submit" ...
value="Absenden">
```

das Absende-Button für Formularfelder erzeugt, wobei das **value**-Attribut den textlichen Inhalts des Buttons bestimmt. Aber das geht nicht nur mit Buttons, sondern auch mit Grafiken, die zur individuellen Gestaltung von Formularen beitragen. Obwohl solche Grafiken nicht unbedingt eine Barriere bedeuten – es müssen immer Alternativtexte mittels **alt**-Attribute angeboten werden – können sie doch Probleme bereiten, denn: wenn du eine Grafik wie folgt einbindest:

```
<FORM action="/cgi-bin/auswertung"
method="post">
<INPUT type="image" name="submit"
src="grafik/jetzt-gehts-los.gif"
alt="Jetzt geht's los">
</FORM>
```

wird eine Art Server-seitige Image-Map erzeugt, das aber nicht Geräte-unabhängig ist und mit der Maus bedient werden müsste. Bei einem Mausklick werden die Koordinaten des Mausklicks in der Grafik mit dem eigentlichen Formularinhalt an den Server geschickt, was aber ohne Maus nicht möglich ist!

Eben wenn du diese Funktionalität zur Steuerung verschiedener Folge-Aktionen einsetzt, solltest du folgende Empfehlungen aus den Spezifikationen zu HTML4.01 (Abschnitt 17.4.1) befolgen:

- Setze mehrere Absende-Buttons ein, die alle einzeln aufgerufen werden und somit eine eindeutige Folge-Anweisung aufrufen.
- Setzte Client-seitige Image-Maps zusammen mit Scripts ein.

Die Einbindung grafischer Absende-Buttons kann auch mit dem **BUTTON**-Element erfolgen. Allerdings wird das in HTML3.2-Browsern z.T. überhaupt nicht angezeigt. Auch das **INPUT**-Element mit **type="button"** bereitet Probleme, indem es als Eingabefeld dargestellt wird.

Grafiken, die als Absende-Buttons für Formulare eingesetzt werden, stellen ebenfalls Navigationselemente dar und sind daher kontrastreich mit angemessen großer Zeichengröße darzustellen. Aussagekräftige Texte und/oder Symbole sollten sie ebenfalls enthalten.

Gruppierung von Formularfeldern

Entsprechend WAI-12.3 sollten größere Informationsblöcke in handhabbare Gruppen aufgeteilt werden. Gerade Screen-Reader-Nutzer, die lange Seiten linearisiert lesen, müssen ein erhöhtes Maß an Konzentration aufbringen, wenn komplexe Sachverhalte, die am Bildschirm in einer Art Workflow erscheinen, auf Webseiten angeboten werden. Hier kommt es selbstverständlich auf die Organisation im Quelltext an, aber es gibt auch HTML-Elemente, die dir helfen, logische Strukturierungen in Formularen einzubringen. Das folgende Beispiel zeigt, wie du verschiedene Eingabefelder innerhalb eines Formulars mit dem **FIELDSET**-Element gruppierst und die jeweilige Gruppe mit dem **LEGEND**-Element benennst. Damit ist es Screen-Reader-Anwendern möglich, schnell und ohne Überprüfung des Quelltextes in einem Formular zu navigieren.

```
<FORM action="/progs/neuanmeldung"
method="post">
  <FIELDSET>
    <LEGEND>Persönliche
Informationen</LEGEND>
    <LABEL for="vorname">Vorname:
</LABEL>
    <INPUT type="text" id="vorname"
name="vorname" tabindex="1">
    <LABEL for="nachname">Nachname:
</LABEL>
    <INPUT type="text" id="nachname"
name="nachname" tabindex="2">
    ... weitere persönliche
Informationen ...
  </FIELDSET>
  <FIELDSET>
    <LEGEND>Interessen</LEGEND>
    ... Formularfelder zu Interessen
...
  </FIELDSET>
</FORM>
```

Hast du lange Listen als Drop-Down-Menü in deinem Formular integriert, solltest du überlegen, ob diese Liste strukturiert in Gruppen angeboten werden kann. Ruft ein Screen-Reader-Benutzer eine solche Menüliste auf, muss er sich alle Menüeinträge vorlesen lassen, um den gewünschten Einztag zu finden. Innerhalb eines **FORM**-Elements wird ein solches Drop-Down-Menü mit dem **SELECT**-Element eingeleitet; die einzelnen Eintragungen der Auswahlliste werden mit dem **OPTION**-Element eingebunden:

```
<FORM ...>
  <SELECT name="auswahl1">
    <OPTION value="a11">1.
Listenelement
    <OPTION value="a12">2.
Listenelement
  </SELECT>
</FORM>
```

Die einzelnen **OPTION**-Elemente kannst du aber auch mit dem **OPTGROUP**-Element gruppieren und somit eine Auswahlliste mit zwei Ebenen erzeugen, wie du das aus den Menüleisten vieler Programme kennst. Die Hauptliste besteht aus den mit dem **label**-Attribut zugeordneten Namen der Gruppe. Wählt der Benutzer dann eine Eintragung aus der Hauptliste an, erscheinen die einzelnen Unterpunkte dieser Gruppe, die mit dem **OPTION**-Element erzeugt werden. Ein Beispiel:

```
<FORM action="/cgi-bin/laender"
method="post">
  <SELECT name="Land">
    <OPTGROUP label="Afrika">
      <OPTION value="af_107">Nigeria
       <OPTION value="af_135">Togo
    </OPTGROUP>
    <OPTGROUP label="Asien">
      <OPTION value="90">Afghanistan
      <OPTION value="155">Saudi-
Arabien
    </OPTGROUP>
  </SELECT>
</FORM>
```

Diese Funktionalität wird von den wenigsten Browsern unterstützt. Obwohl z.B. Netscape 6 das **OPTGROUP**-Element kennt, werden die Hauptgruppen nicht mit ausklappbaren Untergruppen angezeigt, sondern in ein und derselben Liste wie die Unterpunkte jeweils als Überschriften dargestellt, was die Liste verlängert statt verkürzt. MSIE5.5 und Opera 5 ignorieren das **OPTGROUP**-Element. Daher ist die Trennung langer Listen heute sinnvoller mit einer zweistufigen Auswahl zu lösen.

Keine leeren Formularfelder

Manche Screen-Reader können leere Eingabefelder in Formularen nicht ansteuern. Daher solltest du die Vorgabe eines Textes in **TEXTAREA**- und **INPUT**-Elementen berücksichtigen:

```
<FORM action=" /scripts/auslesen"
method="post">
    <TEXTAREA name="adresse"
rows="5" cols="60">
    Gib bitte deine Adresse an.
    </TEXTAREA>
    <INPUT type="submit"
value="Verschicken">
</FORM>
```

Logische Strukturierungs-Elemente

Es kann nicht oft genug gesagt werden: HTML dient der Strukturierung von Inhalten. Willst du Texte im Quelltext hervorheben, bietet HTML einige Elemente an, um dies auf einer strukturellen Ebene zu berücksichtigen. Allen voran sollten die Elemente zur Schaffung von Überschriftebenen (H1, H2, ... H6) verwendet werden. Das **FONT**-Element soll nach W3C nicht verwendet werden – ganz abgesehen davon, dass es nie standardisiert wurde, sollten die Effekte, die mit diesem Element erzeugt werden, alle mit CSS erzeugt werden. Alternative Zugangssoftware sind auf strukturelle Anweisungen angewiesen. Erzeugst du Hervorhebungen anders als mit strukturellen Elementen, werden die Anwender alternativer Zugänge benachteiligt. Auch Suchmaschinen bewerten Begriffe in Überschriften und anderen strukturellen Hervorhebungen höher als normalen Text.

Überschriften mit H1 ... H6

Was dem Durchschnitts-Besucher deiner Seite zur Orientierung hilft, sind Überschriften. Oft werden sie aber rein optisch erzeugt, indem ein Absatz z.B. fett und mit vergrößerter Schrift dargestellt wird. Um jedoch Surfern ohne Bildschirm dieselben Hervorhebungen anzubieten, sollten nach WAI-3.5 die **H1**- bis **H6**-Elemente benutzt werden, und zwar so, dass sie auf einander abgestimmt sind. D.h. die erste Überschrift auf einer Seite sollte mit einem **H1**-Element erzeugt werden; auf **H2** folgt **H3** usw., wobei du natürlich auch mehrere Überschriften der Ebene 1, 2, usw. auf deiner Seite haben kannst. Du solltest bei längeren Dokumenten darauf achten, dass die Strukturierung auch logisch ist: z.B. sollte **H4** nicht auf **H1** folgen.. Keinesfalls solltest du diese Elemente für andere Formatierungen als Zwischenüberschriften benutzen – dafür dient CSS!

```
<h1>Überschrift, z.B. Titel</h1>
... Text / Absätze...
<h2>Überschrift, z.B.
"Übersicht"</h2>
... Text / Absätze...
<h2>Nächste Überschrift, z.B. eine
"Einführung"</h2>
... Absätze, Listen, Tabellen ...
<h3>Überschrift - Unterabschnitt des
voranstehenden H2-Elements</h3>
... Absätze, Listen, Tabellen ...
<h3>Überschrift - Unterabschnitt des
voranstehenden H2-Elements</h3>
... Absätze, Listen, Tabellen ...
<h2>Nächste Überschrift, z.B.
"Weitere Informationen"</h2>
... Absätze, Listen, Tabellen ...
<h3>Überschrift - Unterabschnitt zu
"Weitere Informationen"</h3>
... usw ...
```

Schriftgröße und weitere Angaben zum Erscheinungsbild (Schriftart, Abstände, ...) kannst du über CSS steuern.

In WAI-13.8 wird auch gefordert, dass ein erkennbares Zeichen vor Überschriften eingefügt werden soll. Dies kannst du dann berücksichtigen, wenn du deine Überschriften als nicht oder kaum unterscheidbar zum normalen Text (etwa nur über eine andere Farbe) formatiert hast, damit Sehbehinderte die Hervorhebung besser erkennen:

```
<h3><IMG src="pfeil.gif">Eine
Überschrift, die wie normaler Text
aussehen könnte</h3>
```

Aufzählungen

In WAI-3.6 wird wie für Überschriften auch die Verwendung struktureller Elemente für Listen gefordert und der "Missbrauch" untersagt. Die einleitenden Elemente für verschiedene Elemente sind **UL** (unsortierte bzw. Bullet-Aufzählung), **OL** (ordered list = sortierte bzw. nummerierte Aufzählung) und **DL** (definition list). Diese Listen sollten nicht zu Layout-Zwecken benutzt werden, etwa um Text einzurücken; dafür solltest du CSS verwenden.

Nummerierte Aufzählungen (die auch mit Buchstaben und römischen Zahlen über das **type**-Attribut dargestellt werden können) können für den nicht-sehenden Benutzer schnell unübersichtlich werden, wenn sie sehr verschachtelt sind. In einer 2. oder 3. Verschachtelung einer Liste (etwa eine Inhaltsübersicht) sollte der Bezug jeweils im Listen Punkt angegeben werden, also 1., 2., 2.1, 2.2, 3., usw. statt 1., 2., 1., 2., 3., usw. Das W3C hat zur Abhilfe dieses Problems in CSS2 das **:before**-Pseudo-Element eingeführt, was allerdings heute von kaum einem Browser interpretiert wird. Für komplexe Listen kannst du unsichtbare Tipps für Screen-Reader-Nutzer anbieten, wie auf Seite 60 beschrieben.

Mochtest du die vorangestellten Zeichen in unsortierten Aufzählungen ändern, solltest du nicht auf das LI-Element zu verzichten, sondern statt dessen Grafiken mit dem **IMG**-Element einbinden. CSS bietet die Möglichkeit, benutzerdefinierte Grafiken für Bullet-Listen einzubinden:

```
<HEAD>
  <TITLE>Benutzerdefinierte
Aufzählungszeichen</TITLE>
  <STYLE type="text/css"><!--
    UL { list-style:url(stern.gif)
disc; }
  --></STYLE>
</HEAD>
<BODY>
<UL>
  <LI>Gute Zeiten</LI>
    <LI>Schlechte Zeiten</LI>
    <LI>Andere Zeiten</LI>
</UL>
```

Mit **list-style** kann der Pfad zu einer Grafik angegeben werden, die in diesem Beispiel für alle **UL**-Elemente bzw. darin enthaltene **LI**-Elemente verwendet wird. Kann die Grafik nicht geladen werden, ist eine zweite Darstellungsform angegeben, nämlich **disc** (der Standard-Bullet). Nur können Alternativtexte nicht vergeben werden.

Zitate und Quellenangaben

Auch die vielen logischen HTML-Elemente zur Kennzeichnung von Zitaten und anderen Beispielen und Quellenangaben auf deiner Site sollten ihrer Bestimmung nach eingesetzt werden und nicht etwa für das Layout- Manchmal wird z.B. das **BLOCKQUOTE**-Element zum Einrücken von Text verwendet –nur weil es in den Massen-Browsern so dargestellt wird. Aber dieses und weitere Elemente dienen wie Überschriften und Listen der logischen Strukturierung von Seiten.

BLOCKQUOTE erzeugt übrigens einen Absatz und sollte das **cite**-Attribut (Angabe der Quellen-URL) enthalten:

```
<BLOCKQUOTE
cite="http://www.beispiel.xy/intro">
  Dieser Text ist ein zitierter
Absatz"
</BLOCKQUOTE>
```

Entsprechend sind einzelne Sätze oder Teilsätze innerhalb deines Fließtextes mit dem **Q**-Element (quotation = Zitat) zu kennzeichnen. Weitere "reservierte" strukturelle Elemente für ähnliche Zwecke sind:

CITE	für ein Zitat oder eine Quellenangabe.
DFN	für Begriffe, die sogleich definiert werden.
CODE	für Teile eines Programmiercodes.
SAMP	für Beispielresultate aus Programmierungen.
KBD	für Texte, die der Benutzer z.B. zur Eingabe verwenden sollte.
VAR	für Variablen und andere programmtechnische Angaben
INS	für neu eingefügten Text, z.B. bei Aktualisierungen auf Webseiten.
DEL	für Texte, die gelöscht werden, die jedoch noch dargestellt werden.

Hervorhebung und Betonung

Zur Fett- und Kursivformatierung bietet HTML gleich mehrere Elemente an:

- **B** (bold = fett)
- **I** (italic = Kursiv)
- **STRONG** (Hervorhebung) wird meist fett dargestellt
- **EM** (emphasis = Betonung) wird meist kursiv dargestellt

Strukturelle Elemente sind nur die letzten beiden Elemente. Möchtest du also einzelne Wörter in deinem Fliesstext hervorheben oder betonen, solltest du die **STRONG**- und **EM**-Elemente verwenden. Willst du längere Texte fett oder kursiv darstellen, kannst du HTML-Elemente zur Fett- und Kursivformatierung einsetzen, wobei dir natürlich auch die flexiblere Nutzung von CSS freisteht.

Seltene Erscheinungen

Der Vollständigkeit halber möchte ich auf ASCII-Zeichnungen und **MathML** eingehen. Diese stammen eher aus der Gründerzeit von HTML, als Grafiken selten eingesetzt wurden. Heute sieht man sie wieder, allerdings als Smileys **;-)** und als SMS auf Handy-Displays. **MathML** ist eine eigene Gestaltungssprache zur Darstellung mathematischer Formeln u.ä., und wird vom Browser genauso wie HTML interpretiert.

ASCII-Zeichnungen

Bei ASCII-Zeichnungen empfiehlt die WAI, Grafiken mit Alternativtext zu verwenden. Und möchtest du unbedingt ASCII-Zeichnungen verwenden, solltest du eine Sprungmarke, wie z.B. auf Seite 60 gezeigt, verwenden.

```
<P><A href="#sprungmarke0"
style="display:none;">nachfolgende
ASCII-Zeichnung überspringen</A></P>
<PRE>
  %
   — — — — — — — — — —
— — — —
100 |              *
|
 90 |                 *  *
|
 80 |           *           *
|
 70 |              @           *
|
 60 |           @                 *
|
 50 |        *        @
*           |
 40 |                    @
*        |
 30 |    *  @              @  @
*     |
 20 |
|
 10 |    @                       @
@  @  @     |
      0  5 10 15 20 25 30 35 40 45
50 55 60 65 70
      Blinkfrequenz (Hz)
</PRE>
<A name="sprungmarke">Name für
Zeichnung</A>
```

Der Screen-Reader liest diese Zeichnung Zeile für Zeile durch, was natürlich kein Mensch verstehen kann. Du solltest also eine ergänzende Beschreibung als Link zu einer Textalternative beifügen!

MathML und LaTeX

Auf deiner Webseiten solltest du immer eine Gestaltungssprache verwenden, sofern es eine für deine Zwecke gibt. Das gilt natürlich auch für die Mathematik. Das W3C hat auch eine Spezifikation für eine mathematisch-orientierte Sprache: MathML. Bist du mathematisch-wissenschaftlich tätig, kennst du wahrscheinlich auch das LaTeX-Makropaket zur Formatierung von Text und insbesondere Formeln.

Für Formeln solltest du keinesfalls Grafiken benutzen. Für einfache Zusammenhänge sind einfache HTML-Darstellungen angebracht in der textlichen Form "x + y = z". Aber meist sind die darzustellende Zusammenhänge wesentlich komplexer. Die gängigste Formatierungssprache ist wohl TeX (zusammen mit dem LaTeX-Makropaket). Die kompilierten DVI-Dateien (device-independent = Geräte-unabhängig) sind sehr zugänglich. MathML ist mindestens genauso barrierefrei zu gestalten, ist allerdings nicht ganz so verbreitet.

Weitere Aspekte der Dynamik und Geräte-Unabhängigkeit

Was du zu Bewegung vor allem aus Sicht motorisch eingeschränkter Menschen berücksichtigen solltest, sind bewegende Inhalte, das Neuladen von Seiten sowie das automatische Weiterleiten.

Bewegende Elemente

Es macht sicher für den einen oder anderen Spaß, auf einer Website ein kleines, superschnelles Bildchen mit der Maus zu jagen. Diese und andere "Spielereien", die meist mit JavaScript programmiert werden, können für Körperbehinderte, Blinde und auch für Sehbehinderte ein Problem darstellen. Sofern du eine Funktion – in diesem Fall beim Einfangen des Bildchens – mit der Bewegung in Verbindung bringst, solltest du immer eine Möglichkeit anbieten, die Bewegung zu stoppen. Dies geht beispielsweise, wenn du CSS-Angaben in deinem Script einbindest, das dann vom Benutzer ausgeschaltet werden kann. Denke daran, dass alle Funktionen auf deiner Website mit der Tastatur bedient können sollten.

Automatischer Neuaufruf der Seite

Gibst du deinem Besucher den Hinweis, deine Seite neu zuladen, weil bestimmte Informationen nur dadurch angezeigt werden können, anstatt die Seite automatisch neu zuladen, berücksichtigst du spezielle Probleme, die Screen-Reader mit dynamischen Prozessen auf WWW-Seiten haben. Du solltest also auf die Refresh-Angabe

```
<META http-equiv="refresh"
content="60">
```

in der Kopfzeile deiner HTML-Dokumente nach Möglichkeit verzichten, bis der Benutzer die Möglichkeit hat, selbst die Funktion auszuschalten.

Automatische Weiterleitungen

Gleiches gilt für das automatische Weiterleiten auf andere Seiten. Ein Code wie:

```
<HEAD>
    <META http-equiv="refresh"
content="5; neue_seite.htm">
</HEAD>
<BODY>
    <P>Wenn dein Browser
automatische Weiterleitung
unterstützt, dann wirst du in 5
Sekunden auf unsere <A
href="neue_seite.htm">neuen
Startseite</A> weitergeleitet.
Ansonsten wähle bitte den Link!</P>
</BODY>
```

sollte nicht benutzt werden, weil er nicht zum W3C-Standard gehört und außerdem Besucher deiner Seite verwirren und die History (zuvor besuchte Seiten) deines Browsers durcheinander bringen kann. Anstatt dessen solltest du:

- Server mit einem entsprechenden **http**-Statuscode (301) konfigurieren. **http**-Kopfbereiche sind vorzuziehen, weil sie weniger Übertragungsvolumen haben, weil sie auch auf Nicht-HTML-Dokumenten benutzbar sind und weil sie von Zugangssoftware benutzt werden können, die z.B. nur das **HEAD**-Element abfragen, wie Link-Checker. Außerdem bieten Statuscodes des Typs 30x weitere Informationen zur Seite, etwa ob sie permanent oder temporär umgezogen sind.
- Server-seitige Weiterleitungen einrichten, etwa in ASP mit **<%Response.redirect("neue_seite.htm")%>** (sofern ein ASP-Engine vorliegt)
- Seite mit einer statischen Seite ersetzen mit einem normalen Link.

Neue Browser-Fenster, andere Pop-Ups

Werbe-Banner sind sicher allen bekannt. Viele Screen-Reader-Benutzer haben einen echten Vorteil gegenüber anderen Surfern – die Zusatz-Browserfenster werden vom Screen-Reader nicht erfasst. Aber wie ist das mit wichtigen anderen Informationen, die mit dem **target**-Attribut eines Links in einem neuen Fenster angezeigt werden? Leider bleiben viele Screen-Reader im ursprünglichen Browser-Fenster "hocken" und kümmern sich nicht um das neue Fenster. Wenn also ein Screen-Reader-Benutzer auf ein Link geht, dass mit **target="_blank"** versehen ist, kommt es ihm vor, als ob überhaupt kein Link angegeben wurde. Das neue Fenster bleibt verborgen. Daher solltest du Möglichkeiten benutzen, die die Informationen im gleichen Browser-Fenster anzeigen (WAI-10.1).

Symantik und Meta-Informationen

Meta-Informationen dienen nicht nur solchen Werkzeugen wie Suchmaschinen oder Content-Management-Systemen, sondern auch dem Besucher deiner Seite. Dabei sind nicht die Suchmaschinen-Meta-Elemente gemeint wie **keywords** oder **description**. Der besseren Identifikation deiner Seite dient beispielsweise das **TITLE**-Element in ihrem Kopfbereich:

```
<HEAD>
  ...
  <TITLE>Aussagefähiger
Titel</TITLE>
  ...
</HEAD>
```

Mit dieser Angabe gibst du deinen Besuchern die Möglichkeit, in aller Kürze festzustellen, ob die aufgerufene Seite diejenige ist, die er sucht. Der Titel sollte daher individuell auf jede Seite angepasst werden.

> Das **TITLE**-Element, das im Header vorkommen muss, ist nicht zu verwechseln mit dem **title**-Attribut, das auf praktisch alle Elemente angewandt werden kann. Besonders nützlich ist das **title**-Attribut bei Links, um zusätzliche Informationen über das Ziel des Links zu vermitteln.

Vor dem einleitenden HTML-Element solltest du den Standard angeben, an den sich deine Seiten halten. Typischerweise für HTML 4.01 sieht die erste Zeile deiner WWW-Seiten dann so aus:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.0//EN" REC-html40-
19980424/strict.dtd
```

So kennt Zugangssoftwareden Standard der Seite – in diesem Fall W3C-Spezifikationen. Ob deine Seiten die angegebene Norm auch tatsächlich einhalten, kannst du dann überprüfen bei **validator.w3.org/check**. Neben der strengen Befolgung der Richtlinien kannst du auch angeben, dass eine Seite ein Frameset enthält oder die HTML4.01-Richtlinien im Wesentlichen einhält. Unter der Adresse **validator.w3.org/sgml-lib/catalog** findest du die komplette Liste aller Standards, die du auf diese Weise überprüfen kannst.

Wie lang eine Seite sein soll, ist nicht einfach zu entscheiden. Dazu kommt es einfach zu sehr auf den Inhalt an. Die WAI schlägt vor, längere Texte in kürzere Texte aufzuteilen, wenn dies natürlich und angebracht ist. Z.B. könnte ein längerer Textbeitrag in Inhaltsübersicht und einzelne Abschnitte aufgeteilt werden. Das bedeutet natürlich auch mehr Arbeit, denn auf jeder Seite muss ein Verweis auf die Inhaltsseite und eventuell auch vorherige und nachfolgende Seite eingebaut werden. Aber dadurch erhältst du Seiten, die bequem vom Besucher "durchblättert" werden können, und das Scrollen kann auf ein Minimum reduziert werden.

In manchen Fällen empfiehlt sich genau das Gegenteil, wenn z.B. der Text so umfangreich ist, dass der Interessierter gut beraten ist, die Seite zu downloaden und offline zu lesen. Da sind viele kleine Dateien umständlich herunterzuladen. Hier empfiehlt sich, eine zusätzliche Datei anzubieten, die dann sogar mit einem gängigen Textverarbeitungsprogramm wie Word formatiert sein kann und idealerweise gezippt ist. Du solltest immer Format und Größe in Kilobyte angeben.

Gültige HTML-Elemente

Wir sind nun fast am Ende. Zum Schluss der inhaltlichen Ausführungen folgt eine Tabelle mit den derzeit gültigen HTML4.01-Elementen. Elemente, die nicht mehr in den W3C-Standards vorkommen, sind mit einem Sternchen (*) gekennzeichnet. Elemente, die zu keinem Zeitpunkt einer W3C-Spezifaktion angehörten, wie **BLINK** oder **EMBED**, sind in der Tabelle nicht aufgeführt. In der zweiten Spalte kannst du sehen, ob das Element in HTML3.2 oder HTML2.0 enthalten war, und in der dritten Spalten kannst die Funktion des Elements erfahren.

Element	auch in HTML	Funktion
A	2.0, 3.2	Struktur
ABBR		Struktur
ACRONYM		Struktur
ADDRESS	2.0, 3.2	Meta
APPLET*	3.2	Ersetzt
AREA	3.2	Struktur
B	2.0, 3.2	Layout
BASE	2.0, 3.2	Verarbeitung
BASEFONT*	3.2	Layout
BDO		Verarbeitung
BIG	3.2	Layout
BLOCKQUOTE	2.0, 3.2	Struktur
BODY	2.0, 3.2	Struktur
BR	2.0, 3.2	Layout
BUTTON		Struktur
CAPTION	3.2	Struktur
CENTER*	3.2	Layout
CITE	2.0, 3.2	Struktur
CODE	2.0, 3.2	Struktur
COL		Struktur
COLGROUP		Struktur
DD	2.0, 3.2	Struktur
DEL		Meta
DFN	3.2	Struktur
DIR*	2.0, 3.2	Struktur
DIV	3.2	Struktur
DL	2.0, 3.2	Struktur
DT	2.0, 3.2	Struktur
EM	2.0, 3.2	Struktur
FIELDSET		Struktur
FONT*	3.2	Layout
FORM	2.0, 3.2	Struktur
FRAME		Ersetzt
FRAMESET		Layout
H1	2.0, 3.2	Struktur
HEAD	2.0, 3.2	Struktur
HR	2.0, 3.2	Layout
HTML	2.0, 3.2	Struktur
I	2.0, 3.2	Layout
IFRAME		Ersetzt
IMG	2.0, 3.2	Ersetzt
INPUT	2.0, 3.2	Struktur
INS		Meta
ISINDEX*	2.0, 3.2	Struktur
KBD	2.0, 3.2	Struktur
LABEL		Struktur
LEGEND		Struktur
LI	2.0, 3.2	Struktur
LINK	2.0, 3.2	Meta
MAP	3.2	Struktur
MENU*	2.0, 3.2	Struktur
META	2.0, 3.2	Meta
NOFRAMES		Alternative
NOSCRIPT		Alternative
OBJECT		Ersetzt
OL	2.0, 3.2	Struktur
OPTGROUP		Struktur
OPTION	2.0, 3.2	Struktur
P	2.0, 3.2	Struktur

PARAM	3.2	Verarbeitung
PRE	2.0, 3.2	Layout
Q		Struktur
S*		Layout
SAMP	2.0, 3.2	Struktur
SCRIPT	3.2 (DTD)	Verarbeitung
SELECT	2.0, 3.2	Struktur
SMALL	3.2	Layout
SPAN		Struktur
STRIKE*	3.2	Layout
STRONG	2.0, 3.2	Struktur
STYLE	3.2 (DTD)	Verarbeitung
SUB	3.2	Layout
SUP	3.2	Layout

TABLE	3.2	Struktur
TBODY		Struktur
TD	3.2	Struktur
TEXTAREA	2.0, 3.2	Struktur
TFOOT		Struktur
TH	3.2	Struktur
THEAD		Struktur
TITLE	2.0, 3.2	Meta
TR	3.2	Struktur
TT	2.0, 3.2	Layout
U*	3.2	Layout
UL	2.0, 3.2	Struktur
VAR	2.0, 3.2	Struktur

Die HTML-Attribute in HTML4.01, die die Barrierefreiheit deiner Webseiten betreffen, findest du in nachfolgender Tabelle. Attribute, die in HTML4.01 nicht mehr spezifiziert sind, sind durch ein Sternchen (*) gekennzeichnet. In der zweiten Spalte findest du die Elemente, auf die sich die Attribute anwenden lassen.

Universalattribute, die sich auf fast alle Elemente beziehen, sind entsprechend gekennzeichnet, sollten aber mit den HTML-Spezifikationen abgeglichen werden, wenn du sie einsetzen willst. In der letzten Spalte findest du die Art der Funktion.

Attribut	**für Elemente**	**Funktion**
abbr	TD, TH	Alternative
accesskey	A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA	Zugänglichkeit
alt	APPLET, AREA, IMG, INPUT	Alternative
axis	TD, TH	Struktur
class	Universalattribut	Struktur
dir	Universalattribut	Verarbeitung
for	LABEL	Struktur
headers	TD, TH	Struktur
hreflang	A, LINK	Meta
id	Universalattribut	Struktur
label	OPTION	Alternative
lang	Universalattribut	Meta
longdes	IMG, FRAME,	Alternative

c	IFRAME	
scope	TD, TH	Struktur
style	Universalattribut	Verarbeitung
summary	TABLE	Alternative
tabindex	A, AREA, BUTTON, INPUT, OBJECT, SELECT, TEXTAREA	Zugänglichkeit
Title	Universalattribut	Meta
usemap	IMG, INPUT, OBJECT	Verarbeitung

Folgende Attribute haben keinen unmittelbaren Bezug auf die Zugänglichkeit einer Webseite: Anstatt Layout-Attributen solltest du CSS benutzen. Für Event-Handler solltest du den Geräte-unabhängigen Zugang gewährleisten, indem du Event-Handler für Maus und Tastatur verwendest. Die mit einem Sternchen (*) gekennzeichneten Attribute werden in HTML4.01 nicht mehr spezifiziert.

Weitere Struktur-Attribute: **start*, rowspan, colspan, span**

Weitere Layout-Attribute: **align*, valign*, clear*, nowrap*, char, charoff, hspace*, vspace*, cellpadding, cellspacing, compact*, face*, size*, background*, bgcolor*, color*, text*, link*, alink*, vlink*, border, noshade*, rules, size (nur für manche Elemente noch zulässig), marginheight, marginwidth, frame, frameborder, rows, cols**

Weitere Attribute zur Verarbeitung: **ismap, coords, shape**

Weitere Zugänglichkeits-Attribute: **target, scrolling, noresize**

Weitere Meta-Attributes: **type, cite, datetime**

Event-Handler-Attribute: **onblur, onchange, onclick, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onload, onload, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onreset, onselect, onsubmit, onunload**

Wie testet man die Seite auf Barrierefreiheit?

Eine absolut barrierefreie Website ist kaum erreichbar. Wie in der Einleitung der Broschüre angeführt, gilt es zunächst, die Startseite und andere Eintrittsseiten sowie häufig genutzten Seiten einer Website von Barrieren zu befreien. Eine bis in den letzten Winkel einer bestehenden Website gehende Überarbeitung kann sehr zeitaufwendig und somit kostspielig werden.

Da die Barrierefreiheit anderen Zielen bei der Gestaltung und Pflege von WWW-Seiten, wie etwa Aktualität oder Funktionalität, manchmal weichen muss, musst du die Barrierefreiheit pragmatisch angehen. Auf der Übersicht in der Mitte dieser Broschüre findest du eine tabellarische Übersicht sämtlicher WAI-Richtlinien zu barrierefreiem HTML. Rechts daneben kannst du die Priorität der jeweiligen Richtlinie lesen. Hältst du dich an diese Übersicht als Grundlage für die Überprüfung deiner Seiten und achtest du besonders auf die Richtlinien mit der Priorität 1, bist du auf dem besten Weg, deine Seiten barrierefrei zu gestalten.

Textalternativen

Bilder

Um eine erste Einschätzung der Barrierefreiheit zu erhalten, solltest du einen einfachen Test mit deinem Browser machen, indem du in den Einstellungen die Anzeige von Grafiken ausschaltest und deine Website ohne Bilder betrachtest. Im Microsoft Internet Explorer deaktivierst du die Bildanzeige über EXTRAS|INTERNETOPTIONEN|ERWEITERT|MULTIMEDIA. In Netscape-Browsern findest du diese Option unter BEARBEITEN|ERWEITERT; wo du GRAFIKEN AUTOMATISCH LADEN deaktivierst. Nun solltest du einen ersten Eindruck haben, wie die Alternativtexte auf deinen Seiten von Screen-Readern erfasst werden. Lassen sich deine Seiten bei dieser Einstellung ohne Informations- und Funktionsverlust lesen, können sie prinzipiell auch von blinden und sehbehinderten Menschen gelesen werden. Möchtest du deine Website ohne Layout sehen, um ihre Organisation genauer unter die Lupe zu nehmen, gehst du zu dieser Adresse: **www.delorie.com/web/lynxview.html**.

Hier wird der häufig von Blinden eingesetzte Text-orientierte Lynx-Browser simuliert. Du gibst die Adresse deiner Site ein, worauf die vom Lynx-Browser erfassbaren Informationen angezeigt werden.

Sound

Enthält deine Website Sound, solltest du ähnlich verfahren. Schaltest du deine Lautsprecher ausschaltest und lassen sich dann deine Seiten ohne Informationsverlust navigieren und verstehen, hast du den Anforderungen für taube Surfer Genüge getan!

Online-Werkzeuge

Webseiten, die sich an den aktuellen W3C-Standards orientieren, sind in der Regel sowohl auf- als auch abwärtskompatibel und können von allen Browsern sowie alternativer Zugangs-Software dargestellt werden. Zur Überprüfung auf W3C-Konformität gibt es mehrere Online-Werkzeuge, sog. HTML Validators, die deine Seiten auf korrektes HTML überprüfen und Informationen zum Status deiner Seite geben:

- **www.cast.org/bobby/** ist ein sehr guter HTML Validator, der nützliche Informationen zu Barrieren in deinem Quelltext gibt.
- **validator.w3.org/** ist der HTML Validator der W3C und legt besonderen Wert auf Dokumenten- und Metainformationen.
- **www.anybrowser.com/validateit.html** ist ein HTML3.2 Validator und hilft dir, die Abwärtskompatibilität (z.B. für CSS2) im Auge zu behalten.

Kontraste und Farben

In jedem Fall solltest du Grafiken auf Lesbarkeit durch Sehbehinderte kontrollieren. Die häufigste, aber keineswegs einzige Form von Sehschwäche ist die Rot-Grün-Sehschwäche. Kennst du niemanden, der Sehschwächen hat und dir bei der Einschätzung helfen kann, überprüfe selber die Grafiken auf Kontraste wie auf Seite 21 erwähnt. Ein wichtiger Test ist der "Farbinvertierungs"-Test. In den Optionen im Standard-Browser kannst du eigene Farben für die Darstellung von Webseiten einstellen. Die Seite muss auch dann ohne Informationsverlust lesbar sein, wenn du die Browser-Farben auf weißen Text, schwarzen Hintergrund und gelbe Linktexte einstellst und dabei die Farben der Webseite unterdrückst.

Geräte-Unabhängigkeit und Organisation

Versuche auch, deine Website ganz ohne Maus zu bedienen. Mit TAB bzw. UMSCHALT+TAB musst du von Link zu Link sowie durch Formularfelder und Image-Map-Bereichen springen können. So kannst du übrigens auch sehr schnell die Organisation deines Quellcodes einschätzen– ob z.B. Navigationselemente schnell bedient werden können usw.

Natürlich musst du deine Seite ohne Style-Sheets genau ansehen und prüfen, ob sie immer noch Sinn machen. Die o.a. Simulation des Lynx-Browsers ist dafür eine gute Möglichkeit, aber sie ignoriert gleich alle Layout-Angaben einschl. Tabellen auf deiner Seite. Das Ausschalten der Style-Angaben sollte auch im Standard-Browser getestet werden. In Microft-Browsern geht das über INTERNETOPTIONEN| EINGABEHILFEN, indem du die entsprechende Option zur Verwendung des eigenen Style-Sheets wählst (du musst dafür eine leere CSS-Datei erstellen, auf die du verweisen kannst). Geschickter ist, sofern du deine Style-Angaben aus einer externen Datei über das **LINK**-Element einbindest, wenn du diese Zeile zu Testzwecken im Quelltext auskommentierst. Hast du das DIV-Element eifrig benutzt, betrachte deine Seiten mit einem älteren Browser, etwa Netscape Navigator 3.0.

Skalierbare Größenangaben

Ob die Schriftgrößen deiner Seiten barrierefrei sind, kannst du sehr schnell feststellen, wenn du in den Browsern die jeweilige Option zur Schriftvergrößerung auswählst, die bei Microsoft und Netscape im Menü ANSICHT zu finden sind. Hast du die Schrift vergrößert, kannst du dich fragen, in welchem Verhältnis sie zu Texten in Grafiken steht. Je größer der Unterschied, desto schwieriger ist es für Sehbehinderte, Texte in Grafiken zu lesen.

Zwar ist es gängige Meinung, die unterste Bildschirmauflösung, die du für die Darstellung deiner Seiten testen solltest, sei 800x600 Pixel – dennoch solltest du dir auch die 640x480-Auflösung anschauen und sicherstellen, dass alles zumindest noch lesbar ist. Auch HTML-Elemente wie **IMG** und **TABLE** sollten sich skalieren lassen!

Installiere Software aus dem WWW

Im Idealfall solltest du Informationen und einige kostenlose Programme vom Internet laden:

Checkliste der WAI:

- HTML: **www.w3.org/TR/WCAG10/full-checklist.pdf**
- CSS: **www.w3.org/TR/CSS-access**

Software:

- JAWS (Screen-Reader) und MAGix (Screen-Magnifier): **www.freedomscientific.de/**
- Lynx (Text-orientierter Browser): **www.rene4u.com/intool.htm**
- IBM Homepage Reader (Audio-Browser): **www-3.ibm.com/able/**

Weiterführende Informationen (Links)

In einer Broschüre über Webdesign gehören auch Links ins WWW, die zur Vertiefung in die Materie sehr nützlich sein können. Viele der Angaben erschienen bereits im Text, und die Liste ist daher eine zusammenfassende Wiederholung.

W3C-Richtlinien:

Seit Mitte 2001 steht zwar die Version 2.0 der WAI-Richtlinien auf dem W3C-Server zur Einsicht bereit (**www.w3.org/TR/WCAG20/**), jedoch wird ausdrücklich darauf hingewiesen, dass es sich um eine vorläufige Version handelt, die weitere Veränderungen erfahren kann, und dass die Richtlinien nicht zitiert werden sollen. Die dieser Broschüre zugrunde liegenden Richtlinien sind die in der Version 1.0 von Mai 1999:

- **www.w3.org/TR/WAI-WEBCONTENT**
  Die offiziellen Kriterien des World Wide Web Konsortiems über die Gestaltung barrierefreier WWWW-Seiten ("Web Content Accessibility Guidelines 1.0" – WCAG1.0).
- **www.w3.org/Consortium/Offices/Germany/Trans/WAI/webinhalt.html**
  Eine nicht-verifizierte Übersetzung der WCAG1.0 von René Hartmann.
- **www.w3.org/TR/1999/REC-html401-19991224/**
  HTML4.01-Richtlinien des W3C.
- **www.w3.org/TR/REC-CSS1**
  CSS1-Richtlinien des W3C.
- **www.w3.org/TR/REC-CSS2**
  CSS2- Richtlinien des W3C.

Artikel und Informationen deutscher Sprache

- **www.barrierefreies-webdesign.de/knowware/**
  Hier findest du u.a. Beispiele aus dieser Broschüre.
- **www.teamone.de/selfaktuell/artikel/design/barrierefrei/index.htm**
  Jan Eric Hellbusch "Kontraste und andere Hürden" (Ausführlicher Artikel als Einführung in die Thematik).
- **www.bigub.de**
  Website des Vereins "Behinderte in Gesellschaft und Beruf" mit dem Schwerpunkt "Barrierefreies Internet".
- **www.einfach-fuer-alle.de**
  Projekt der "Aktion Mensch" zu barrierefreiem Webdesign.
- **www.lynet.de/~mhaenel/welcome.html**
  Website von Matthias Hänel mit Informationen zu der Nutzung von Computer durch blinde Menschen.
- **www.hoerfilm.de**
  Informationen über Audio-Deskription (für Filme).

Technische Informationen

- **trace.wisc.edu/world/java/java.htm**
  Trace R&D Center – Informationen über Barrierefreiheit und Usability von Java.
- **www.microsoft.com/enable/msaa/default.htm**
  Informationen zu Barrierefreiheit von Microsoft.
- **www-3.ibm.com/able/access.html**
  "Special Needs System" von IBM u.a. mit Informationen über Java.
- **www.macromedia.com/support/flash/ts/documents/tn4150.html**
  Flash barrierefrei mit OBJECT und EMBED einbinden.
- **ncam.wgbh.org/accessncam.html**
  Über die Gestaltung von QuickTime-Movies und deren Zugänglichkeit.
- **www.usdoj.gov/crt/508/508docs2.html**
  Informationen und Anforderingen an Websites der US-amerikanischen Justizbehörde.
- **developer.netscape.com/docs/manuals/index.html**
  Informationen von Netscape (wobei Netscape keine Informationen zu Barrierefreiheit anbietet).

Werkzeuge:

- **validator.w3.org/check**
  HTML-Validator nach den W3C-Spezifikationen.
- **www.cast.org/bobby/**
  HTML-Validator unter besonderer Berücksichtigung der WAI-Richtlinien.
- **anybrowser.com/**
  HTML-Validator zur Überprüfung der Kompatibilität von WWW-Seiten in verschiedenen Browsern.
- **www.w3.org/WAI/Resources/Tablin/**
  Werkzeug zur Überprüfung der Linearisierbarkeit von Tabellen.
- **www.delorie.com/web/lynxview.html**
  Simulation des textorientierten Browsers "Lynx"
- **www.w3.org/Protocols/HTTP/Performance/Pipeline**
  Überprüft die Geschwindigkeit beim Aufruf deiner Seiten.
- **ncam.wgbh.org/webaccess/magpie/**
  Erstellung von Untertiteln in verschiedenen Multimedia-Formaten.
- **vischeck.com**
  Überprüft den Kontrast und die Farben auf deiner Site.
- **www.justsmil.com/tools/**
  Verschiedene Werkzeuge zur Gestaltung zugänglicher Multimedia im SMIL-Format.

Software:

- **www.freedomscientific.de**
  Demoversionen des Screen Readers "JAWS" mit Sprachausgabe sowie des Screen-Magnifiers "MAGix".
- **www-3.ibm.com/able/**
  Demoversion des Audio-Browsers "Homepage Reader" von IBM.
- **www.rene4u.com/intool.htm**
  Download des Browsers "Lynx" (Text-orientierter Browser).
- **www.baum.de/webwizard.htm**
  Download des Web-Interface für den Screen Reader "Virgo".
- **www.webformator.de/**
  Download des Web-Interface für den Screen Reader "Blindows".

Abschließende Bemerkungen

Ich hoffe, das Lesen hat dir Spaß gemacht und du trägst die Idee des barrierefreien Webdesigns weiter. Das Thema ist eigentlich nicht besonders kompliziert – man muss es nur wissen! Dass Blinde ins Internet gehen, hat dich vielleicht vor dem Lesen dieser Broschüre überrascht. Ich hoffe, dass dir nun klar ist, dass wirklich fast jeder einen Computer bedienen kann – dass das manchmal nicht der Fall ist, liegt zumindest nicht unbedingt an den technischen Gegebenheiten.

Viele der Besucher – oder Kunden – deiner Website haben eine gesundheitliche Einschränkung. Nach amerikanischen Angaben hat jeder fünfte Surfer eine Einschränkung, und etwa 8% haben eine Behinderung. Weltweit ist die Quote eher höher einzustufen.

Im Prinzip kann eine temporäre oder dauerhafte Einschränkung durch Unfall oder Krankheit jeden treffen. Auch situative Bedingungen können den Zugang zum Web erschweren, etwa zu kleine Displays.

Die Überlegung, dass es mit Screen-Readern einen ganz anderen Zugang zur digitalen Welt gibt als mit dem Bildschirm, sehe ich als eine Horizonterweiterung. Letzten Endes kommt es immer wieder darauf an, alle Elemente möglichst Geräte-unabhängig anzubieten. So benötigen z.B. Grafiken eine zweite, ebenso wichtige Darstellungsform, nämlich einen beschreibenden Text.

Viele der angesprochenen Punkte stellen eine Herausforderung dar, die einen Verzicht auf die gängige Praxis bedeutet, wie etwa die Nutzung von Style-Sheets statt Tabellen als Layouthilfe oder Vorgaben von Schriftfarbe und –größe. Diese Umstellung ist aber für alle vorteilhaft:

- Gefällt deine Seite deinen Besuchern, werden sie sie wahrscheinlich weiterempfehlen.
- Barrierefreiheit erhöht den allgemeinen Zugang, auch für Suchmaschinen.
- Die Basis für zukünftige Technologien wie XML, Speech API wird eingebaut

Die Überarbeitung der wichtigen Seiten, d.h. der häufig benutzen Seiten und der Seiten, die von Benutzern regelmäßig zu erst aufgerufen werden bzw. der Navigation dienen, muss bei der Behebung von Barrieren an erster Stelle stehen. Die Regeln dafür sind, wie beschrieben, sehr einfach. Eine einmal wahrgenommene Barriere ist, so hoffe ich, für die Zukunft etwas, worauf du dann verzichten kannst.

Ich hoffe sehr, dass der deutsche Bundestag es schafft, bald ein Antidiskriminierungsgesetz zu verabschieden, in dem auch die Barrierefreiheit in der Informationstechnologie berücksichtigt wird, so wie das heute schon durch solche Gesetze in den USA und anderen EU-Staaten der Fall ist. Damit dürfte nicht nur der Zugang zu vielen Sites für Menschen mit Behinderungen erleichtert werden, sondern vermutlich würde auch die Wahrnehmung in der Öffentlichkeit stärker werden, dass Menschen mit Behinderungen gleiche Fähigkeiten und Rechte haben wie andere.

151/158: Windows 98 für Einsteiger Viele Übungen und Illustrationen - Die Elemente oder Objekte auf dem Desktop - Fenster - Titel- und Menüleiste - Der Wechsel zwischen Programmen - Rechtsklicken - Menüpunkte …

Extra 3: Word 97 für Anfänger Übungen: Dokument erstellen, bearbeiten und formatieren - Textausrichtung und Drucken - Rechtschreibung und Silbentrennung - Absatzformatierung - Text verschieben und kopieren - Aufzählungen - Rahmen und Schattierung - Suchen und Ersetzen - Ordnung in Deinen Dokumenten - Tabellen - Spalten und Initiale - Dokumentvorlagen .. Shortcuts

164: Word 2000 für Einsteiger Sinnvolle Grundeinstellungen - Regeln zur Texteingabe - Dokumente speichern - Bewegen in Word - Seitenansicht & Drucken - Schriftart, Schriftgröße und Zeichenformate - Tabellen ...

156: Excel 97 für Einsteiger Eingabe von Formeln - Zahlenformatierung - Summieren von Daten - Haushaltsbudget - Navigation - Formatierung - Diagramme - Funktionen …

155: Excel 97 für Fortg. Sortieren - Filter - Pivot-Tabellenbericht - Mustervorlagen - Konsolidierungen Gruppierung und Gliederung - Matrixformeln …

Extra 2: Excel 2000 für Einsteiger - Inhalt wie 156

146: Start mit Access 7/97 Analyse der Daten - Tabellen - Kundenkartei - Rechnungen - Erstellen der Datenbank und ihrer ersten Tabelle - Tabellen - Beziehungen - Sortieren ...

162, Access 2000 für Einst. Das Datenbankfenster– deine Kommandozentrale, Eingabe in ein Formular, Berichte – und ihr Entwurf, Tabellen und Tabellenentwurf ...

Special 1: PowerPoint 2000 f. Einst. Die verschiedenen Ansichten - Text bearbeiten - Arbeit mit ClipArts, Grafiken und Fotos ...

163: Internet für Einsteiger Internet erobern Schritt für Schritt! - Die wichtigsten Dienste im Netz - Welches Programm wofür? Browser genügt! - Vorüberlegungen: Modem oder ISDN? - Kabel, ADSL, Satellit? - Modem/ISDN installieren - Wahl des richtigen Dienstleisters - Suchmaschinen - Download - Internet-Explorer im Überblick - Zip-Format - E-Mail - Chat - Usenet - Cookies ...

122: Homepages selbst erstellen (48 Seiten) Der Aufbau von HTML-Dokumenten und HTML-Befehlen - Grundstruktur von HTML-Dokumenten - Grundlegende HTML-Befehle - Verweise in HTML Dokumenten - Mails versenden lassen - Anzeigen von Grafiken - Die graphische Gestaltung von Webseiten - Farben und Hintergrundbilder - Grafiken einfügen - Tabellen - Formulare

161: HomePages für Einsteiger (3. Ausgabe Dez. 2000, 72 Seiten) HTML, die Sprache des World Wide Webs, Wie gelange ich ins Netz, So entsteht die erste Datei, Wir arbeiten mit dem Editor!, Ein HTML-Dokument erzeugen und gestalten, Die wichtigsten HTML-Befehle in der Übersicht, So fügst Du Grafiken in Deine HTML-Dokumente ein, interne und externe Anker und Verweise, Farben für Links und Schrift … und vieles mehr

Plus 12: HomePages für Fortg. (80 Seiten) HTML und XHTML - Grundaufbau eines XHTML-Dokuments - HTML-TIDY, HTML-Kit - Java, JavaScript, Objekte, Eigenschaften und Methoden - JavaScript-Code in (X)HTML einbinden - Funktionen und Event-Handler - JavaScript inline und extern: Navigationsbuttons und Links - XHTML-Referenz: Die wichtigsten Befehle - Style Sheets: Die wichtigsten Attribute von CSS - und …

PLUS 6: JavaScript für Einsteiger (2. Ausgabe, 72 Seiten) Grundlagen - Was sind HTML-Seiten? - HTML-Kurzreferenz - HTML und JavaScript - Erste JavaScript-Programmierung - Event-Handler - Funktionen - Variablen - Schleifen mit for- Schleifen mit while - document.frames- document.forms - Text-Eingabefelder- Radio- und Checkbuttons- Auswahllisten- Das String-Objekt - Arrays - Frames- Fragespiel

PLUS 8: E-Mail mit Outlook Express 5 - Die Wahl des richtigen Dienstleisters - Sinnvolle Grundeinstellungen - E-Mails verfassen, abschicken und abholen - angehängte Dateien - Usenet ...

Special 3/KW 165: Outlook 98/2000/2002 f. Einst. - viele Übungen: Notizen, Termine, Kalender, E-Mail ...

Die beste Information über den Inhalt der Hefte bekommst du durch das kostenlose Download der ersten 15-20 Seiten von allen Heften:

www.knowware.de

Populäre Computer-Hefte für Einsteiger und Fortgeschrittene

Bestellung

Dein KnowWare-Händler kann alle Hefte bei seinem Lieferanten ohne Probleme und Risiko bestellen.

Du kannst auch bei Bonner Presse Vertrieb bestellen, entweder via www.knowware.de oder per Telefon, Fax oder Brief:

Bonner Pressevertrieb, Moeserstr. 2-3,
49074 Osnabrück, knowware@bpv-online.de
Tel: +49 (0)541 33145-20
Fax:+49 (0)541 33145-33

Versand und Verpackung in DM - Deutschland
DM 2,90 bis 3 Hefte
DM 3,90 bis 7 Hefte
DM 7,50 bis 10 Hefte
mehr als 10 Hefte: versandkostenfrei

Auslandsporto: siehe die Homepage

Name
Anschrift
E-Mail
Tel.

Geplante Hefte:
Flash, Excel 2000 Fortg., Musik/MP3
Works 1

*** Bestseller**

St	Nr	KnowWare PLUS	
	1	Windows-Tuning mit der Registry	7,-
	2	Windows Tips und Tricks	7,-
	3	PC Tuning mit Erfolg optimieren	7,-
	5	Word Tips & Tricks	7,-
	6*	JavaScript für Einsteiger, 2. Ausg.	7,-
	7	Windows schneller machen	7,-
	8*	E-Mail mit Outlook Express 5	7,-
	9	Staroffice 5.x für Einsteiger	7,-
	10	Paint Shop Pro 5/6 für Einsteiger	7,-
	11	Linux im Netzwerk	7,-
	12*	HomePages für Fortg.	8,-
	13	ISDN für Einsteiger	7,-
	14	Dreamweaver 3/4 für Einsteiger	8,-
	15	CGI & Perl für Einsteiger	8,-
	16	Bildbearbeitung für Einsteiger	8,-
	17	Windows 2000 für Fortg.	8,-
	18	Access: Formulare und Berichte	8,-
	19	Java für Einsteiger	8,-

St	Nr	KnowWare	DM
	100	Nutze Deinen PC optimal	5,-
	104	Was ist denn DOS?	7,-
	105	Start mit Windows 3.11	5,-
	107	Start mit Access 2	6,-
	112	Weiter mit Excel (Ver. 5/7)	6,-
	122*	WWW - Homepages selbst erstellen	7,-
	125	Batchprogrammierung DOS	7,-
	126	Excel VBA Makro-Programmierung	7,-
	129	Word 7 für Anfänger	7.-
	130	Start mit CorelDraw 5	6,-
	131*	Start mit Datenbanken und SQL	7,-
	132	Word 7 für Fortgeschrittene	7,-
	133	Intranet, HTML und Java, 2. Ausgabe	7,-
	135*	Excel 7 für Anfänger	7,-
	136	Windows NT 4.0 für Einsteiger	7,-
	138	Word für Studenten Ver. 7/97/2000	7,-
	139	Start mit Windows 95	7,-
	140	Start mit PowerPoint 7	7,-
	141	PC Aufrüsten, vol. 1	7,-
	143	Rund um den PC (für Anfänger)	7,-
	145	Start mit Excel (7, auch 5 und 97)	7,-
	146*	Start mit Access 7/97	7,-
	148	Windows 95 für Einsteiger	7,-
	151*	Windows 98 für Einsteiger (=158)	7,-
	152	Internet Explorer 4 für Einsteiger	7,-
	153	Linux für Einsteiger	7,-
	154	Access 97/2000 für Fortg.	7,-
	155*	Excel 97 für Fortgeschrittene	7,-
	156*	Excel 97 für Einsteiger (=150)	7,-
	157*	Start ins Internet, 4. Ausgabe	7,-
	159*	Frontpage 2000 für Einsteiger	7,-
	160	Weiter mit Word 97/2000	7,-
	161*	HomePages für Einsteiger	8,-
	162*	Access 2000 für Einsteiger	8,-
	163*	Internet für Einsteiger	8,-
	164*	Word 2000 für Einsteiger	8,-
	165	Outlook 98/2000/2002 für Einsteiger	8,-
	166	Windows ME/98 für Einsteiger	8,-
St	**Nr**	**KnowWare EXTRA**	**DM**
	2*	Excel 2000 für Einsteiger	7,-
	3*	Word 97 für Anfänger (=147)	7,-
	4*	Windows-Netzwerke für Einsteiger	8,-
	5	Windows 2000 für Einsteiger	7,-
	6	C++ für Einsteiger	8,-
	7	PHP und MySQL Einsteiger	8,-
	8	Barrierefreies Webdesign	8,-
St	**Nr**	**KnowWare SPECIAL**	
	1*	PowerPoint 2000 für Einsteiger	7,-
	2*	CD-Brennen für Einsteiger	7,-
	3*	Outlook 98/2000 für Einsteiger	7,-
	4*	Visual Basic für Einsteiger	7,-
St	**Nr**	**KnowWare Sonderheft**	
	1	Mac für Einsteiger - OS 8.5-9	9,80

Bestseller im KnowWare Verlag