Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

Technische Universität Wien

DIPLOMARBEIT

Barrierefreies Webdesign

Prinzipien, deren Anwendung und Konzeption eines Lehrganges

Ausgeführt am

Interuniversitären Institut für Informationssysteme
zur Unterstützung sehgeschädigter Studierender

an der Technischen Universität Wien

unter der Anleitung von o.Univ.Prof. Dipl.-Ing. Dr. A Min Tjoa
und Ass.Prof. Dipl.-Ing. Dr.techn. Wolfgang Zagler
als verantwortlich mitwirkender Assistenzprofessor

durch

Christoph Rettinger

1040 Wien, Mostgasse 8/9

September 2003

Aus Gründen der Lesbarkeit wurde im Text die männliche Form gewählt, nichtsdestoweniger beziehen sich die Angaben auf Angehörige beider Geschlechter.

Inhaltsverzeichnis

Vorwort **7**

I Theoretischer Teil

1 Barrierefreies Webdesign **11**
1.1 Guidelines 13

2 Curriculum **15**
2.1 Zielgruppe 15
2.2 Ziel des Kurses 15
2.3 Didaktische Überlegungen 16
2.4 Themen 17
2.4.1 Einführung in das barrierefreie Webdesign 17
2.4.2 Gestaltung äquivalenter Textinformation 17
2.4.3 Markup & Stylesheets 18
2.4.4 Layout & Navigation 18
2.4.5 Links & Image-Maps 19
2.4.6 Verwendung von Tabellen 19
2.4.7 Formulare 19
2.4.8 Frames 20
2.4.9 Sprache 20
2.4.10 Browser Kompatibilität 21
2.4.11 Hilfsmittel 21

II Skriptum

1 Einführung **7**
1.1 Barrierefreies Webdesign / Accessible Web-Design 7
1.2 Ein erstes Beispiel 8

2 Äquivalente Text-Information **13**
2.1 Das alt-Attribut 15
2.2 Das longdesc-Attribut 16

2.3 D-Links 16
2.4 Beschreibung von Bildern mit äquivalenter Text-Information 18

3 Markup & Stylesheets I **25**
3.1 HTML 4 26
3.2 Structural vs. Presentational Markup 28
3.3 Korrektes Verwenden von Markup 29

4 Markup & Stylesheets II **37**
4.1 Relative und absolute Größen 37
4.2 Bildschirmauflösungen 38
4.3 Das Testen von Webseiten 41

5 Layout & Navigation **47**
5.1 Grundlagen der Navigation 47
5.2 Verbesserte Tabellenstruktur 48
5.3 Skip Links 48
5.4 Table of Contents / Site Map 51
5.5 Linklisten 53
5.6 Meta-Information 54
5.7 Farben 57
5.8 Schriftarten 60

6 Links & Image-Maps **61**
6.1 Links 61
6.2 Image-Maps 64

7 Tabellen **69**
7.1 caption-Tags und summary-Attribute 70
7.2 Das headers-Attribut 72
7.3 Relative Größenangaben 73

8 Formulare **75**
8.1 Gruppieren von Informationen 75
8.2 Das Anordnen von Formularfeldern 78
8.3 Labels 81
8.4 Titles 81
8.5 Navigation mit der Tastatur 83

9 Frames **87**
9.1 Das noframes-Tag 88
9.2 Das name- und das title-Attribut 88
9.3 Alternative Seiten 91

10 Sprache **93**

10.1 Spezifikation der Sprache 93
10.2 Abkürzungen und Akronyme 94
10.3 Political Correctness 96

11 Browserkompatibilität **97**
11.1 Javascript 97
11.2 Alternative Seiten 106

12 Hilfsmittel **107**
12.1 Validatoren 107
12.2 Korrigierende Tools 112
12.3 Browser 116

III Anhang

A Der Onlinekurs auf CD-ROM **1**

Vorwort

Im Zuge meines Studiums und bei meiner Arbeit als Programmierer von dynamischen Webseiten habe ich viel Erfahrung im Umgang mit dem Internet gesammelt. Dabei wurde ich auf viele gute und wesentlich mehr „schlechte“ Seiten aufmerksam.

Durch meine frühere Tätigkeit bei der Forschungsgruppe für Rehabilitationstechnik an der Technischen Universität Wien hat sich meine Einstellung im Bezug auf die Gestaltung von Webseiten verändert. Ich wurde auf die Barrieren aufmerksam, denen blinde oder gehörlose Menschen im täglichen Leben begegnen. Dass solche Personen auch bei der Benutzung des Internets mit Schwierigkeiten zurechtkommen müssen, ist leicht nachzuvollziehen. Doch dass sich selbst scheinbar geringfügige Einschränkungen wie zum Beispiel Farbenblindheit oder die mangelnde Fähigkeit, eine Maus zu verwenden, negativ auf die Nutzbarkeit vieler Internetseiten auswirken, ist weniger offensichtlich.

Das Hauptanliegen meiner Diplomarbeit ist, ein Bewusstsein für diese Problematik zu schaffen. Mit dem Kurs, den ich im Rahmen dieser Diplomarbeit entwickelt habe, möchte ich einerseits die Probleme aufzeigen, auf die behinderte Personen oder Personen, die alternative Browser verwenden, im Internet treffen, andererseits möchte ich Lösungsansätze vorstellen.

Im Wesentlichen geht es mir darum, den Gestaltern von Webseiten ein Bewusstsein für die Möglichkeiten zu vermitteln, die ihnen zur Verfügung stehen, um einem möglichst großen Publikum den Zugang zu den Sites zu erleichtern. Viele Designer sind sich nicht bewusst, dass sie durch die Verwendung von verschiedenen Technologien, wie zum Beispiel Javascript, einen nicht unbeträchtlichen Teil der User bei der Benutzung ihrer Seiten behindern.

Die Techniken und Werkzeuge, die behinderten Menschen zur Nutzung des Computers und somit auch des Internets zur Verfügung stehen, können nur dann eingesetzt werden, wenn bestimmte Richtlinien in der Gestaltung von Webseiten berücksichtigt werden. In internationaler Kooperation wurden solche Richtlinen entwickelt. Die dafür verantwortliche Dachorganisation ist das W3C, das für Normungen im Internet zuständige Konsortium, die spezielle Initiative ist die Web Accessibility Initiative (WAI) [1].

Um die Umsetzung dieser Richtlinien in Österreich zu unterstützen, habe ich ich Lehr- und Lernunterlagen für das Accessible Webdesign entwickelt, die Webdesignern Hilfe-

stellungen geben sollen, diesen Anforderungen gerecht zu werden. Zu Beginn möchte ich einen Einblick in die Theorie des Accessible Webdesign geben und im Weitern ein Curriculum strukturieren. Darauf aufbauend entwickle ich einen Kurs bestehend aus einem Skriptum, einem Onlinekurs und einem Aufgabenblock.

Der Kurs stellt Schritt für Schritt die wichtigsten Konzepte und Techniken vor, um Webseiten im Sinne der WAI zu gestalten. An Hand von Beispielen werden die einzelnen Konzepte und Techniken verdeutlicht und veranschaulicht. Der Aufgabenblock bietet die Möglichkeit, die Lerninhalte zu vertiefen und in die Praxis umzusetzen.

Natürlich kann dieser Kurs nicht alle Aspekte des „barrierefreien Webdesigns“ umfassend beleuchten. Er kann jedoch einen Einblick bieten und erste Erfahrungen vermitteln, sodass Webdesigner diese in ihre Arbeit einfließen lassen können und für das Thema sensibilisiert werden.

Teil I.

Theoretischer Teil

1. Barrierefreies Webdesign

Das Internet ist wahrscheinlich die wichtigste Informationsquelle des 21. Jahrhunderts und wird es auch weiterhin bleiben. Gerade deshalb ist es wichtig, den Zugang zum Internet allen Menschen in gleichberechtigter Weise zu ermöglichen.

Aus vielen Bereichen unserer modernen Gesellschaft ist das Internet nicht mehr wegzudenken. Viele Leistungen werden inzwischen ausschließlich über das Internet angeboten. Dazu gehören z.B. Onlineshops, elektronische Dienste von Bibliotheken und Ämtern, Nachrichten oder Angebote im Unterhaltungssektor.

Information in maschinenlesbarer Form bieten behinderten Menschen eine große Chance. Informationen, die auf „herkömmliche“ Weise angeboten werden (etwa über Printmedien), sind für behinderte Menschen oft nur sehr schwer zugänglich, da es kompliziert und teuer sein kann, diese Informationen in einer angepassten Form bereit zu stellen. Eine Reihe von Beeinträchtigungen (Blindheit, Gehörlosigkeit, Mobilitätseinschränkungen) können hingegen durch die neuen Medien im Bereich der Informationsbeschaffung und Kommunikationstechnologien kompensiert werden.

Die neuen Medien stellen für behinderte Menschen eine Herausforderung dar [2]. Die Schwierigkeiten, mit denen diese Personen bei der Verwendung des Internets zu kämpfen haben, sind vielfältig.

Im deutschen Sprachraum wurde in letzter Zeit der Begriff des „barrierefreien“ Webdesigns geprägt. Dieser Begriff bezeichnet das Vorhaben, Webseiten so zu gestalten, dass sie von jedermann gelesen und bedient werden können.

Der Kurs, der im Rahmen dieser Diplomarbeit entwickelt wurde, soll helfen, Internetseiten „barrierefrei“ zu gestalten.

Diese „Barrierefreiheit“ ist außerdem ein Anliegen des „Erfinders“ des World Wide Web, Tim Berners-Lee.

> The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.
> *Tim Berners-Lee, W3C Director and inventor of the World Wide Web* [1]

Behinderte Menschen haben gerade auch bei der Nutzung des Internets eine Reihe von technischen Barrieren zu überwinden, die den meisten Webdesignern nicht bewusst sind.

Bereits die Verwendung von Textbrowsern oder allein die fehlende Fähigkeit, Farben zu unterscheiden, schränken die Usability vieler Webseiten enorm ein.

Am offensichtlichsten sind die Probleme blinder Personen. Verzichtet man auf alle Bilder, Farben und einen Großteil der Formatierungen, sind viele Webseiten nur mehr sehr eingeschränkt nutzbar.

Nur wenigen Webdesignern oder Projektverantwortlichen ist bewusst, dass es sich bei behinderten Menschen um keine kleine Gruppe von Personen handelt. Beispielsweise leiden in Österreich nach dem letzten Mikrozensus von 1995 3,1 Millionen Menschen unter einer Sehbeeinträchtigung. Bei mehr als 400.000 Personen kann diese nicht durch eine Brille oder ein anderes Hilfsmittel behoben werden. Auf beiden Augen vollkommen blind sind davon etwa 4.600 Personen[1] [3] [4].

Eine Untersuchung am Georgia Institute of Technology im Jahr 1996 ergab sogar, dass etwa 8% der Internetbenutzer eine Behinderung haben [5].

Natürlich gibt es eine Reihe von Programmen und technischen Hilfsmitteln für behinderte Menschen, mit denen das Lesen von Internetseiten erleichtert oder erst ermöglicht wird. Dazu gehören unter anderem Screenreader, die den Text einer Webseite vorlesen, Braillezeilen, die das Lesen von Webseiten in Blindenschrift ermöglichen, Bildschirmlupen, Maus-Emulatoren [6] [7] [8] [9] [10]. Doch gerade diese Programme sind darauf angewiesen, dass die Webseiten entsprechend gestaltet sind.

Der deutsche Begriff des „barrierefreien Webdesigns“ meint also, dass die Barrieren, die behinderten Menschen bei der Benutzung des Internets in den Weg gestellt sind, abgebaut und in Zukunft vermieden werden. Da diese Barrieren allerdings nicht so offensichtlich sind wie eine Treppe für eine Person im Rollstuhl oder eine Ampel für eine blinde Person, ist in diesem Bereich vor allem eine Bewusstseinsbildung bei Webdesignern nötig.

Der im Englischen gebrauchte Begriff „web accessibility“ (Zugänglichkeit) besitzt eine erheblich weitere Bedeutung als das deutsche „barrierefreie Webdesign“:

> The set of properties that allows a product, service, or facility to be used by people with a wide range of capabilities, either directly or in conjunction with assistive technologies. Although the term ‘accessibility’ typically addresses users who have a disability, the concept is not limited to disability issues. *Human Factors and Ergonomics Society* [11]

„Accessible web design“ hat nicht nur behinderte Personen im Blickfeld, sondern hat allgemein ein „gutes Webdesign“ zum Ziel. Das Problem an “schlechtem Design“ ist, dass es vor allem behinderten Personen Schwierigkeiten bereitet, indem bei der Konzeption ausschließlich an die Benutzer von Standardbrowsern gedacht wird.

[1] Die Zahl der „funktional blinden“ Personen liegt zwischen 8.000 und 12.000.

Jedes gute Webdesign ist somit gleichzeitig „accessible“ oder barreirefreies Webdesign [12].

Barrierefreies Webdesign wird oft irrtümlich mit ästhetisch wenig ansprechenden Webseiten gleichgesetzt. Bei der Gestaltung von gut zugänglicher Information muss aber auf „neue Technologien“ nicht verzichtet werden. Das Anliegen des barrierefreien Webdesign ist es vielmehr, beim Design von Webseiten die Abstimmung auf die speziellen Bedürfnisse behinderter Menschen als zusätzlichen Aspekt mit einzubeziehen.

Darauf bezieht sich auch das „Universelle Webdesign“, das von Michael Coopers „Universellem Design“ abgeleitet ist:

> Universal design of technology refers to software and hardware features that are created with a wide range of users in mind [13].

Bei der Gestaltung von Webseiten soll gemäß dieser Aussage an eine möglichst breit gefächerte Gruppe von Menschen gedacht werden, um Probleme bei der Bedienung bereits im Vorfeld zu vermeiden. Ein solches Problem ist die Vielzahl an aktuellen Browsern. Bei der Gestaltung von Webseiten muss darauf geachtet werden, dass diese auf allen gängigen Browsern verwendbar sind [14].

Dies bedeutet natürlich nicht, dass die Vorteile neuerer Browser und aktueller HTML-Spezifikationen nicht eingesetzt werden dürfen. Die Seite soll auch auf älteren Browsern noch lesbar sein, auch wenn das graphische Design dort, z.B. durch fehlende Stylesheets, nicht dem ursprünglichen Design entspricht.

1.1. Guidelines

Um den oben genannten Anforderungen gerecht zu werden, hat das World Wide Web Consortium (W3C) Richtlinien zur Gestaltung von Webseiten herausgeben. Diese wurden im Rahmen der WAI [1] (Web Accessibility Initiative) in einer weltweiten Kooperation entwickelt. Sie beinhalten die Grundidee des barrierefreien Webdesigns und sind auf der Seite der W3C [15] veröffentlicht.

Die W3C hat im Zuge dieser Initiative „Guidelines“, „Checklists“ und „Techniques“ entworfen, die als Grundlage für barrierefreies Webdesign dienen. Im Wesentlichen werden dabei die bereits bestehenden und meist unbeachteten Punkte der HTML-Spezifikation erläutert, die von Anfang an für barrierefreies Webdesign gedacht waren.

Der im Rahmen dieser Diplomarbeit entwickelte Kurs vermittelt die wichtigsten Inhalte dieser Richtlinie und bringt eine Reihe von Beispielen, die die damit verknüpften Konzepte verdeutlichen.

Im WWW sind eine Reihe weiterer Regeln zu finden, die aber im Wesentlichen den selben Inhalt haben. Eine Bestimmung, die vor allem für Webseiten der Regierung der Vereinigten Staaten Amerikas maßgeblich ist, ist Section 508 [16], ein Teil des Antidiskriminierungsgesetzes. Seit 1998 müssen demzufolge alle Webseiten der amerikanischen Bundesregierung diesen behindertengerechten Leitsätzen entsprechen.

Auch in Deutschland existiert ein Gleichstellungsgesetz, das vorschreibt, dass Internetseiten von öffentlichen Institutionen so zu gestalten sind, dass sie auch von behinderten Personen uneingeschränkt verwendet werden können.

In Österreich wird zur Zeit an einem Gesetz gearbeitet, das staatlichen Stellen vorschreibt, ihre Webseiten barrierefrei zu gestalten. Auch im Artikel 7 der österreichischen Bundesverfassung ist festgehalten, dass niemand wegen seiner Behinderung benachteiligt werden darf, und somit die Gleichbehandlung von behinderten und nichtbehinderten Menschen in allen Bereichen des täglichen Lebens gewährleistet werden soll [17].

Im Juni 2000 verabschiedete die EU den Aktionsplan „eEurope 2002“ zur Förderung der Verwendung des Internets. Dadurch wurde auch für Österreich festgelegt, dass niemand von Informationen ausgeschlossen werden darf, und

> that special attention should be given to disabled people and the fight against ‘info-exclusion’ [18].

Das Gelingen von barrierefreiem Webdesign hängt vom Einsatz eines jeden Einzelnen ab [19]. Vom einzelnen Designer bis hin zu den großen Organisationen kann und muss jeder seinen Beitrag dazu leisten, um das Internet zu einem Medium zu machen, das von allen Menschen uneingeschränkt verwendet werden kann.

Ein weniger „technisch orientierter“ Ansatz zur Vermittlung der Idee des barrierefreien Webdesigns findet sich in einem Workshop der Butler Universität, der sich an Teilnehmer ohne HTML-Kenntnisse richtet [20]:

- Be consistent. Predictable design and navigation greatly enhance accessibility.
- Be clear. Describe what you are doing and how you are doing it, rather than relying solely on visual information.
- Be sensible. All visitors will appreciate good organization and clean design. Let your pages breathe, figuratively and literally, so that people can find the information you have gathered.

Auch hier findet sich der Gedanke, dass jedes gute Webdesign zugleich barrierefreies Webdesign ist.

2. Curriculum

2.1. Zielgruppe

Der Kurs richtet sich in erster Linie an Webdesigner und allgemein an alle Personen, die mit der Erstellung und Gestaltung von Webseiten vertraut sind. Die Aufgabe jedes Designers ist es, sowohl graphisch ansprechende Seiten zu gestalten, als auch deren möglichst gute Usability zu gewährleisten. In diesem Sinne eignet sich der Kurs auch für Grafiker, Projektleiter und andere Personen, die im Internetbereich tätig sind, auch wenn sie die Webseiten nicht selbst schreiben.

Für alle diese unterschiedlichen Personengruppen ist das Wichtigste die Bewusstseinsbildung im Bereich „barrierefreies Webdesign“ – die Qualität von Webseiten hängt nicht nur vom graphischen Design ab, sondern auch – oder vor allem davon – dass sie möglichst benutzerfreundlich sind.

Besonders für Seiten von öffentlichen Einrichtungen ist es wichtig, bei der Gestaltung darauf zu achten, dass sie mit allen Arten von Browsern sowie mit entsprechenden Lesehilfen für behinderte Menschen darstellbar sind. Es ist jedoch auch im Interesse des Großteils der Webseitenanbieter, dass ihre Seiten von einer Vielzahl von Personen besucht und verwendet werden können. Insofern spricht der Kurs auch Personen an, die im Rahmen eines Projektes dafür zuständig sind, Webseiten zu gestalten und zu betreuen.

Die Voraussetzung, um möglichst viel von diesem Kurs zu profitieren, sind grundlegende Kenntnisse von HTML (Hypertext Markup Language) und CSS (Cascading Stylesheets).

2.2. Ziel des Kurses

Der Kurs verfolgt im Wesentlichen zwei Ziele, die untrennbar miteinander verbunden sind.

Einerseits soll er Wissen und Techniken vermitteln, wie barrierefreie Webseiten erstellt werden können, andererseits soll er helfen, bei bestehenden Projekten Probleme zu erkennen und die Seiten entsprechend barrierefrei umzugestalten. Der Kurs orientiert sich an den von der W3C herausgegebenen WAI-Richtlinen. Mit zahlreichen Beispie-

len werden die einzelnen Konzepte und Techniken veranschaulicht. Zur Umsetzung des dargebotenen Stoffes werden Aufgaben und Lösungen zur Verfügung gestellt.

Andererseits wird versucht, ein Bewusstsein für die Problematik zu vermitteln, vor der Menschen mit speziellen Bedürfnissen stehen. Es wird ein Einblick in die Schwierigkeiten von Personen gegeben, die z.B. einen Text- oder Voicebrowser verwenden, oder keine Maus zur Verfügung haben.

Generell verfolgt der Kurs natürlich das Ziel, gutes, und somit barrierefreies Webdesign zu fördern, um behinderten Menschen, und Personen mit speziellen Bedürfnissen den Zugang zum Internet und den darin gebotenen Inhalten, zu erleichtern.

2.3. Didaktische Überlegungen

Es kann davon ausgegangen werden, dass die Teilnehmer grundlegende Kenntnisse von Struktur und Aufbau sowie der Erstellung von Webseiten haben. Bei diesen Themen setzt der Kurs an, da sie als Grundlage für das barrierefreie Webdesign dienen. Um den Inhalt des Kurses greifbar zu machen, wurde eine Vielzahl von Beispielen verwendet. Zumeist handelt es sich dabei sowohl um den Sourcecode, als auch um einen Screenshot der daraus resultierenden Webseite. Dadurch können die Teilnehmer die Themengebiete sofort umsetzen und in die Praxis übernehmen. Außerdem sind Beispiele rein psychologisch eine gute Möglichkeit, Dargebotenes besser zu behalten.

Natürlich eignet sich das Internet als Medium zur Vermittlung dieser Beispiele am besten, da ein Experimentieren mit den Beispielen direkt möglich ist. Dadurch ist der Onlinekurs besonders geeignet, um sich auch selbstständig und praktisch mit der Materie auseinanderzusetzen. Aber auch im Skriptum, das als Unterstützung für den Vortrag gedacht ist, sind sowohl Teile des Sourcecodes als auch die Screenshots abgedruckt.

Der Kurs hält sich inhaltlich großteils an die WAI-Richtlinien, ist jedoch anders aufgebaut, da er zur besseren Vermittlung nach praktischen Gesichtspunkten geordnet ist.

Um die Beispiele kürzer und prägnanter zu gestalten, und um nicht von der eigentlichen Intention abzulenken, sind diese eigens für den Kurs erstellt und nicht von bestehenden Webseiten kopiert.

Der Aufgabenblock, der als Vertiefung des Kurses dient, ist bewusst aus der Praxis genommen: Es handelt sich um ein zusammenhängendes Projekt mit mehreren Seiten, die unterschiedliche Informationen darstellen und entsprechend gestaltet werden sollen. Zu Beginn erhalten die Teilnehmer eine Seite, wie man sie häufig im Internet findet. Diese wird dann Schritt für Schritt umgestaltet, sodass am Ende eine barrierefreie Webseite vorliegt.

Dies zeigt einerseits die Beispiele im großen Zusammenhang, andererseits wird ver-

deutlicht, dass die Umgestaltung einer bereits existierenden, nicht unter dem Aspekt des barrierefreien Webdesigns erstellten Webseite, durchaus möglich ist, und dass sich der Aufwand dafür in Grenzen hält und sich auch lohnt.

2.4. Themen

2.4.1. Einführung in das barrierefreie Webdesign

Die Einführung dient dazu, den Teilnehmenden einen Einblick in die Thematik zu geben. HTML-Konformität wird als Grundlage des barrierefreien Webdesigns vorgestellt.

Inhalte

- Gestalten HTML-konformer Webseiten.
- Erste Elemente des barrierefreien Webdesigns.
- Vorstellung von einschlägigen Online-Tools.

2.4.2. Gestaltung äquivalenter Textinformation

Um nicht-textuelle Inhalte auch für Text- und Voicebrowser zugänglich zu machen, muss für diese eine äquivalente Textinformation zur Verfügung gestellt werden. An Hand von Beispielen werden die unterschiedlichen Einsatzmöglichkeiten und -notwendigkeiten vorgestellt.

Inhalte

- Beschreibung von Bildern und Platzhaltern mit Hilfe des `alt`-Attributes.
- Kategorisierung von Bildern für die Wahl von `alt`-Texten.
- Ausführliche Beschreibungen mit Hilfe des `longdesc`-Attributes oder eines D-Links.
- Gestaltung von Listen mit Bildern als Aufzählungszeichen.
- Handhabung von Tooltipps.

2.4.3. Markup & Stylesheets

Ein zentrales Thema ist die fachgerechte Verwendung von HTML-Markup gemäß der HTML 4 Spezifikation, da falscher Einsatz zu Schwierigkeiten bei der Verwendung von manchen Browsern führt.

Weiters wird aufgezeigt, wie mit der Verwendung von Stylesheets Seiten flexibel gestaltet werden können, sodass sie für möglichst viele User gut lesbar sind.

Inhalte

- Einführung in HTML 4 mit den dazugehörigen Standards.
- Unterschiede zwischen structural und presentational Markup.
- Gestaltung von Überschriften, Absätzen, Listen und Zitaten.
- Verwendung von relativen und absoluten Größenangaben bei Stylesheets.
- Beachten von unterschiedlichen Bildschirmgrößen.
- Verschiedene Testmöglichkeiten von Webseiten.

2.4.4. Layout & Navigation

Vor- und Nachteile gängiger Navigationstechniken werden vorgestellt und mögliche Verbesserungsansätze werden zur Diskussion gestellt. Außerdem wird in diesem Kapitel eine Einführung in die Farbenlehre gegeben, um auf die Effekte verschiedener Farbkombinationen aufmerksam zu machen.

Inhalte

- Vor- und Nachteile gängiger Navigationstechniken.
- Korrekte Tabellenstruktur für Layouttabellen.
- Verwendung von Skip-Links.
- Erstellung eines „Table of Contents“.
- Gestaltung von Linklisten zur Förderung der Lesbarkeit.
- Metadaten zum Spezifizieren von Zusammenhängen zwischen Dokumenten.
- Grundlagen über Farben und Schriftarten.

2.4.5. Links & Image-Maps

Ein besonders brisantes Thema ist die Gestaltung von Links und Image-Maps. Bei deren unsachgemäßer Gestaltung erschweren sie die Navigation vor allem für blinde Personen. Die zentralen Techniken zur fachgerechten Gestaltung von barrierefreien Image-Maps werden vorgestellt.

Inhalte

- Wahl von Linktexten und „Front Loading“.
- Verwendung des `title`-Attributes.
- Gruppieren von Links zu sinnvollen Einheiten.
- Clientside und serverside Image-Maps.

2.4.6. Verwendung von Tabellen

Beim modernen Webdesign wird eine Vielzahl von zum Teil verschachtelten Tabellen verwendet. Damit die Zusammenhänge der Daten einer Tabelle von Browsern richtig aufbereitet werden können, müssen bestimmte Vorkehrungen getroffen werden.

Inhalte

- Unterschiede zwischen Layout- und Datentabellen.
- Angabe zusätzlicher Information mit Hilfe des `caption`-Tags und des `summary`-Attributes.
- Strukturieren von Tabellen mit dem `headers`-Attribut.
- Verwendung relativer Größenangaben.

2.4.7. Formulare

Es soll vermittelt werden, dass bei den immer zahlreicher verwendeten Formularen im Internet eine logische Gruppierung der einzelnen Formularelemente wichtig ist. Der logische Aufbau soll durch klare Zuordnung von Überschriften und Formularelementen ersichtlich gemacht, und der Umstand, dass Formulare auch ohne die Verwendung einer Maus oder Javascript verwendbar sind, aufgezeigt werden.

Inhalte

- Gruppieren von Formularelementen mit dem `fieldset`- und dem `optgroup`-Element.
- Günstige und ungünstige Anordnung von Formularelementen.
- Verwendung von Labels und Titles, zur Verdeutlichung von Zusammenhängen zwischen Formularelementen und Beschriftung.
- Navigation mit der Tastatur.
- Schwierigkeiten bei der Verwendung von Javascript.

2.4.8. Frames

Ein generell umstrittenes und für das barrierefreie Webdesign problematisches Thema ist die Verwendung von Frames. Es wird aufgezeigt, dass einige Browser nicht gut geeignet sind, Frames darzustellen. Daher wird darauf hingewiesen, wie man Frameseiten durch zusätzliche Information zugänglicher macht.

Inhalte

- Schwierigkeiten bei der Verwendung von Frames.
- Einsatz des `noframes`-Tags.
- Navigationshilfe mit dem `name`- und dem `title`-Attribut.
- Die Verwendung von alternativen Seiten als „letzter Ausweg“.

2.4.9. Sprache

Bei der Gestaltung von Webseiten ist eine einfache Sprache dem Verständnis und somit der allgemeinen Zugänglichkeit förderlich. Es soll aufgezeigt werden, welche sprachlichen Mittel und Normen zu einer besseren Lesbarkeit beitragen.

Inhalte

- Hauptsprache und Sprachwechsel auf einer Webseite.
- Abkürzungen und Akronyme.
- Probleme bei der Political Correctness.

2.4.10. Browser Kompatibilität

Accessible Webdesign umfasst auch die Rücksichtnahme auf die unterschiedlichen Funktionsweisen der verschiedenen Browser, die derzeit auf dem Markt sind. Daher muss bei der Verwendung gewisser Technologien wie zum Beispiel Javascript darauf geachtet werden, dass die dadurch dargebotene Information auch anderwärtig verfügbar ist.

Inhalte

- Verschiedene Szenarien bei der Verwendung von Javascript.
- Interaktion mit Webelementen ohne Maus.
- Verwendung alternativer Seiten.

2.4.11. Hilfsmittel

Derzeit befindet sich eine Reihe von Hilfsmitteln zur Entwicklung barrierefreier Webseiten auf dem Markt. Die wichtigsten sollen den Kursteilnehmern vorgestellt werden.

Inhalte

- W3C HTML Validation Service
- Bobby
- WAVE
- HTML-Tidy
- A-Prompt
- Lynx
- Opera
- IBM Home Page Reader
- pwWebSpeak

Literaturverzeichnis

[1] *Web Accessibility Initiative (WAI)*, 2003, `http://www.w3.org/WAI/`

[2] Zagler Wolfgang: *Kommunikationstechnik für behinderte und alte Menschen*, Skriptum zur gleichnamigen Vorlesung an der TU-Wien, 2003

[3] *Mikrozensus Juni 1995*, `http://www.gesis.org/Dauerbeobachtung/Mikrodaten/Daten/Abteilungsdaten/Mikrozensen/mz_1995/doc95.htm`

[4] *Auszug aus: „Personen mit körperlichen Beeinträchtigungen.“ des Mikrozensus 1995* `http://kremser.wonne.cc/statistik/sehbehinderung.html` (8.6.2003)

[5] National Council on Disability: *Access to Multimedia Technology by People with Sensory Disabilities*
`http://www.ncd.gov/newsroom/publications/sensory.html` (11.8.2003)

[6] Djamel Hadjadj, Burger Dominique: *BrailleSurf: An HTML Browser for visually handicapped people*, Technology and Persons with Disabilities Conference 1999

[7] Laws Catherine, Asakawa Chieko: *IBM Home Page Reader: The Voice of the Word Wide Web*, Technology and Persons with Disabilities Conference 1999

[8] D'Amour Jean-Marie: *How Assistive Software Supports Web Accessibility*, Technology and Persons with Disabilities Conference 2002

[9] Takagi Hironobu, Asakawa Chieko: *Web Content Transcoding for Voice Output*, Technology and Persons with Disabilities Conference 2002

[10] Whittaker Curt, u.a.: *Adaptive Technology at Southern Oregon University*, White Paper, May 2001

[11] Human Factors Engineering of Software User Interfaces: *ANSI/HFES 200.2 Part 2: Accessibility (May, 2000 Draft)*

[12] Pavka Anitra, Digital Web Magazine: *Accessible By Design*,
`http://www.digital-web.com/features/feature_2002-04.shtml` (14.6.2003)

[13] Cooper Michael: *Universal Design of a Web Site*, Technology and Persons with Disabilities Conference 1999

[14] *Viewable with Any Browser Campaign*, `http://www.anybrowser.org/campaign`

[15] *World Wide Web Consortium*, `http://www.w3.org/`

[16] *NCI's Web Accessibility Plans: Responding to Section 508*, `http://oc.nci.nih.gov/web508/index.html` (23.6.2003)

[17] *Auszug aus dem Artikel 7 der österreichischen Bundesverfassung*, `http://www.service4u.at/info/VERFASS.html` (16.6.2003)

[18] *eEurope 2002 Action Plan*, `http://europa.eu.int/information_society/eeurope/action_plan/index_en.htm` (14.6.2003)

[19] Brewer Judy, Dardailler Daniel: *W3C web accessibility initiative: progress, challenges, resources*, Technology and Persons with Disabilities Conference 1999

[20] McCain Tom: *A Common Sense Approach to Web Accessibility* Technology and Persons with Disabilities Conference 1999

Teil II.

Skriptum

Skriptum

Barrierefreies Webdesign

von Christoph Rettinger

im Rahmen meiner Diplomarbeit
an der Technischen Universität Wien

© September 2003

Aus Gründen der Lesbarkeit wurde im Text die männliche Form gewählt, nichtsdestoweniger beziehen sich die Angaben auf Angehörige beider Geschlechter.

Inhaltsverzeichnis

1 Einführung **7**
1.1 Barrierefreies Webdesign / Accessible Web-Design 7
1.2 Ein erstes Beispiel 8

2 Äquivalente Text-Information **13**
2.1 Das alt-Attribut 15
2.2 Das longdesc-Attribut 16
2.3 D-Links 16
2.4 Beschreibung von Bildern mit äquivalenter Text-Information 18

3 Markup & Stylesheets I **25**
3.1 HTML 4 26
3.2 Structural vs. Presentational Markup 28
3.3 Korrektes Verwenden von Markup 29

4 Markup & Stylesheets II **37**
4.1 Relative und absolute Größen 37
4.2 Bildschirmauflösungen 38
4.3 Das Testen von Webseiten 41

5 Layout & Navigation **47**
5.1 Grundlagen der Navigation 47
5.2 Verbesserte Tabellenstruktur 48
5.3 Skip Links 48
5.4 Table of Contents / Site Map 51
5.5 Linklisten 53
5.6 Meta-Information 54
5.7 Farben 57
5.8 Schriftarten 60

6 Links & Image-Maps **61**
6.1 Links 61
6.2 Image-Maps 64

7 Tabellen **69**

7.1 caption-Tags und summary-Attribute 70
7.2 Das headers-Attribut 72
7.3 Relative Größenangaben 73

8 Formulare **75**
8.1 Gruppieren von Informationen 75
8.2 Das Anordnen von Formularfeldern 78
8.3 Labels 81
8.4 Titles 81
8.5 Navigation mit der Tastatur 83

9 Frames **87**
9.1 Das noframes-Tag 88
9.2 Das name- und das title-Attribut 88
9.3 Alternative Seiten 91

10 Sprache **93**
10.1 Spezifikation der Sprache 93
10.2 Abkürzungen und Akronyme 94
10.3 Political Correctness 96

11 Browserkompatibilität **97**
11.1 Javascript 97
11.2 Alternative Seiten 106

12 Hilfsmittel **107**
12.1 Validatoren 107
12.2 Korrigierende Tools 112
12.3 Browser 116

Vorwort

Das World Wide Web (WWW) nimmt in unserer Gesellschaft als Kommunikations- und Informationsmedium eine zentrale Rolle ein. Viele Tätigkeiten sind sowohl im geschäftlichen als auch im privaten Leben ohne das Internet nicht mehr denkbar. Zahlreiche Firmen bieten ihre Produkte beispielsweise ausschließlich über das Internet an, an den Wiener Universitätsbibliotheken muss man für die Suche nach Literatur einen Web-Browser verwenden.

Gerade deshalb ist es wichtig, allen Bevölkerungsgruppen den Zugang zum Internet zu ermöglichen. Menschen mit Behinderungen wird jedoch oft unbewusst der Zugang zum Internet erschwert oder sogar unmöglich gemacht. Aus diesem Grund muss beim Design von Internetseiten darauf geachtet werden, dass sie für möglichst viele Benutzer gut zu verwenden sind.

> The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.
> *Tim Berners-Lee, W3C Director and 'inventor' of the World Wide Web*

Menschen mit Behinderungen stehen bereits Hilfsmittel zur Verfügung, um mit Computern im Allgemeinen und mit Webseiten im Speziellen umzugehen. Dazu zählen unter anderem Voice-Browser, Braille-Ausgabegeräte und Bildschirmlupen. Diese Programme können jedoch nur sehr eingeschränkt verwendet werden, wenn bei der Gestaltung von Webseiten nicht explizit Rücksicht auf deren besondere „Bedürfnisse“ genommen wird.

Ein einfaches Beispiel dafür, dass „normale“ Webseiten oft nicht den Ansprüchen behinderter oder anderwärtig eingeschränkter User entsprechen, sind Bilder, deren Information nicht zusätzlich textuell zur Verfügung gestellt wird. Blinde Personen und Benutzer von Textbrowsern sind von diesen Informationen ausgeschlossen – die Zugänglichkeit solcher Seiten ist stark eingeschränkt.

Der deutsche Begriff des „barrierefreien Webdesigns“ meint, dass solche Einschränkungen und Barrieren, die behinderten Menschen bewusst oder unbewusst in den Weg gestellt sind, abgebaut, und in Zukunft vermieden werden. Da diese Barrieren allerdings nicht so offensichtlich sind wie eine Treppe für eine Person im Rollstuhl oder eine Ampel

für eine blinde Person, ist in diesem Bereich vor allem eine Bewusstseinsbildung bei Webdesignern nötig.

Das Ziel von barrierefreiem Webdesign ist nicht, keine Bilder mehr zu verwenden oder gänzlich auf Multimedia-Inhalte zu verzichten, sondern Webseiten so zu gestalten, dass sie auch ohne Bilder zu verwenden sind.

Die Techniken und Werkzeuge, die behinderten Menschen zur Nutzung des Computers und somit auch des Internets zur Verfügung stehen, können nur dann eingesetzt werden, wenn bestimmte Richtlinien in der Gestaltung von Webseiten berücksichtigt werden. In internationaler Kooperation wurden solche Richtlinien entwickelt. Die dafür verantwortliche Dachorganisation ist das W3C, das für Normungen im Internet zuständige Konsortium, die spezielle Initiative ist die Web Accessibility Initiative (WAI).

In diesem Kurs geht es um die Umsetzung dieser Richtlinien. Anhand von praktischen Beispielen werden die einzelnen Punkte der WAI Richtlinien behandelt, und durch Aufgaben und Musterlösungen die zu erlernenden Prinzipien und die damit verbundenen Techniken verdeutlicht.

Die einzigen Voraussetzungen für die Teilnahme an diesem Kurs sind Offenheit für das Thema und grundlegende HTML Kenntnisse.

1. Einführung

1.1. Barrierefreies Webdesign / Accessible Web-Design

Im englischen Sprachraum haben sich bereits die Begriffe „accessible technologies“ und „accessible web design“ eingebürgert. „Accessible“ bedeutet übersetzt „zugänglich“ oder „erreichbar“ und meint in diesem Zusammenhang, dass die auf Webseiten befindlichen Informationen in unterschiedlicher Weise zugänglich sind, man könnte auch sagen „alle Sinne ansprechend“. Dadurch werden sie nicht nur für behinderte Personen leichter zugänglich.

In diesem Sinne des „accessible web design“ ist ein System, das seine Information nur via Audio anbietet, für gehörlose Personen nicht „accessible“.

Im Deutschen gibt es noch keinen gleichwertigen Begriff für das englische „accessible web design“. In diesem Kurs verwende ich dafür den deutschen Begriff „barrierefreies Webdesign“. Als „barrierefreies Webdesign“ kann also allgemein das Vorhaben bezeichnet werden, Webseiten so zu gestalten, dass behinderte Menschen keinen Zugangsbeschränkungen für Webseiten aufgrund deren Design und Struktur begegnen.

Es geht also darum, Barrieren zu erkennen und beim Design von Webseiten zu vermeiden.

Beschäftigt man sich mit barrierefreiem Webdesign, muss man sich bewusst machen, dass viele Benutzer unter bestimmten Einschränkungen im Internet surfen:

- Sie leiden unter Beeinträchtigungen beim Sehen oder Hören und haben dadurch Schwierigkeiten beim Navigieren in Web-Dokumenten.
- Sie haben Schwierigkeiten beim Lesen und Verstehen von Texten.
- Sie sind auf andere Eingabegeräte als Maus und Tastatur angewiesen.
- Sie verwenden kleine Bildschirme oder reine Textausgabe.
- Sie haben langsame Internetverbindungen.
- Sie befinden sich in Situationen, in denen Augen, Ohren oder Hände mit anderen Tätigkeiten beschäftigt sind.

- Sie verwenden ältere und alternative Browser (Voice-Browser, Braille-Browser), und verschiedene Betriebssysteme.

Um als Web-Designer dennoch zu gewährleisten, dass die Webseiten für Menschen mit verschieden Beeinträchtigungen zugänglich sind, könnte man die Seiten von entsprechend vielen Usern testen lassen und sie nach deren Feedback umgestalten, doch diesen Aufwand kann sich wohl kaum ein Entwicklerteam leisten.

Als Hilfsmittel und Richtlinien für Webdesigner wurde deshalb eine Reihe von Standards entwickelt, die als Grundlage des Webdesigns verwendet werden sollten. Durch das Einhalten dieser Richtlinien kann gewährleistet werden, dass Menschen trotz unterschiedlicher Einschränkungen besseren Zugang zum Internet erhalten.

In zwei Bereichen sind Standards für das World Wide Web definiert:

- **HTML**: Indem man sich an die HTML-Spezifikationen hält, wird gewährleistet, dass verschiedene Browser die Inhalte darstellen können. Die korrekte Verwendung von Markup und Stylesheets ist ein weiterer wichtiger Aspekt.
- **Behindertengerechtes Design**: Zusätzlich zur HTML-Konformität existiert ein maßgeblicher Standard im WWW zur barrierefreien Gestaltung von Webseiten. Dieser ist die von der Internet-Normierungsinstitution, der W3C, getragene „World Wide Web Accessibility Initiative[1]“.

1.2. Ein erstes Beispiel

Zu Beginn werden einige Aspekte vorgestellt, die beim barrierefreien Webdesign von zentraler Bedeutung sind. Anhand einer einfachen Webseite wird versucht, die ersten, scheinbar kleinen Verbesserungen vorzustellen, die Webseiten der Zugänglichkeit schon ein großes Stück näher bringen.

Zum Testen der Webseite werden zwei Programme verwendet, die sich im Erstellen barrierefreier Webseiten als sehr hilfreich erweisen. Sie überprüfen, ob Webseiten einem gewissen Standard entsprechen. Mit diesen Tools lassen sich viele Fehler und Unsauberkeiten ausbessern, auch wenn sie nicht garantieren können, dass Webseiten dann „perfekt barrierefrei“ sind.

Im folgenden Beispiel werden diese Online-Tools verwendet:

- **W3C HTML Validation Service**: `http://validator.w3.org/`
- **Bobby**: `http://cast.org/bobby/`

[1] `http://www.w3.org/WAI/`

Eine einfache Webseite

In Abbildung 1.1 ist der Code einer einfachen HTML-Seite zu sehen. Es handelt sich dabei um die schlichte Vorstellung eines Hotels, wie sie auf den verschiedensten Fremdenverkehrsseiten zu finden sein könnte. Abbildung 1.2 zeigt die Ausgabe in einem graphischen Browser.

```
<html>
<head>
   <title>Hotel Steindl</title>
</head>
<div style="color:#00235C"><b>Hotel Steindl</b> 
   <img src="../images/stars5.gif" width="60"></div>
<table width="500">
   <tr>
      <td>
         Das elegante Hotel Steindl liegt im Herzen Wiens an der
         berühmten Ringstraße, einem der schönsten Boulevards
         der Welt. <br> Wenige Gehminuten vom Hotel
         entfernt befinden sich bekannte Sehenswürdigkeiten wie
         das Burgtheater, ...
      </td>
      <td><img src="../images/hotel.jpg" width="200"></td>
   </tr>
</table>
</html>
```

Abbildung 1.1.: Sourcecode einer schlichten Webseite.

Abbildung 1.2.: Darstellung des Beispiels in einem graphischen Browser.

HTML-Konformität

Der erste Schritt dahin, dass obige Seite barrierefrei wird, ist, sie auf HTML-Konformität zu überprüfen. Dies geschieht am besten mit dem W3C HTML Validation Service. Validiert man die Seite so, wie sie hier dargestellt wurde, erhält man folgende Fehlermeldungen:

- **No Character Encoding detected!**
  Im Sourcecode der Webseite muss angegeben werden, welcher Zeichencode dem Source zu Grunde liegt. Dadurch können das Validation Service und vor allem die Browser den Code erst richtig interpretieren.

- **Fatal Error: no document type declaration.**
  Ebenso muss angegeben werden, aufgrund welcher HTML-Version das Dokument validierbar sein soll. Zurzeit gibt es drei gängige Versionen: HTML 4.01 strict, HTML 4.01 transitional und HTML 4.01 frameset. Auf die Unterschiede wird im Kapitel 3 „Markup & Stylesheets“ eingegangen.

Um anzugeben, dass als Character-Encoding die ISO-8859-1 Norm, und als HTML-Version die Version 4.01 verwendet wird, werden in den Sourcecode die folgenden zwei Zeilen hinzugefügt:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

Somit kann die Seite validiert werden. Beim nächsten Validierungsversuch der Seite stellt sich heraus, dass bei den beiden Bildern kein `alt`-Attribut (alternativer Anzeigetext) gesetzt ist, also keine äquivalente Textinformation der Graphiken zur Verfügung gestellt wurde. Dieses `alt`-Attribut ist eines der wichtigsten Hilfsmittel, Webseiten mit Graphiken für blinde Menschen, Personen mit Sehschwierigkeiten und Personen, die nur einen Text-Browser verwenden können, zugänglich zu machen.

Darauf, welchen Text man im `alt`-Attribut einer bestimmten Graphik verwenden sollte, wird im nächsten Kapitel ausführlich eingegangen. In dem oben genannten Beispiel wären mögliche `alt`-Attribute „Zimmeransicht“ für das Bild des Hotelzimmers und „*****“ für die Graphik mit den fünf Sternen. Das Bild des Hotelzimmers könnte auch detaillierter beschrieben werden, zum Beispiel mit „Zimmeransicht mit Doppelbett“.

Nach diesen Änderungen ist die Webseite HTML 4.01 konform, das W3C HTML Validation Service gibt auch keine Fehlermeldung mehr aus. Durch die zusätzlich eingefügte Information entspricht die Webseite nun dem HTML 4.01-Standard.

Accessible Design

Nachdem die Beispielseite HTML konform ist, soll sie zusätzlich den Anforderungen des Accessible Web-Designs gerecht werden. Zur Überprüfung der Webseite wird das Programm „Bobby“ verwendet. Bei der ersten Überprüfung des HTML-konformen Codes wird einerseits eine Reihe von Fehlermeldungen ausgegeben, andererseits wird der Benutzer aufgefordert, seine Seite nach zentralen Kriterien des Accessible Web-Designs zu überprüfen.

„Bobby“ gibt neben den Fehlermeldungen auch Aufforderungen zurück, die nicht maschinell überprüfbaren Eigenschaften selbst zu kontrollieren. Zu diesen Eigenschaften gehört zum Beispiel der Farbkontrast der Seite oder das Verwenden einer übersichtlichen Navigation. Da auf die meisten dieser Eigenschaften im Rahmen dieses Kurses noch genauer eingegangen wird, und da manche dieser Meldungen auf das simple Beispiel nicht zutreffen, wird im Folgenden nur auf die Fehlermeldungen und auf den für dieses Beispiel relevanten Hinweis zur Farbgestaltung des Textes eingegangen:

- **If you use color to convey information, make sure the information is also represented another way.**
  Wenn Information mittels Farbe dargestellt wird, muss gewährleistet sein, dass durch das Weglassen von Farbe keine Information verloren geht. In dem oben genannten Beispiel wird die Überschrift rot dargestellt, aber es ist sonst in keiner Weise gekennzeichnet, dass es sich um eine Überschrift handelt. Das sollte mit Hilfe des `<h1>`-Tags geschehen. So können auch Voice- und Textbrowser die Überschrift entsprechend präsentieren.

- **Use relative sizing and positioning (% values) rather than absolute (pixels).**
  Damit Webseiten gut skalierbar sind, soll man möglichst keine absoluten Größenangaben verwenden. Dadurch kann gewährleistet werden, dass die Webseite auch mit Monitoren, die nur eine kleinere Abmessung oder Auflösung haben, mit Textbildschirmen oder mit Screenreadern gut dargestellt werden können. Anstatt die Breite der Tabelle mit `500` anzugeben, wäre eine relative Angabe mit `50%` besser.

- **Provide a summary for tables.**
  Werden Tabellen dazu verwendet, komplexe Inhalte tabellarisch darzustellen, ist es wichtig, eine Zusammenfassung der Tabelle anzugeben, damit diese alternativ zur Tabelle dargestellt werden kann. In dem aktuellen Beispiel wird keine Zusammenfassung verwendet, da die Tabelle nur zur visuellen Gliederung dient. Deshalb wird bei der Tabelle `summary=""` angegeben, um anzuzeigen, dass für diese Tabelle keine Zusammenfassung notwendig ist.

- **Identify the language of the text.**
  In jeder Webseite sollte die Sprache angegeben werden, in der der Großteil des

Textes verfasst ist. Das ermöglicht z.B. Programmen, die Webseite mittels eines Sprach-Synthesizers vorzulesen. Die Sprache kann als `lang`-Attribut innerhalb des `<html>`-Tags angegeben werden.

Die nun barrierefreie Webseite wird in Abbildung 1.3 gezeigt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="ge">
<head>
   <title>Hotel Steindl</title>
   <meta http-equiv="Content-Type" content="text/html;
         charset=ISO-8859-1">
</head>
<h1 style="color:#00235C; font-size: 100%;"><b>Hotel Steindl</b>
    <img src="../images/stars5.gif" alt="*****" width="60"></h1>
<table width="50%" summary="">
   <tr>
      <td>
         Das elegante Hotel Steindl liegt im Herzen Wiens an der
         berühmten Ringstraße, einem der schönsten Boulevards
         der Welt. <br> Wenige Gehminuten vom Hotel
         entfernt befinden sich bekannte Sehenswürdigkeiten wie
         das Burgtheater, ...
      </td>
      <td>
         <img src="../images/hotel.jpg" width="200"
              alt="Zimmeransicht">
      </td>
   </tr>
</table>
</html>
```

Abbildung 1.3.: Die ausgebesserte HTML-Seite.

2. Äquivalente Text-Information

Eines der wichtigsten Prinzipien des barrierefreien Webdesign ist das Verwenden äquivalenter Text-Information für nicht-textuelle Inhalte wie Bilder, Videos, Flash-Animationen oder Audio-Dateien. Äquivalente Textinformation gibt die Information des nicht-textuellen Mediums so wieder, dass sie auch mit Text- und Voicebrowsern zugänglich ist. Dies geschieht vor allem mit Hilfe des `alt`-Attributes. Nicht nur blinde Personen oder Personen mit Sehbehinderungen, sondern auch Benutzer, die Text-Browser verwenden, mit einem mobilen Gerät die Webseite anschauen, oder die die Anzeige der Bilder bei ihrem Browser deaktiviert haben, haben sonst keine Möglichkeit, auf diese Information zuzugreifen. Für diese Benutzer muss eine text-äquivalente Information bereitgestellt werden, wenn die Bilder zum Verständnis der Seite nötig sind.

Ein Vorteil von äquivalenter Text-Information ist, dass sie dem Benutzer auf unterschiedliche Arten präsentiert werden kann. So kann die Information z.B. in Braille-Schrift für blinde Personen aufbereitet werden, und auch mit Hilfe eines Sprach-Synthesizers vorgelesen werden.

Ein weiterer Vorteil bei der Verwendung von `alt`-Attributen ist, dass die Texte durch „Indexing Robots“ erfasst werden, und somit die Webseiten auch aufgrund der Bildinformation von Search Engines gefunden werden.

Den Wert von äquivalenter Textinformation für die Zugänglichkeit von Webseiten erkennt man schnell, wenn man sich eine Webseite mit einem Textbrowser ansieht. Dafür eignet sich auch das für alle wichtigen Plattformen frei verfügbare Programm Lynx[1]. Dieses ist auch als Internet-basierender Emulator erhältlich: Lynx Viewer von Delorie[2].

Mit dem Beispiel in Abbildung 2.1 soll gezeigt werden, wie fehlende Information zu Problemen bei der Verständlichkeit und bei der Navigation auf Webseiten führen kann. Auf der Beispielseite soll zwischen zwei Sprachen gewählt werden, die jedoch nur von den Graphiken der Landesflaggen repräsentiert werden. Kann man diese nicht sehen und erhält stattdessen nur die internen Namen der Graphiken, ist man „verloren“. Wie die Ausgabe dieser Seite mit einem graphischen und einem textbasierenden Browser aussieht, zeigen Abbildung 2.2 und Abbildung 2.3.

Das Ergebnis ist unbrauchbar, wenn man die Information, die die Bilder vermitteln, nicht hat und dadurch nicht sieht, welche Sprache sich hinter welchem Link „verbirgt“.

[1] `http://lynx.browser.org`
[2] `http://www.delorie.com/web/lynxview.html`

```
<h1>Wählen Sie eine Sprache</h1>

<a href="sample-02-01a.html">
   <img src="/images/flag_a.gif" border="0">
</a>

<img src="/images/spacer.gif" width="10" height="1">

<a href="sample-02-01b.html">
   <img src="/images/flag_b.gif" border="0">
</a>
```

Abbildung 2.1.: Eine HTML-Seite zur Auswahl der gewünschten Sprache ohne alt-Attribute.

Abbildung 2.2.: Die Wahlmöglichkeit, von einem graphischen Browser dargestellt.

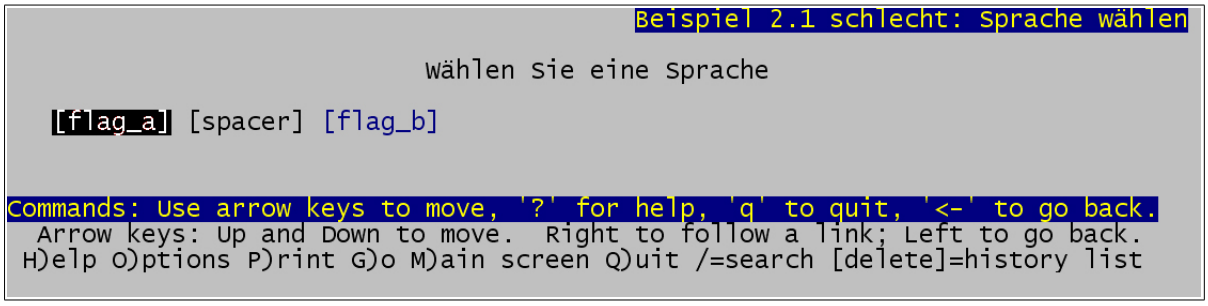


Abbildung 2.3.: Da bei den Graphiken keine alt-Attribute angegeben wurden, ist im Textbrowser nicht zu erkennen, zwischen welchen Sprachen man wählen kann. Man sieht nicht nur die aussagelosen Namen der Bilder, sondern zusätzlich auch den Namen eines ansonsten unsichtbaren Platzhalters.

Dem User bleibt dann nur die Möglichkeit, die einzelnen Links auszuprobieren, um zu erfahren, wohin sie führen. Wenn, wie im Beispiel in Abbildung 2.4, der Bildinformation entsprechende `alt`-Tags verwendet werden, ist die Darstellung in einem Text-Browser (Abbildung 2.5) auch zufrieden stellend und somit barrierefrei.

```
<h1>Wählen Sie eine Sprache</h1>

<a href="sample-02-01a.html">
   <img src="/images/flag_a.gif" alt="Deutsch" border="0">
</a>

<img src="/images/spacer.gif" alt="" width="10" height="1">

<a href="sample-02-01b.html">
   <img src="/images/flag_b.gif" alt="English" border="0">
</a>
```

Abbildung 2.4.: Auf der Webseite mit alt-Attributen besitzen die Graphiken ihrer Verwendung entsprechend gewählte Texte. Der Platzhalter ist durch das leere alt-Attribut auch in Textbrowsern nicht sichtbar.

```
                                                Beispiel 2.1 gut: Sprache wählen

                              Wählen Sie eine Sprache

   Deutsch English

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 2.5.: Unter der Verwendung von alt-Attributen kann die Seite auch mit Textbrowsern ohne Einschränkungen verwendet werden.

2.1. Das alt-Attribut

Wie bereits erwähnt, ist das `alt`-Attribut die wichtigste Möglichkeit, äquivalente Textinformation zur Verfügung zu stellen. Manche Browser stellen das `alt`-Attribut als Tooltip dar. Beim Opera-Browser ist es beispielsweise möglich, alle Bilder mit der Taste G auszublenden und stattdessen die `alt`-Texte anzuzeigen. Bei Text-Browsern ist es Standard, den `alt`-Text an Stelle der Bilder anzuzeigen. Daher sind `alt`-Attribute für barrierefreies Webdesign unverzichtbar. Bilder und Image-Maps sollen demzufolge immer in Kombination mit `alt`-Attributen verwendet werden. Auch bei Formular-Eingabeelementen und

Java Applets ist eine Verwendung von `alt`-Attributen meistens sinnvoll. Um HTML 4 konform zu arbeiten, sind `alt`-Attribute für Bilder und Image-Maps erforderlich.

Die folgende Tabelle listet die der HTML 4 -Konformität entsprechende Verwendung der `alt`-Attribute noch einmal auf:

`<img>`	Einbinden von Bildern	notwendig
`<area>`	Image maps	notwendig
`<input>`	Form-Eingabeelement	optional
`<applet>`	Einbinden von Java-Applets	optional

Innerhalb der `alt`-Attribute ist nur normaler Text erlaubt, das heißt keine HTML-Tags. Sonderzeichen können mit Hilfe der Zeichenkombinationen „&...;“ und „&#...;“ „umschrieben“ werden. Da die unterschiedlichen Browser individuell auf Zeilenumbrüche innerhalb des `alt`-Attributes reagieren, sollte man auf diese besser verzichten.

2.2. Das longdesc-Attribut

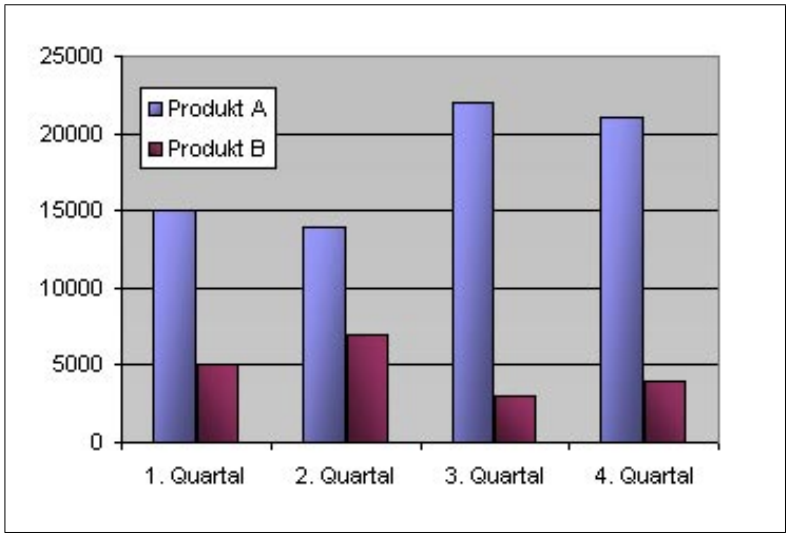
Um zu vermeiden, dass eine Seite durch zu ausführliche `alt`-Attribute unlesbar wird, sollten die `alt`-Texte möglichst kurz und bündig sein. Ist es zum Verständnis notwendig, eine noch genauere Beschreibung eines Bildes zu geben, kann auf eine Seite verwiesen werden, auf der das Bild ausführlich beschrieben ist. Zu diesem Zweck wurde mit der HTML 4 Spezifikation das `longdesc`-Attribut eingeführt. Es enthält einen Link auf die entsprechende Seite, der nur unter bestimmten Voraussetzungen dargestellt wird.

Zurzeit wird das `longdesc`-Attribut noch nicht von allen Browsern unterstützt. Daher wird im Weiteren eine Alternative dazu angeführt.

2.3. D-Links

Eine weitere, von Browserversionen unabhängige Möglichkeit, Bilder durch einen ausführlicheren Text zu erklären, ist mittels eines Description- oder D-Links. Direkt nach einem Bild wird ein „D“ eingefügt. Dieses dient als Link auf die Seite, auf der das Bild näher erklärt wird. Jeder Benutzer hat dadurch die Möglichkeit, dem Link zu folgen, um eine textuelle Darstellung der graphischen Information zu erhalten (Abbildung 2.6 und Abbildung 2.7).

Der Vorteil von D-Links gegenüber den `longdesc`-Attributen ist, dass sie auch bei älteren Browserversionen funktionieren, auch wenn sie graphisch nicht so ansprechend sind, da sie immer dargestellt werden.



D

Abbildung 2.6.: Verwendung eines D-Links. Die longdesc-Attribute werden in graphischen Browsern nicht dargestellt.

```
<img src="turnover.jpg" longdesc="turnover_desc.html">
<a href="turnover_desc.html">D</a>
```

Abbildung 2.7.: Verwendung eines D-Links und eines longdesc-Attributes zur genaueren Beschreibung eines Bildes.

2.4. Beschreibung von Bildern mit äquivalenter Text-Information

Um die Lesbarkeit und Zugänglichkeit einer Seite durch äquivalente Text-Information zu erhöhen, ist die richtige Wahl von `alt`-Texten von großer Bedeutung. Schlechte `alt`-Texte können genauso störend sein wie fehlende. Damit sich Texte in der Praxis als brauchbar erweisen, können einige Richtlinien zu deren Gestaltung angeben werden. Diese sollen das Verwenden von `alt`-Texten vereinfachen und helfen, Fehler zu vermeiden:

Die optimale Länge

Obwohl `alt`-Attribute eine alternative Beschreibung für das entsprechende Bild sein sollen, wird das `alt`-Attribut oft dazu missbraucht, eine zusätzliche Beschreibung für ein Bild anzugeben, oder zu beschreiben, wohin ein Link führt, der dem Bild hinterlegt ist. Damit wird jedoch der eigentliche Zweck verfehlt, denn Usern, die die Bilder nicht sehen können, wird auf diese Weise Information vorenthalten.

Sind `alt`-Texte zu lang, können sie zu einem großen Ärgernis werden. Wird auf einer Webseite ein kleines Backsteinhaus als Link zurück zur Eingangsseite eine Webseite verwendet, wäre der Text „Kleines Backsteinhäuschen mit rauchendem Schornstein“ ein unbrauchbarer Text. In diesem Fall sollte der Text z.B. „Hauptseite“, oder „Zur Hauptseite“ lauten.

Bilder erfüllen unterschiedliche Funktionen auf Webseiten. Je nach dem, welchem Zweck ein Bild auf einer Webseite dient, muss der `alt`-Text entsprechend gewählt werden.

Bilder als Gestaltungselemente

Wenn Bilder nur als Dekoration einer Seite verwendet werden, ist es ratsam, einen leeren `alt`-Text anzugeben (`alt=""`). Von den meisten Browsern wird der `alt`-Text dann nicht angezeigt. Dadurch müssen sich Benutzer von Voice- und Textbrowsern nicht mit der für sie irrelevanten Information beschäftigen.

Bei manchen Anwendungen können solche Gestaltungselemente auch eine entscheidende Rolle spielen, zum Beispiel bei der Wahl eines Hotelzimmers. In diesem Fall ist es dann am besten, das Bild mit einem kurzen und prägnanten `alt`-Text zu versehen, und gegebenenfalls einen Link auf eine genauere Beschreibung mit Hilfe eines `longdesc`-Attributes oder eines D-Links anzugeben.

„Spacer“

Unter einem „Spacer“ versteht man ein Bild, das dazu verwendet wird, Abstände zwischen HTML-Objekten zu erzeugen, z.B. zwischen zwei aneinander grenzenden Bildern. Da Spacer sehr häufig eingesetzt werden, aber keine Information enthalten, sollte auf jeden Fall darauf geachtet werden, dass sie für Benutzer von Text- und Voicebrowsern nicht sichtbar sind. Je nach Anwendung verwendet man dafür `alt=""`, oder, wenn auch bei einer reinen Textansicht ein Abstand nötig ist `alt=" "`.

Firmenlogos

Bei der Wahl des `alt`-Textes für Firmenlogos gibt es mehrere Möglichkeiten. Wenn der Firmenname bereits als normaler Text auf der Webseite vorkommt, ist es zumeist ausreichend `alt=""` anzugeben, denn dann ist es nicht notwendig, diese Information doppelt zur Verfügung zu stellen. Wird der Firmenname nur in Form eines Bildes präsentiert, schreibt man am einfachsten den Firmennamen in den `alt`-Text, oder auch „Firma XY Logo“. Nutzlos sind Texte, in denen der Name der Firma nicht vorkommt, z.B. „Firmenlogo“ oder „Logo im GIF-Format“, denn sie helfen einem Benutzer, der Bilder nicht sehen kann, nicht weiter.

Bilder als Links

Wenn Bilder als Links dienen, sollte auf jeden Fall ein `alt`-Text verwendet werden. Die meisten Screen Reader lesen alle Links entsprechend der Tab-Order nacheinander vor. Ist bei einer als Link verwendeten Graphik ein „leerer“ `alt`-Text angegeben, wird entsprechend nur angekündigt, dass es sich um einen Link handelt und sonst nichts vorgelesen. Kann oder möchte man dennoch einen „leeren“ `alt`-Text verwenden, sollte man solche Links an das Ende der Tab-Liste stellen, indem man den `tabindex="1000"` setzt, So werden die „namenlosen“ Links erst am Ende vorgelesen.

Der `alt`-Text von Bildern, die innerhalb von Links vorkommen, sollte nicht angeben, dass es sich um einen Link handelt, da das durch die entsprechenden Browser geschieht.

```
<a href="/index.html">
   <img src="home.gif" alt="Link zur Homepage">
   <img src="spacer.gif" alt="">
   zur Homepage
</a>
```

Besser:

```
<a href="/index.html"><img src="home.gif" alt="Homepage"></a>
<img src="spacer.gif" alt="">
<a href="/index.html">zur Homepage</a>
```

Aufzählungen

Werden Bilder als Aufzählungszeichen verwendet, gibt es ebenso mehrere Möglichkeiten. Ein schlechter Text für einen Button wäre „ein roter blinkender Punkt“. Der Text ist viel zu lang, und würde jeden Benutzer nerven, wenn die Liste aus mehr als drei Punkten besteht. Da der Alternative Text zur Gliederung dienen soll, bieten sich `alt`-Texte wie `*`, `-` oder an.

Bei der Verwendung von Screen Readern sind `alt`-Texte besser, die sich gut und schnell vorlesen lassen. `Punkt`, `bullet` oder `item` wären hierfür gut geeignet. Ist die Liste sehr lang, und ist klar ersichtlich, dass es sich um eine Liste handelt, kann man auch einen leeren `alt`-Text verwenden.

Werden besondere Aufzählungszeichen verwendet, um einzelne Punkte hervorzuheben, muss darauf geachtet werden, dass die Unterscheidung auch ohne die Bilder gut zu identifizieren ist. Abbildung 2.8 zeigt eine Liste mit entsprechenden `alt`-Attributen für die verschiedenen Bilder. Abbildung 2.9 und Abbildung 2.10 zeigen die Ausgabe in einem graphischen und einem textbasierenden Browser. Obwohl hier geeignete `alt`-Texte verwendet wurden, ist es mit einem Textbrowser nicht möglich herauszufinden, welche Hotels direkt gebucht werden können.

```
Die rot markierten Hotels können direkt gebucht werden:<br><br>

<img src="bullet.gif"    alt="-">  Hotel Am Rosenhügel  <br>
<img src="redbullet.gif" alt="*">  Hotel Eisenhauer     <br>
<img src="redbullet.gif" alt="*">  Hotel Herzog         <br>
<img src="bullet.gif"    alt="-">  Hotel Schmidt        <br>
```

Abbildung 2.8.: In dieser Liste werden korrekte alt-Texte eingesetzt. Doch die Erklärung ist ausschließlich auf die Graphiken abgestimmt.

Indem im Erklärungstext das gleiche Bild mit demselben `alt`-Text wie in der Liste verwendet wird, ist die Markierung, wie in Abbildung 2.11 und Abbildung 2.12 gezeigt, auch in einem text-basierenden Browser eindeutig.

Bilder mit Textinhalt

Häufig werden auf Webseiten einzelne Wörter oder Texte, die besonders gestaltet sein sollen, als Bilder dargestellt (Abbildung 2.13). Da die Gestaltung in der Regel keine zusätzliche Information enthält, reicht es, den auf dem Bild dargestellten Text als `alt`-Text zu verwenden.

Hotels

Die rot markierten Hotels können direkt gebucht werden:

- Hotel Am Rosenhügel
- Hotel Eisenhauer
- Hotel Herzog
- Hotel Schmidt

Abbildung 2.9.: In graphischen Browsern kann man anhand der Farben leicht bestimmen, welche Hotels direkt buchbar sind.

```
                                                    Beispiel 2.2 schlecht: hotels

                                    Hotels

   Die rot markierten Hotels können direkt gebucht werden:
   - Hotel Am Rosenhügel
   * Hotel Eisenhauer
   * Hotel Herzog
   - Hotel Schmidt

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 2.10.: In einem Textbrowser ist es auch mit den alt-Texten nicht möglich herauszufinden, auf welche der Markierungen der Erklärungstext Bezug nimmt.

```
Die mit einem <img src="redbullet.gif" alt="*"> markierten Hotels
können direkt gebucht werden:<br><br>

<img src="bullet.gif"    alt="-">  Hotel Am Rosenhügel  <br>
<img src="redbullet.gif" alt="*">  Hotel Eisenhauer     <br>
<img src="redbullet.gif" alt="*">  Hotel Herzog         <br>
<img src="bullet.gif"    alt="-">  Hotel Schmidt        <br>
```

Abbildung 2.11.: Werden sowohl in dem Erklärungstext als auch in der Liste alt-Texte verwendet, ist auch ohne die Graphiken die Struktur gut zu erkennen.

```
                                                    Beispiel 2.2 gut: Hotels

                                    Hotels

   Die mit einem * markierten Hotels können direkt gebucht werden:
   - Hotel Am Rosenhügel
   * Hotel Eisenhauer
   * Hotel Herzog
   - Hotel Schmidt

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 2.12.: Auch in Textbrowsern kann erkannt werden, auf welche Markierungen der Erklärungstext Bezug nimmt.

Abbildung 2.13.: Der Schriftzug „Willkommen“ graphisch ausgestaltet.

```
<img src="welcome.gif" alt="Welcome">
```

Dies gilt im Besonderen auch für animierte Texte oder animierte Graphiken. Bringt die Animation keine wesentliche Information, sollte man sie im `alt`-Attribut nicht erwähnen. In diesem Sinne ist bei dem in Abbildung 2.14 dargestellten animierten Gästebuch-Link der `alt`-Text „animierter Gästebuchlink“ nicht sinnvoll, „Gästebuch“, oder „zum Gästebuch“ gibt hier die gesamte Information der Graphik wieder.

Abbildung 2.14.: Ein animierter Gästebuch-Link, dessen Information durch den schlichten alt-Text „Gästebuch“ wiedergegeben kann.

```
<img src="viewGuestbook.gif" alt="Gästebuch">
oder
<img src="viewGuestbook.gif" alt="Gästebuch anzeigen">
```

Unverzichtbare Information in Bildern

In manchen Zusammenhängen beinhalten Bilder wichtige Informationen, ohne die der Inhalt der Seite unverständlich ist. Zum Beispiel bei mathematischen Funktionen, die als Graphiken dargestellt werden, ist es oft nicht möglich, diese Information mit äquivalenter Text-Information darzustellen. In diesem Fall sollte der Benutzer darauf hingewiesen werden, dass das Dokument ohne die Bilder wahrscheinlich nicht zu verstehen ist.

alt-Texte als Tooltips

Einige graphischen Browser stellen die `alt`-Texte als Tooltips dar. Das verleitet dazu, den `alt`-Text als zusätzliche Beschreibung für Bilder zu verwenden oder beispielsweise Copyright-Informationen anzuzeigen. Der `alt`-Text soll aber keine zusätzliche ergänzende Beschreibung sein, sondern eine alternative Darstellung für den Fall, dass die Bilder nicht angezeigt oder wahrgenommen werden können.

Um trotzdem zusätzliche Information in Form von Tooltipps zur Verfügung zu stellen, ohne das `alt`-Attribut „missbrauchen“ zu müssen, gibt es (ab HTML 4.0) das `title`-Attribut. Damit kann zusätzliche Information für ein Bild angegeben werden, die als Tooltip erscheint. Wird ein Leerzeichen als `title`-Text angegeben, wird bei den aktuellen Browsern kein Tooltip angezeigt, auch wenn ein `alt`-Text angegeben ist.

Nebeneinander liegende Bilder

Sind Bilder so auf einer Webseite angeordnet, dass sie direkt nebeneinander dargestellt werden, werden auch die `alt`-Texte nacheinander angezeigt. Dadurch kann es wie in Abbildung 2.15 gezeigt, leicht zu Missverständnissen kommen. Um dieses zu verhindern, ist es notwendig, die Texte so zu gestalten, dass sie deutlich voneinander getrennt sind.

```
<img src="button_back.jpg" alt="zurück">
<img src="button_homepage.jpg" alt="zur Homepage">
```

Abbildung 2.15.: Betrachtet man diese Bilder mit einem Textbrowser, erhält man scheinbar nur einen alt-Text: „Zurück zur Homepage“.

Um zu verhindern, dass in einem text-basierenden Browser „zurück zur Homepage“ angezeigt wird, fügt man beispielsweise ein „`|`“ in den `alt`-Text ein:

```
<img src="button_back.jpg" alt="zurück |">
<img src="button_homepage.jpg" alt="zur Homepage">
```

Eine andere Möglichkeit ist, die `alt`-Texte in Klammern zu setzen:

```
<img src="button_back.jpg" alt="[zurück]">
<img src="button_homepage.jpg" alt="[zur Homepage]">
```

3. Markup & Stylesheets I

Um Browsern die optimale Darstellung von Webseiten zu ermöglichen, müssen Webseiten den entsprechenden Internet-Standards angepasst sein. Das gilt im Besonderen auch für Browser, die auf die speziellen Bedürfnisse behinderter Menschen zugeschnitten sind, da die Benutzer solcher Browser besonders auf eine benutzerfreundliche Präsentation von Webseiten angewiesen sind.

Die Standards des World Wide Web entwickelt das World Wide Web Consortium (W3C) in Zusammenarbeit mit ca. 450 Organisationen aus aller Welt.

Die für Webdesign relevanten Standards und Guidelines des W3C sind unter anderem:

- **HTML/XHTML**, (Extensible) HyperText Markup Language[1]
  Die HyperText Markup Language ist die am häufigsten verwendete Sprache zur Formatierung von Inhalten von Webseiten. Ein Markup ist dabei eine Textauszeichnung, die die Formatierung eines Textabschnittes festlegt.
  XHTML ist der Nachfolger von HTML. Dabei handelt es sich um eine strenger definierte Form der HTML-Spezifikation. Jedes XHTML-Dokument ist ein XML-Dokument. XHTML kann als die Übergangssprache angesehen werden, die für einen reibungslosen Übergang von HTML nach XML sorgen soll.

- **CSS**, Cascading Style Sheets[2]
  Cascading Style Sheets dienen dazu, das Layout von Webseiten zu gestalten. Es handelt sich dabei um eine Ergänzungssprache zu HTML, um das Layout genauer zu kontrollieren. Durch Formatvorlagen, auf die mit Verweisen mehrfach zugegriffen werden kann, werden Inhalt und Darstellung von Webseiten getrennt.

- **DOM**, Document Object Model[3]
  Das Document Object Model ist eine Spezifikation, bei der die Struktur von dynamischen HTML und XML Dokumenten so beschrieben wird, dass sie von einem Web-Browser verändert werden kann. Innerhalb des DOM wird ein Dokument als hierarchische Struktur und nicht als Ansammlung von Wörtern aufgefasst.

[1] `http://www.w3.org/MarkUp/`
[2] `http://www.w3.org/Style/CSS/`
[3] `http://www.w3.org/DOM/`

- **XML**, Extensible Markup Language[4]
  Bei der eXtensible Markup Language handelt es sich um eine Sprache zur Beschreibung von Inhalten und Strukturen. „Extensible“ meint hier, dass die Sprache in einem spezifizierten Rahmen den eigenen Anforderungen entsprechend erweiterbar ist. Es handelt sich dabei um eine reine Datenbeschreibungssprache, das Layout kann mit Hilfe von XSL gesteuert werden.

- **XSL**, Extensible Stylesheet Language[5]
  XSL umfasst eine Reihe von Standards für das Definieren von XML Dokumentumwandlung und -darstellung. Mit XSL können beispielsweise XML Dokumente in XHTML Dokumente transformiert werden.

Gründe, warum Webseiten nicht barrierefrei, das heißt, für manche Benutzer unbrauchbar sind, sind, dass Tags und Attribute verwendet werden, die nicht im W3C-Standard definiert sind oder dass sie für Zwecke verwendet werden, für die sie nicht gedacht sind, dass technische Gegebenheiten wie Bildschirmmindestgrößen vorausgesetzt werden, oder dass Seiten auf bestimmte Browser abgestimmt sind.

Korrektes Verwenden von HTML-Tags und -Attributen sowie CSS-Technologien sind die Grundlage für gutes Webdesign. Um auf die speziellen Bedürfnisse behinderter Menschen Rücksicht zu nehmen, sollte man sich zusätzlich an den Guidelines der WAI für Accessible Web Design orientieren.

3.1. HTML 4

Vom Web-Gründer Tim Berners-Lee entwickelt, wurde HTML im Zuge des Web-Booms zum erfolgreichsten und verbreitetsten Dateiformat des Internets. HTML zeichnete sich von Beginn an vor allem durch seine Einfachheit in Struktur und Anwendung aus, die erste Version von HTML ist heute nicht mehr in Verwendung. Die Spezifikation zu dieser HTML-Version ist auf den Seiten des W3C auch gar nicht mehr verfügbar. HTML 2.0 wurde im November 1995 offizieller Sprachstandard. Diese Spezifikation für HTML ist beim W3C noch verfügbar, auch wenn bereits weitere Versionen entwickelt wurden. HTML 3.2 wurde nach langen Diskussionen am 14. Jänner 1997 offizieller Sprachstandard, ein Großteil dieser Neuerungen kann heute jedoch als problematisch angesehen werden.

Am 18. Dezember 1997 wurde die Spezifikation von HTML 4.0[6] veröffentlicht, und seit 24. Dezember 1999 gibt es die erste Revision HTML 4.01[7]. Mit der Einführung von

[4] `http://www.w3.org/XML/`
[5] `http://www.w3.org/Style/XSL/`
[6] `http://www.w3.org/TR/1998/REC-html40-19980424/`
[7] `http://www.w3.org/TR/html4/`

HTML 4 wurden nicht nur neue Tags und Attribute eingeführt, sondern auch eine Reihe von Elementen, die teilweise erst mit HTML 3.2 eingeführt wurden, als „deprecated“, also „missbilligt“ eingestuft. Sie sollen künftig wieder aus dem HTML-Sprachstandard entfallen, weil sie durch andere, ergänzende Technologien, wie CSS, realisierbar sind.

Um den Übergang von älteren HTML-Versionen auf die Verwendung von HTML 4 in Kombination mit Stylesheets zu erleichtern, gibt es derzeit HTML 4.01 in drei verschiedenen, sich in Auswahl der erlaubten Elemente unterscheidenden „flavours“. Auf jeder Seite sollte angegeben werden, welcher der folgenden HTML 4 Versionen sie entspricht.

- **HTML 4.01 Strict**
  „HTML 4.01 Strict“ entspricht gänzlich der neuen HTML 4.01 Spezifikation, die als „deprecated“ eingestuften Tags und Attribute sind somit nicht zu verwenden.

- **HTML 4.01 Transitional**
  „HTML 4.01 Transitional“ dient als Übergangslösung, und die Verwendung von „deprecated“ Elementen ist erlaubt.

- **HTML 4.01 Frameset**
  Bei „HTML 4.01 Frameset“ handelt es sich um eine eigene Zusammenstellung von Elementen zur Spezifikation von `<frameset>`-Seiten.

Mit einer DOCTYPE-Definition ist es möglich anzugeben, welcher „flavour“ von einer Seite implementiert wird. Diese Anweisung wird in die erste Zeile einer HTML-Seite eingefügt. Je nach Version umfasst diese Anweisung einen der folgenden Blöcke:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN"
                      "http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                      "http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
                      "http://www.w3.org/TR/html4/frameset.dtd">
```

Dadurch wird es auch ermöglicht, die Seiten mit einem entsprechenden Programm auf die Erfüllung der jeweiligen HTML 4 Version zu überprüfen. Ein Service dafür, das von dem W3C zur Verfügung gestellt wird, findet sich unter: `http://validator.w3.org/`.

Um den HTML 4.01 Standard zu erfüllen, muss ein Webseite zumindest die in Abbildung 3.1 gezeigte Grundstruktur aufweisen.

Wie in Abbildung 3.1 dargestellt, besteht ein gültiges HTML 4.01 Dokument zumindest aus den folgenden Teilen:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<html>
   <head>
      <title>HTML 4.01 Strict Template</title>
      <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
   </head>
   <body>
      <p>Ein HTML 4.01 Strict Template.</p>
   </body>
</html>
```

Abbildung 3.1.: Minimale HTML 4 Grundstruktur

- Aus einer DOCTYPE-Definition, die die genaue HTML 4 Version angibt,
- der `<html>`-, `<head>`-, `<body>`-Struktur,
- einem `<title>` und
- der Spezifikation des zugrunde liegenden Zeichensatzes z.B. mit Hilfe des `<meta>`-Tags.

3.2. Structural vs. Presentational Markup

Generell kann zwischen zwei Arten von HTML-Tags unterscheiden werden. Die **structural** oder **logischen** Tags repräsentieren die innere Struktur eines Dokumentes. Die graphische Umsetzung dieser Tags ist nicht genau festgelegt und kann von unterschiedlichen Browsern und Plattformen individuell gestaltet werden.

Mit den **presentational** oder **physischen** Tags wird ein Text formatiert, ohne ihm durch das Markup eine besondere inhaltliche Bedeutung zuzuteilen.

Zum Hervorheben von Text gibt es beispielsweise die Tags `<em>` und `<strong>`. Damit kann die besondere Bedeutung eines Wortes oder eines Textes gekennzeichnet werden. Graphische Browser stellen dies kursiv beziehungsweise fett dar. Soll ein Text hingegen aus rein ästhetischen Gründen fett oder kursiv dargestellt werden, verwendet man stattdessen die Tags `<b>` und `<i>`.

Der richtige Einsatz von structural und presentational Markup ermöglicht gerade auch Voice- und Textbrowsern, den Inhalt vernünftig darzustellen. Ein Voicebrowser könnte einen Text innerhalb von `<strong>`-Tags zur Hervorhebung mit lauter Stimme vorlesen,

einen Text innerhalb von `<b>`-Tags, die nur zur graphischen Gestaltung fett formatiert sind, hingegen nicht.

Da die presentational Tags von vielen Designern zur Strukturierung von Seiten verwendet werden, ist die Verwendung von structural Markups der von presentational Markups vorzuziehen. Die gewünschte graphische Gestaltung dieser Tags kann dann mit Hilfe von Stylesheets erfolgen. Durch das Weglassen von Stylesheets kann man leicht kontrollieren, ob die Struktur der Webseite immer noch gut erkenntlich ist.

3.3. Korrektes Verwenden von Markup

Unter Markup versteht man alle Tags und Attribute, also die gesamte Information in Dokumenten, die sich auf deren Struktur und Layout beziehen. Besonders um Webseiten barrierefrei zu gestalten, ist die spezifikationsgemäße Verwendung von Markup vonnöten. Im Folgenden soll auf die zentralen Themengebiete der Verwendung von Markup eingegangen werden.

Überschriften

Die Struktur von Texten wird maßgeblich durch Überschriften widergespiegelt. Daher ist es unerlässlich, Überschriften mit den entsprechenden Markups (`<h1>` bis `<h6>`) zu kennzeichnen. Die Texthierarchie wird durch die unterschiedlichen Ebenen 1 bis 6 mit allen Browsern klar ersichtlich. Besonders Screenreadern und Textbrowsern kommt die Verwendung dieser Tags zugute, da die Überschriften entsprechend dargestellt und vorgelesen werden können. Bei einer graphischen Gestaltung mittels presentational Tags wäre dies nicht möglich. In allen Browsern gibt es eine standardmäßige Darstellung von Überschriften, die aber mit Stylesheets leicht umgestaltet werden kann.

Auch wenn an Stelle eines Textes ein Bild als Überschrift verwendet wird, sollte diese s auf jeden Fall auch als solche markiert werden. Dies ermöglicht Text- und Voicebrowsern, den `alt`-Text als Überschrift darzustellen.

```
<h1><img src="Kapitel1.gif" alt="Kapitel 1"></h1>
```

Diese Tags sollten hingegen nicht dazu verwendet werden, Aufmerksamkeit auf einen Textteil zu lenken, wenn dieser nicht zur Strukturierung dient.

```
<center>
   <h1>Sonderangebot!</h1>
   <h2><a href="sonderangebot.html">Hier klicken!</a></h2>
</center>
```

In diesem Fall ist es „sauberer“, den Text mit presentational Markup auffällig zu präsentieren.

```
<div style="font-size:200%; text-align:center;">Sonderangebot!</div>
<div style="font-size:150%; text-align:center;">
   <a href="sonderangebot.html">Hier klicken!</a>
</div>
```

Noch flexibler ist das Verwenden von Klassen, um bei wiederholtem Vorkommen solcher Textteile einen einheitlichen, leicht veränderbaren Stil zu erhalten (Abbildung 3.2).

```
<head>
   <style>
      .sonderGross   { font-size:200%; text-align:center; }
      .sonderKlein   { font-size:150%; text-align:center; }
   </style>
</head>
<body>
   ...
   <div class="sonderGross">Sonderangebot!</div>
   <div class="sonderKlein">
      <a href="sonderangebot.html">Hier klicken!</a>
   </div>
   ...
</body>
```

Abbildung 3.2.: Die Verwendung von Klassen zur Hervorhebung von Textteilen.

Absätze

Einzelne Absätze sollten, wie in Abbildung 3.3 gezeigt, jeweils in `<p>`-Tags eingeschlossen werden, anstatt das gewünschte Layout mit Zeilenumbrüchen oder transparenten Bildern zu erreichen. In graphischen Browsern ist kein Unterschied sichtbar, doch Textbrowser ignorieren aufeinander folgende Zeilenumbrüche und transparente Bilder, deren `alt`-Text leer sein sollte. Somit ist nicht nachzuvollziehen, in welche Absätze ein Text gegliedert ist.

Werden Absätze durch `<p>`-Tags markiert, können auch sie von allen Browsern entsprechend dargestellt werden. Soll ein Absatz besonders formatiert werden, kann sein Layout über Stylesheets definiert werden.

In Abbildung 3.4 ist ersichtlich, wie sich korrektes Verwenden von `<p>`-Tags und Darstellen von Absätzen durch mehrere Zeilenumbrüche in Textbrowsern unterscheiden.

```
<p>Erster korrekt behandelter Absatz. Korrekte Absätze stehen
zwischen &lt;p>-Tags, damit die Absätze auf allen Browser vernünftig
dargestellt werden.</p>

<p>Zweiter korrekt behandelter Absatz. Auch dieser Absatz steht
innerhalb eines &lt;p>-Tags.</p>

<p>So sollte man keine Absätze schreiben, denn dieser Absatz steht
nicht innerhalb eines &lt;p>-Tags.<br><br>

Dieser zweite falsche Absatz verwendet wie der obige &lt;br>-Tags um
Absätze zu gestalten!<br><br></p>
```

Abbildung 3.3.: Verwendung von Paragraphen.

```
                                                    Beispiel 3.1: Absätze

   Erster  korrekt  behandelter  Absatz. Korrekte Absätze stehen zwischen
   <p>-Tags,  damit  die Absätze auf allen Browser vernünftig dargestellt
   werden.

   Zweiter korrekt behandelter Absatz. Auch dieser Absatz steht innerhalb
   eines <p>-Tags.

   So  sollte man keine Absätze schreiben, denn dieser Absatz steht nicht
   innerhalb eines <p>-Tags.
   Dieser  zweite  falsche  Absatz  verwendet  wie der obige <br>-Tags um
   Absätze zu gestalten!

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 3.4.: Darstellung von Absätzen in Textbrowsern.

Listen

Zur Erstellung von Listen existieren zwei unterschiedliche Formatierungsmöglichkeiten. Nummerierte Listen werden mit dem `<ol>`-Tag gekennzeichnet, unnummerierte Listen mit dem `<ul>`-Tag. Die einzelnen Elemente der Liste werden jeweils mit einem `<li>`-Tag gekennzeichnet. Besonders bei unnummerierten Listen werden häufig Graphiken als Aufzählungszeichen verwendet, und der Text diesen angefügt. Das hat zum Effekt, dass die Liste, wenn keine Bilder dargestellt werden können, nicht als solche erkennbar ist, da die logische Gliederung fehlt.

Wie in Abbildung 3.5 ersichtlich, ist es mit Hilfe von Stylesheets möglich, den ordnungsgemäß gestalteten Listen Bilder als Aufzählungszeichen zuzuweisen.

```
<head>
   <style>
      *  {   font-family: verdana }
      li {   list-style-image: url("../images/redbullet.gif"); }
   </style>
</head>
<body>
   <p>Gestaltung einer Liste ohne die dafür vorgesehenen
   Formatierungstags:</p>
   <div style="margin-left:20px">
      <img src="../images/redbullet.gif" alt="*">Erstes Element<br>
      <img src="../images/redbullet.gif" alt="*">Zweites Element<br>
      <img src="../images/redbullet.gif" alt="*">Drittes Element<br>
      <img src="../images/redbullet.gif" alt="*">Viertes Element<br>
   </div>
   <p>Eine entsprechend der Spezifikation erstellte Liste:</p>
   <ul>
      <li>Erstes Element
      <li>Zweites Element
      <li>Drittes Element
      <li>Viertes Element
   </ul>
</body>
```

Abbildung 3.5.: Unterschiedliche Gestaltung von Listen.

Obwohl nicht alle graphischen Browser die korrekte Darstellung von `<li>`-Elementen mit Bildern unterstützen, ist es für barrierefreies Webdesign von Vorteil, Listen spezifikationsgemäß zu gestalten.

Gestaltung einer Liste ohne die dafür vorgesehenen Formatierungstags:

- Erstes Element
- Zweites Element
- Drittes Element
- Viertes Element

Eine entsprechend der Spezifikation erstellte Liste:

- Erstes Element
- Zweites Element
- Drittes Element
- Viertes Element

Abbildung 3.6.: In graphischen Browsern unterscheidet sich die Darstellung verschieden gestalteter Listen nicht.

Werden dennoch Bilder als Aufzählungszeichen verwendet, sollte unbedingt das `alt`-Attribut gesetzt werden, wie dies im obigen Beispiel getan wurde. Darauf wird im Kapitel 2 „Äquivalente Text-Information“ näher eingegangen.

Zitate

Auch um Zitate auf Webseiten entsprechend darzustellen, existieren zwei Formatierungen (Abbildung 3.7). `<q>` dient als Zitat innerhalb eines Absatzes und `<blockquote>` formatiert das Zitat als eigenen, eingerückten Absatz.

Wie das Ergebnis in einem graphischen Browser aussieht, zeigt Abbildung 3.8.

Von Textbrowsern (Abbildung 3.9) wird das Zitat innerhalb eines Absatzes zum Beispiel mit doppelten Anführungszeichen dargestellt.

Um den logischen Aufbau einer Seite zu wahren, sollte das `<blockquote>`-Tag keinesfalls dafür verwendet werden, um einen beliebigen Absatz einzurücken, sondern nur um Zitate zu kennzeichnen. Zum Einrücken von Absätzen sollten Formatierungen mit Hilfe von Stylesheets erstellt werden.

```
<head>
   <style>
      q   { font-style:italic; }
   </style>
</head>
<body>
   <p>Das Hauptanliegen der Web Content Accessibility Guidelines
   ist:</p>

   <blockquote cite="http://www.w3.org/TR/WCAG10/">
      The guidelines address two general themes: ensuring
      graceful transformation, and making content understandable
      and navigable.
   </blockquote>

   <p><q cite="http://www.w3.org/TR/WCAG10/">Ensuring
      graceful transformation</q> bedeutet zum Beispiel, dass
      eine Webseite auch noch mit einer geringen Auflösung gut
      lesbar ist.
   </p>

</body>
```

Abbildung 3.7.: Verwendung von q- und blockquote-Tags.

Das Hauptanliegen der Web Content Accessibility Guidelines ist:

> The guidelines address two general themes: ensuring graceful transformation, and making content understandable and navigable.

Ensuring graceful transformation bedeutet zum Beispiel, dass eine Webseite auch noch mit einer geringen Auflösung gut lesbar ist.

Abbildung 3.8.: Darstellung von Zitaten in einem graphischen Browser.

```
                                                    Beispiel 3.3: Zitate

   Das Hauptanliegen der Web Content Accessibility Guidelines ist:

     The  guidelines  address  two  general  themes:  ensuring  graceful
     transformation, and making content understandable and navigable.

   "Ensuring  graceful  transformation"  bedeutet zum Beispiel, dass eine
   Webseite auch noch mit einer geringen Auflösung gut lesbar ist.

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 3.9.: Darstellung von Zitaten in einem Textbrowser.

4. Markup & Stylesheets II

4.1. Relative und absolute Größen

Es gibt verschiedene Möglichkeiten, in HTML Größen anzugeben. Man unterscheidet zwischen absoluten und relativen Einheiten. Für barrierefreies Webdesign ist es meistens empfehlenswert, relative Angaben zu verwenden.

In HTML gibt es nur eine relative und eine absolute Größe, Prozent und Pixel, für Cascading Stylesheets sind die folgenden absoluten Einheiten spezifiziert. Die Längenangaben beziehen sich auf den Ausdruck der Seite.

- **cm**: Zentimeter.
- **mm**: Millimeter.
- **in**: Inch (= 2.54 cm).
- **pt**: Punkte, in CSS definiert als 1/72 in.
- **pc**: Pica (= 12 pt).

Relative Größen beziehen sich immer auf eine Referenzgröße. Dies ist beispielsweise die Höhe der aktuellen Schriftart, oder die Seitenbreite. Die folgenden relativen Einheiten sind in CSS definiert:

- **em**: Einheit im Bezug auf die Ziffernbreite.
- **ex**: Einheit im Bezug auf die x-Höhe.
- **px**: Pixel.
- **%**: Prozent.

Die x-Höhe ist die Höhe des kleinen Buchstabens „x“. Diese Größe ist auch für Schriftarten definiert, die kein „x“ enthalten.

Die Angabe in Pixel bezieht sich auf die Auflösung des darstellenden Gerätes, in der Regel eines Monitors oder eines Druckers. Bei Bildschirmen entspricht `1px` normalerweise genau einem Bildpunkt. Bei Druckern wird `1px` durch eine, von der Auflösung des Druckers abhängige Anzahl an Punkten dargestellt.

In den meisten Fällen sollten die relativen Einheiten `em`, `ex` und `%` verwendet werden, damit die Seiten besser skalieren. Das bedeutet, dass die Seite bei veränderter Schrift- oder Bildschirmgröße so an den vorhanden Platz angepasst werden kann, dass sie diesen möglichst gut ausnützt. In manchen Fällen ist es sinnvoller, absolute Größen zu verwenden, beispielsweise wenn es um die genaue Platzierung von Graphiken geht. Vor allem bei der Angabe von Schriftgrößen sollten relative Einheiten zu Gunsten der Skalierbarkeit eingesetzt werden.

Im Beispiel in Abbildung 4.1 werden Abstände einmal mit absoluten und einmal mit relativen Einheiten festgelegt. Im ersten Fall sind alle Maße absolut (`pt` und `pc`), im zweiten Fall ist die Schriftgröße relativ und im dritten sind alle Einheiten mit `em` angegeben.

Abbildung 4.2 zeigt, dass in einem graphischen Browser mit Standardsettings alle drei Tabellen gleich aussehen.

Bei einigen graphischen Browsern ist es möglich, die Bezugs-Schriftgröße zu ändern. Dadurch kann die Schriftgröße individuell eingestellt werden. Dies kann für Personen mit Sehbehinderungen oder für ältere Menschen die Lesbarkeit erhöhen. Vergrößert werden allerdings nur solche Elemente, die mit relativen Einheiten angegeben wurden (Abbildung 4.3).

- In Abbildung 4.3 ist bei der ersten Tabelle keine Größenänderung im Vergleich zur Abbildung 4.2 zu bemerken. Hier wurden nur absolute Größenangaben verwendet.
- In der zweiten Tabelle vergrößert sich zwar die Schrift, die Balken bleiben aber auf Grund der absoluten Längenangabe gleich lang.
- Die letzte Tabelle skaliert in alle Richtungen, denn die verwendeten Einheiten sind relativ.

4.2. Bildschirmauflösungen

Auch um unterschiedliche Bildschirmauflösungen berücksichtigen zu können, sollten relative Größenangaben verwendet werden. Immer öfter wird beim Design von Webseiten davon ausgegangen, dass sie mit einer Auflösung von 1024x768 Pixel angezeigt werden. In der Praxis gibt es aber noch viele Monitore, deren höchste Auflösung 800x600 oder 640x480 ist. Man kann auch nicht davon ausgehen, dass alle Benutzer mit 1024x768 Bildschirmen die Webseiten in einem maximal großen Fenster anschauen.

```
<table style="font-size:12pt; text-align:left">
   <tr><th>Programm</th><th>Laufzeit</th><th>Balken</th></tr>
   <tr><td>Bubble sort</td><td>6,8 sec</td>
       <td><div style="width:6.8pc"> </div></td></tr>
   <tr><td>Selection sort</td><td>4,0 sec</td>
       <td><div style="width:4.0pc"> </div></td></tr>
   <tr><td>Insertion sort</td><td>1,6 sec</td>
       <td><div style="width:1.6pc"> </div></td></tr>
</table>

<table style="font-size:1em; text-align:left">
   <tr><th>Programm</th><th>Laufzeit</th><th>Balken</th></tr>
   <tr><td>Bubble sort</td><td>6,8 sec</td>
       <td><div style="width:6.8pc"> </div></td></tr>
   <tr><td>Selection sort</td><td>4,0 sec</td>
       <td><div style="width:4.0pc"> </div></td></tr>
   <tr><td>Insertion sort</td><td>1,6 sec</td>
       <td><div style="width:1.6pc"> </div></td></tr>
</table>

<table style="font-size:1em; text-align:left">
   <tr><th>Programm</th><th>Laufzeit</th><th>Balken</th></tr>
   <tr><td>Bubble sort</td><td>6,8 sec</td>
       <td><div style="width:6.8em"> </div></td></tr>
   <tr><td>Selection sort</td><td>4,0 sec</td>
       <td><div style="width:4.0em"> </div></td></tr>
   <tr><td>Insertion sort</td><td>1,6 sec</td>
       <td><div style="width:1.6em"> </div></td></tr>
</table>
```

Abbildung 4.1.: Verwendung von absoluten und relativen Längenangaben.

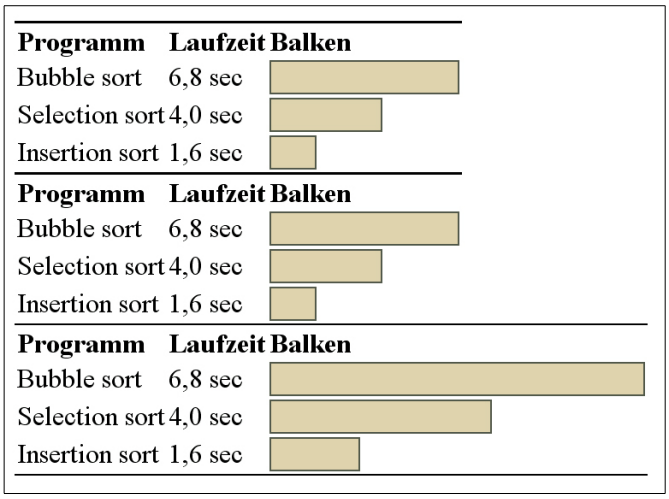


Abbildung 4.2.: Verschiedene Größenangaben in einem graphischen Browser mit Standardsettings.

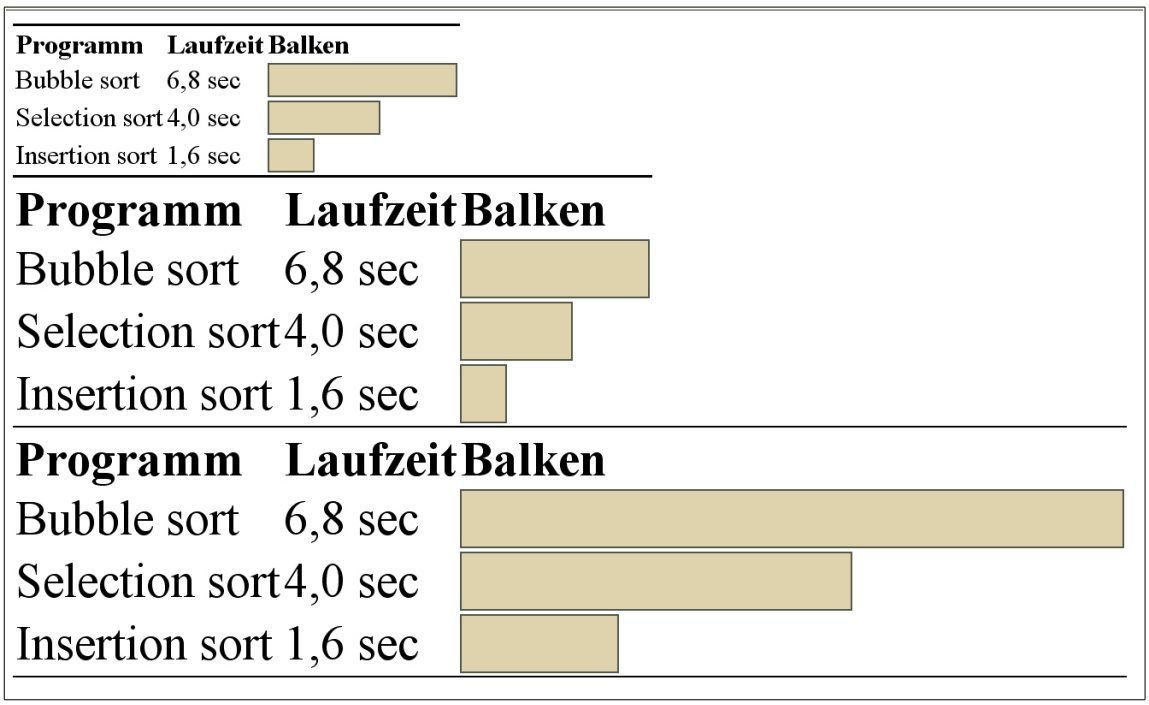


Abbildung 4.3.: Sind große Schriftarten ausgewählt, werden in graphischen Browsern Elemente mit relativen Größenangaben vergrößert dargestellt.

Ein weiteres Problem bei der Verwendung fixer Seitengrößen ist, dass Seiten, die eine gewisse Breite übersteigen, nicht komplett auf A4-Papier gedruckt werden können, da sie auch von Druckern nicht schmäler dargestellt werden können.

Im Hinblick auf mobiles Surfen mit PDAs, das zunehmend an Bedeutung gewinnt, ist es wichtig, dass Seiten auch mit kleinen Bildschirmen dargestellt werden können. Zur Verdeutlichung, mit welchen Dimensionen man zu rechnen hat, seien hier einige typische Auflösungen angeführt:

Gerät	**Auflösung**
Palm Tungsten T	320x320 px
Compaq iPAQ	240x320 px
Palm V	160x160 px
HP Jornada 620	640x240 px

Benutzer von Braille-Zeilen stehen unter der Einschränkung, dass diese maximal 80 Zeichen pro Zeile anzeigen können. Bei portablen Geräten können sogar nur 20 bis 24 Zeichen pro Zeile dargestellt werden. Sind Textzeilen breiter, muss daher nicht nur vertikal sondern auch horizontal gescrollt werden, um die ganze Seite lesen zu können.

Auch bei graphischen Browsern, die in Kombination mit Monitoren mit geringer Auflösung verwendet werden, ist es häufig nötig zu scrollen. Deshalb ist es nicht ratsam, Scrollbars auf Webseiten zu deaktivieren, da Information dadurch nicht erreichbar sein könnte.

Aus diesen Gründen ist es wichtig zu testen, ob sich Webseiten auch mit kleineren Auflösungen noch gut darstellen lassen.

4.3. Das Testen von Webseiten

Es gibt eine Reihe von einfachen Möglichkeiten um zu überprüfen, ob HTML-Markups und Stylesheets spezifikationsgemäß verwendet wurden. Ein Großteil der Fehler kann dadurch entdeckt und ausgebessert werden.

Überprüfen auf HTML-Validität

Wie in Kapitel 3 „Markup & Stylesheets I“ gezeigt, muss eine Webseite dem HTML 4 Standard entsprechen, um zu gewährleisten, dass sie von allen aktuellen Browsern vernünftig dargestellt werden kann. Um zu überprüfen, ob eine Webseite diesem Standard entspricht, werden vom W3C folgende Services zur Verfügung gestellt, mit denen Seiten online getestet werden können:

- **W3C HTML Validation Service**: `http://validator.w3.org/`
- **W3C CSS Validatior**: `http://jigsaw.w3.org/css-validator/`

Testen ohne Stylesheets

Wurde eine Seite mit Hilfe von externen Stylesheets gestaltet, ist es empfehlenswert, sich die Seite auch ohne Stylesheets anzusehen. So lässt sich sehr gut erkennen, ob die Struktur klar verständlich ist. Ist der Aufbau der Seite dann unklar, wurde das Markup nicht ordnungsgemäß verwendet und sollte überarbeitet werden.

Auch inhaltliche Missverständnisse lassen sich vermeiden, wenn darauf geachtet wird, das structural und das presentational Markup kontextgemäß zu verwenden. Die Webseite in Abbildung 4.4 lässt sich ohne Stylesheets nicht verstehen, auch wenn die Darstellung mit Stylesheets in einem graphischen Browser gut verständlich ist (Abbildung 4.5).

```
<head>
   <link rel="stylesheet" href="sample-04-02.css">
</head>
<body>
   <p>Im folgenden Satz sind alle Substantive markiert:</p>

   <p><span class="substantiv">Franz</span> jagt im völlig
   verwahrlosten <span class="substantiv">Taxi</span> quer durch
   <span class="substantiv">Bayern</span>.</p>
</body>
```

Abbildung 4.4.: Unsachgemäße Verwendung von Stylesheets.

Im folgenden Satz sind alle Substantive markiert:

Franz jagt im völlig verwahrlosten **Taxi** quer durch **Bayern**.

Abbildung 4.5.: Darstellung mit aktivierten Stylesheets. Die markierten Wörter sind rot und fett dargestellt.

Sind bei der Webseite aus Abbildung 4.4 die Stylesheets deaktiviert, wird das zur Hervorhebung verwendete presentational Markup ignoriert. Dadurch geht die Aussage des Textes verloren (Abbildung 4.6).

Im folgenden Satz sind alle Substantive markiert:

Franz jagt im völlig verwahrlosten Taxi quer durch Bayern.

Abbildung 4.6.: Darstellung mit deaktivierten Stylesheets. Die markierten Wörter sind nicht mehr hervorgehoben.

Anstatt das `<span>`-Tag zu verwenden, das bei deaktivierten Stylesheets keinen sichtbaren Unterschied hervorruft, wird in Abbildung 4.7 das `<strong>`-Tag verwendet. In diesem Fall wird bei aktivierten Stylesheets genau das gleiche Ergebnis angezeigt, während bei deaktivierten Stylesheets auf die vorhandene Funktionalität des `<strong>`-Tags zurückgegriffen wird (Abbildung 4.8).

```
<head>
   <link rel="stylesheet" href="sample-04-02.css">
</head>
<body>
   <p>Im folgenden Satz sind alle Substantive markiert:</p>

   <p><strong class="substantiv">Franz</strong> jagt im völlig
   verwahrlosten <strong class="substantiv">Taxi</strong> quer durch
   <strong class="substantiv">Bayern</strong>.</p>
</body>
```

Abbildung 4.7.: Verwendung von Stylesheets und dem strong-Tag.

Im folgenden Satz sind alle Substantive markiert:

Franz jagt im völlig verwahrlosten **Taxi** quer durch **Bayern**.

Abbildung 4.8.: Darstellung mit deaktivierten Stylesheets unter Verwendung von strong-Tags. Die markierten Wörter sind nun auch ohne Stylesheet hervorgehoben.

Testen mit einem Textbrowser

Das Testen von Webseiten mit Lynx oder anderen textbasierenden Webbrowsern ist eine weitere gute Möglichkeit, Barrieren in Webseiten zu finden.

Dies trägt nicht nur dazu bei, Bilder ohne entsprechendes `alt`-Attribut zu finden, sondern zeigt unter anderem auf, wie sich eine Webseite in linearisierter Form zeigt. In einem Textbrowser können nicht alle Tabellenzellen, die nacheinander dargestellt werden sollen, auch wirklich so dargestellt werden. Ansonsten könnte es passieren, dass aus einer Tabellenstruktur jeweils die erste Zeile jeder Zelle nacheinander angezeigt wird, und dann erst die zweite Zeile der Zellen. Dadurch kommt es zu Verständnisschwierigkeiten, da unabhängige Textteile miteinander vermischt werden, wie zum Beispiel die Navigation mit dem Inhalt einer Seite.

Im folgenden Kapitel wird auf Tabellen- und Navigationsstrukturen näher eingegangen.

Anstatt einen Textbrowser zu installieren, kann auch der Lynx Viewer von Delorie verwendet werden. Dieses Online-Service emuliert den Lynx-Browser. Auch wenn es nicht der Verwendung eines Textbrowsers gleichkommt, vermittelt es doch einen guten Eindruck, auf welche Art viele behinderte Personen den Inhalt des Internets wahrnehmen. Will man sich damit auseinandersetzen, wie blinde Personen das Internet erleben, kann man Seiten auch mit einem Voicebrowser testen, auch wenn diese Form des Surfens sehr ungewohnt und auch anstrengend ist. Dafür kann beispielsweise der IBM Home Page Reader verwendet werden.

- **Lynx**: `http://lynx.browser.org`
- **Lynx Viewer von Delorie**: `http://www.delorie.com/web/lynxview.html`
- **IBM Home Page Reader**: `http://www.ibm.com/de/accessibility/hpr.html`

Browser

Leider werden nicht von allen graphischen Browsern, die derzeit auf dem Markt sind, Webseiten gleich angezeigt. Dies liegt unter anderem daran, dass kein Browser alle HTML 4 Tags und Attribute unterstützt, und dass auch gängige Tags und Attribute unterschiedlich interpretiert werden. Es gibt auch Browser, die eigens definiertes Markup unterstützen, das sonst von keinem anderen Browser verarbeitet werden kann.

Daher ist es ratsam, Webseiten mit möglichst vielen Browsern zu testen. Im Besonderen gilt dies für Webseiten, die nicht HTML 4 konform sind, sondern Browser-spezifische Tags und Attribute verwenden. In diesem Fall muss darauf geachtet werden, dass die Grundfunktionalität, wie zum Beispiel die Navigation, auch bei anderen Browsern erhalten bleibt.

Die Rückwärtskompatibilität von Webseiten kann gut mit dem Web Page Backward Compatibility Viewer von Delorie getestet werden. Mit diesem Service kann ein Browser „simuliert“ werden, indem man festlegt, welche Features unterstützt werden. Damit ist

es möglich, sich das Ergebnis einer Seite ohne Javascript, Bilder und Hintergrundfarben anzuschauen.

- **Web Page Backward Compatibility Viewer** von Delorie:
  `http://www.delorie.com/web/wpbcv.html`

5. Layout & Navigation

Eine einheitliche und gut durchdachte Navigation ist für jeden Webauftritt unabdingbar. Sie erleichtert es dem Benutzer, sich auf den Internetseiten zurechtzufinden. Sie sollte die Fragen „wo bin ich?“ und „wie komme ich wo hin?“ beantworten. Je größer eine Webseite ist, umso mehr muss man sich mit der Gestaltung der Navigation auseinandersetzen.

Eine gut strukturierte Navigation ist im Besonderen für blinde Personen und Personen mit kognitiven Einschränkungen wichtig. Blinde Personen können nicht wie sehende mit wenigen Blicken eine Seite „scannen“, und somit die Information in Blöcke wie Navigation, Haupttext, Werbung und Footer unterteilen. Das zeilenweise Lesen einer Webseite stellt demnach besondere Ansprüche an eine Navigation.

Für Personen mit kognitiven Einschränkungen hingegen ist es besonders wichtig, Linktexte gut zu wählen, wichtige Information an den Anfang von Absätzen zu stellen und Möglichkeiten zu schaffen, die gesuchte Information auf Webseiten rasch zu finden.

Ein weiterer Aspekt, über den man sich beim Design Gedanken machen muss, ist die Auswahl von Farben und Schriftarten.

5.1. Grundlagen der Navigation

Eine typische Webseite hat oft den in Abbildung 5.1 und Abbildung 5.2 gezeigten Aufbau.

Die Hauptinformation ist umgeben von Logos, Navigationen, Überschriften, Wertungen, etc. Für die meisten Benutzer ist es relativ einfach, sich auf dieser Seite zu orientieren. Das Betrachten der Seite mit einem Textbrowser (Abbildung 5.3) gibt einen ersten Einblick in die Barrieren, die sich bei so einer Anordnung ergeben können.

In Abbildung 5.3 ist zu sehen, dass die äußere Tabelle von Lynx linearisiert wurde. Das bedeutet, dass alle Zellen untereinander dargestellt werden. Dadurch rutscht der Hauptteil weiter nach unten. Da nun die Hauptnavigation am Anfang der Seite erscheint, muss bei jeder Webseite zuerst die Navigation (vor-) gelesen werden, bevor man zum Haupttext gelangt, auch wenn diese auf jeder Seite gleich ist. Dies ist besonders ärgerlich, wenn der Header und die Hauptnavigation sehr umfangreich sind, wie dies oft bei Portalen oder News-Seiten der Fall ist.

Logo	Firmenname	
Hauptnavigation • Navigation 1 • Navigation 2 • Navigation 3 • Navigation 4 • Navigation 5	**Überschrift** Hier befindet sich der Haupttext. Er ist umgeben von Firmeninformation und Navigationsmenüs. Für sehende Personen ist es kein Problem den Haupttext zu identifizieren, unter anderem da sie solche Anordnungen bereits gewohnt sind. Auch im Haupttext können Tabellen verwendet werden: **Spalte 1** **Spalte 2** **Spalte 3** Wert 1 Wert 2 Wert 3 Wert über 2 Spalten Wert 4	**Subnavigation** • Navigation A • Navigation B • Navigation C • Navigation D • Navigation E
© by Firma		

Abbildung 5.1.: Typischer tabellarischer Aufbau einer Seite mit Navigation.

Um dieses Problem zu lösen, gibt es im Wesentlichen zwei Möglichkeiten. Beide Lösungsvorschläge lassen sich auch kombinieren.

5.2. Verbesserte Tabellenstruktur

Das Problem an herkömmlichen Tabellenstrukturen ist, dass die Navigation links vom Hauptteil, und damit beim Linearisieren vor dem Hauptteil angeordnet ist. Eine Möglichkeit, den Haupttext „nach vorne zu bringen“, ist in Abbildung 5.4 abgebildet. Durch das Einfügen leerer Zellen vor den Navigationszellen wird der Haupttext nun vor der Hauptnavigation angeordnet und erscheint nach dem Linearisieren auch vor dieser. Um das ursprüngliche Aussehen beizubehalten, kann die Höhe der Zellen auf ein Pixel reduziert werden.

5.3. Skip Links

Ein Problem an umfangreichen Seiten ist, dass um den eigentlichen Haupttext eine Menge zusätzlicher Information platziert ist. Dies kann dazu führen, dass ein Screen-Reader einige Minuten vorliest, ehe er die Hauptinformation erreicht.

Für Personen, die sich nicht auf „konventionelle Weise“ einen Überblick über eine Seite verschaffen können, sollte man Orientierungshilfen anbieten. Dies kann man beispielsweise dadurch erreichen, dass am Anfang einer Seite Links platziert werden, die zu den einzelnen Teilen einer Seite führen.

```
<table cellspacing="0" cellpadding="4">
   <tr>
      <td class="top">Logo</td>
      <td class="top">Firmenname</td>
      <td class="top"></td>
   </tr>
   <tr>
      <td valign="top" width="100" class="mainnavi">
         <b>Hauptnavigation</b>
         <ul>
            <li>Navigation 1</li>
            <li>Navigation 2</li>
            <li>Navigation 3</li>
            <li>Navigation 4</li>
            <li>Navigation 5</li>
         </ul>
      </td>
      <td valign="top">
         <h1>Überschrift</h1>

         <p>Hier befindet ...</p>

         <p>Auch im Haupttext können Tabellen verwendet werden:</p>

         <table border="0" width="250">
            <tr align="left"><th>Spalte 1</th><th>Spalte 2</th>
                              <th>Spalte 3</th></tr>
            <tr><td>Wert 1</td><td>Wert 2</td><td>Wert 3</td></tr>
            <tr><td colspan="2">Wert über 2 Spalten</td><td>Wert 4
         </table>
      </td>
      <td valign="top" width="100" class="subnavi">
         <b>Subnavigation</b>
         <ul>
            <li>Navigation A</li>
            <li>Navigation B</li>
            <li>Navigation C</li>
            <li>Navigation D</li>
            <li>Navigation E</li>
         </ul>
      </td>
   </tr>
   <tr>
      <td class="footer" colspan="3">&copy; by Firma</td>
   </tr>
</table>
```

Abbildung 5.2.: Sourcecode einer Seite mit tabellarischer Navigation.

```
                                                    Typischer Webseitenaufbau

   Logo Firmenname
   Hauptnavigation
     * Navigation 1
     * Navigation 2
     * Navigation 3
     * Navigation 4
     * Navigation 5

Überschrift

   Hier befindet sich der Haupttext. Er ist umgeben von Firmeninformation
   und Navigationsmenüs. Für sehende Personen ist es kein Problem den
   Haupttext zu identifizieren, unter anderem da sie solche Anordnungen
   bereits gewohnt sind.

   Auch im Haupttext können Tabellen verwendet werden:

   Spalte 1 Spalte 2   Spalte 3
   Wert 1   Wert 2     Wert 3
   Wert über 2 Spalten Wert 4
   Subnavigation
     * Navigation A
     * Navigation B
     * Navigation C
     * Navigation D
     * Navigation E

   © by Firma

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 5.3.: Ergebnis in einem Textbrowser. Die Hauptinformation kommt erst nach der Navigation.

Abbildung 5.4.: Verbesserte Tabellenstruktur, bei der der Hauptteil in Voice- und Textbrowsern vor der Navigation erscheint.

Für das Beispiel aus Abbildung 5.1 bieten sich folgende Links als Navigationshilfe an:

```
<a href="#navi">   Hauptnavigation </a>
<a href="#main">   Haupttext       </a>
<a href="#sub">    Subnavigation   </a>
<a href="#footer"> Footer          </a>
```

Da man mit Hilfe dieser Links Teile einer Webseite überspringen und auslassen kann, werden sie „Skip Links“ genannt.

Damit diese Links in einem graphischen Browser nicht angezeigt werden, kann man entweder die Schriftfarbe gleich der Hintergrundfarbe wählen, oder einfach (1 Pixel große) transparente Bilder mit entsprechenden `alt`-Attributen verwenden. Abbildung 5.5 zeigt einen Ausschnitt aus der entsprechend gestalteten Seite aus Abbildung 5.1.

```
...
<a href="#navi"><img src="blank.gif" border="0" alt="Hauptnavigation"/>
<a href="#main"><img src="blank.gif" border="0" alt="Haupttext"/></a>
<a href="#sub"><img src="blank.gif" border="0" alt="Subnavigation"/></a>
<a href="#footer"><img src="blank.gif" border="0" alt="Footer"/></a>
...
<a name="navi">
<b>Hauptnavigation</b>
...
<a name="main">
<h1>Überschrift</h1>
...
```

Abbildung 5.5.: Verbesserte Navigation durch Skip Links.

Mit Hilfe von Skip Links kann man Benutzern von Text- oder Voicebrowsern auch die Möglichkeit geben, Stellen zu überspringen, die möglicherweise nicht von Interesse für den Leser sind. Dies gilt unter anderem für ASCII-Art. In Abbildung 5.6 kann die ASCII-Art mit einem Skip Link übersprungen werden.

5.4. Table of Contents / Site Map

Nicht nur im Sinne des barrierefreien Webdesigns ist es sinnvoll, einen „Table of Contents“ (ToC) zur Verfügung zu stellen. Dabei handelt es sich um eine Seite, auf der eine Übersicht über das gesamte Angebot der Webseite gegeben wird. Die einzelnen Seiten und Abschnitte können über die Links auf dieser Seite erreicht werden.

Vor allem für blinde Personen und Personen mit kognitiven Einschränkungen ist es wichtig, die Struktur einer Webseite zu verdeutlichen. Dadurch wird es erleichtert, ein

```
Überspringen des <a href="#afterascii">ASCII-Arts</a>

                    ___.-~"~-._   __....__
                  .'     `     \ ~"~       ``-.
                 /`  _       )   `\               `\
                /`   a)     /      |                `\
               :`          /       |                  \
         <`-._|`   .-.   (       /    .              `;\\
          `-.  `--'_.'-.;\___/'    .       .         | \\
          _     /:--`      |         /      /         .'   \\
       ("\    /`/          |        '      '          /     :`;
       `\'\_/`/            .\      /`~`=-.:         /      ``
         `._.'             /`\     |        `\       /(
                          /  /\    |          `Y    /  \
                jgs  J   /   Y    |           |   /`\   \
                    /    |    |   |           |   |    |   |
                   "---"   /___|          /___|   /__|
                           '""""          '""""   '""""

<a name="afterascii">
```

Abbildung 5.6.: Überspringen von ASCII-Arts mit Skip Links

mentales Konzept der Webseite zu erstellen, und sich somit auf der Seite zurechtzufinden.

Für text- oder sprachbasierende Browser kann es hilfreich sein, über das `title`-Attribut wie in Abbildung 5.7 und Abbildung 5.8 die entsprechende Ebene mit anzugeben:

```
<ul>
   <li><a href="kapitel1.html" title="Ebene 1">Kapitel 1</a>
      <ul>
         <li><a href="kapitel1.html#Thema1" title="Ebene 2">1. Thema</a>
         <li><a href="kapitel1.html#Thema2" title="Ebene 2">2. Thema</a>
      </ul>
   <li><a href="kapitel2.html" title="Ebene 1">Kapitel 2</a>
      <ul>
         <li><a href="kapitel2.html#Thema1" title="Ebene 2">1. Thema</a>
         <li><a href="kapitel2.html#Thema1" title="Ebene 2">2. Thema</a>
      </ul>
</ul>
```

Abbildung 5.7.: Table of Contents mit title-Attribut.

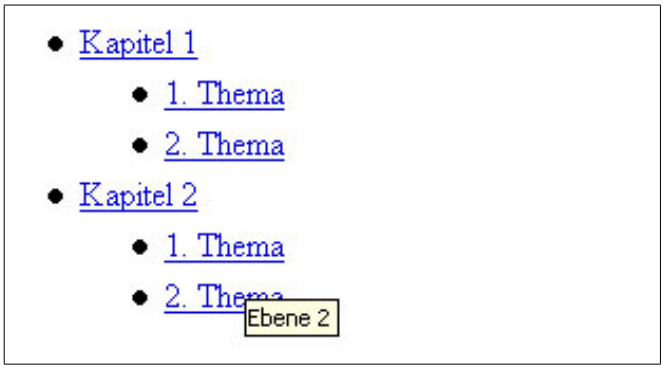


Abbildung 5.8.: Table of Contents in einem graphischen Browser.

5.5. Linklisten

Da Linklisten häufig auf Webseiten verwendet werden, wird auch hier auf deren übersichtliche Gestaltung eingegangen. Unachtsam gestaltete Linklisten sind oft schlecht lesbar und daher sehr unübersichtlich. Das Beispiel in Abbildung 5.9 zeigt drei unterschiedliche Möglichkeiten eine Linkliste zu gestalten.

Kapitel 1: Einführung
Kapitel 2: Äquivalente Text-Information
Kapitel 3: Markup & Stylesheets I
Kapitel 4: Markup & Stylesheets II
Kapitel 5: Navigation

Kapitel 1: Einführung

Kapitel 2: Äquivalente Text-Information

Kapitel 3: Markup & Stylesheets I

Kapitel 4: Markup & Stylesheets II

Kapitel 5: Navigation

- Kapitel 1: Einführung
- Kapitel 2: Äquivalente Text-Information
- Kapitel 3: Markup & Stylesheets I
- Kapitel 4: Markup & Stylesheets II
- Kapitel 5: Navigation

Abbildung 5.9.: Unterschiedliche Gestaltung von Linklisten. Listen mit Aufzählungszeichen sind am besten lesbar.

Eine gängige, aber schlecht lesbare Möglichkeit ist es, Links innerhalb eines Textes untereinander aufzuführen. Die Lesbarkeit wird deutlich erhöht, wenn zwischen den Links der Zeilenabstand vergrößert wird. Umfasst ein Link mehr als eine Zeile, wird der positive Effekt dieser Darstellungsvariante noch deutlicher.

Werden die einzelnen Links zusätzlich durch Aufzählungszeichen markiert, wird die Lesbarkeit noch einmal bedeutend besser. Es bietet sich also an, Links als ungeordnete Liste zu gestalten.

Es ist nicht nur intuitiv zu erkennen, sondern auch empirisch belegt, dass die letzte Variante am besten zu lesen ist. Vor allem wenn die Listen länger werden, verschachtelt sind, oder die Texte nicht in eine Zeile passen, bietet sich an, die Liste mit dem `<ul>`-Tag oder kleinen Bildern zu gestalten.

5.6. Meta-Information

Bei Meta-Information handelt es sich um Information über eine Webseite, die in der `<head>`-Section des Sourcecodes angegeben, auf der Webseite selbst aber nicht dargestellt wird. Suchmaschinen greifen auf diese Information, die unter anderem Kurzbeschreibung und Schlagwörter umfasst, zu. Außerdem dient die Meta-Information dazu, den Titel und die inhaltlichen Zusammenhänge einer Webseite anzugeben.

Titel

Es ist wichtig, jeder Seite einen aussagekräftigen Titel zuzuweisen. Dieser wird in der Titelzeile, in der History und in den Bookmarks des Browsers sowie im Ergebnis bei

Suchmaschinen angezeigt. Der Titel sollte kurz und prägnant sein und sowohl den Namen der ganzen Webseite als auch der einzelnen Seite enthalten.

Inhalt

Die Beschreibung des Inhalts wird in erster Linie von Suchmaschinen verwendet, um die Seite zu indizieren. Daher sollte darauf geachtet werden, den Inhalt sorgfältig zu wählen, um das Hauptthema jeder einzelnen Seite zu beschreiben.

Diese Beschreibung kann mit Hilfe des `<meta>`-Tags spezifiziert werden:

```
<meta name="[Typ]" content="[Inhalt]" ...>
```

Die folgende Tabelle gibt einen Einblick, welchem „Typ“ eine Meta-Information entsprechen kann:

author	gibt den Autor der Webseite an.
expires	enthält Datum und Uhrzeit, wie lange der Inhalt der Seite gültig ist.
keywords	gibt eine Liste von Schlagwörtern an, die den Inhalt der Seite beschreiben. Dies hilft Search Engines, die Seite zu kategorisieren.
description	umfasst eine kurze Zusammenfassung der Seite.

Verweise

Ein großer Vorteil des Internets ist, dass man Dokumente durch Links vernetzen kann. Diese Eigenschaft kann auch innerhalb einer Webseite dazu verwendet werden, die Beziehungen der einzelnen Seiten zueinander zu verdeutlichen. Dies ist nicht nur durch herkömmliche Web-Links möglich, sondern auch als Meta-Information. Diese kann von Browsern dazu verwendet werden, eigene Menüs zu erstellen, innerhalb derer man auf die vorherige oder nachfolgende Seite und spezielle Seiten mit Informationen über das aktuelle Dokument zugreifen kann.

Mit Hilfe des `<link>`-Tags können die Zusammenhänge zwischen den einzelnen Webseiten angegeben werden. Wenn eine Seite z.B. in einer Reihe von Webseiten steht, kann man wie folgt angeben, welche Seite davor und welche Seite danach anzuzeigen ist:

```
<link rel="prev" content="chapter4.html">
<link rel="next" content="chapter6.html">
```

Das `rel`-Attribut dieser speziellen Links wird dazu verwendet anzugeben, welcher Zusammenhang zwischen der aktuellen Seite und der „Zielseite“ besteht. Die folgenden Verweise werden bereits von neueren Browsern unterstützt:

start	Es wird ein Verweis auf die Startseite des Dokumentes erstellt. Dies zeigt auch Suchmaschinen an, welche Seite als Startseite dient.
contents	Zeigt auf das Inhaltsverzeichnis des Dokumentes.
first	Verweist auf das erste Dokument in einer Reihe von Dokumenten.
prev	Zeigt das in der logischen Reihenfolge vorige Dokument an.
next	Führt zum nächsten Dokument.
last	Zeigt auf das letzte Dokument.
up, parent	Verweist in einer Hierarchie von Dokumenten auf das nächsthöhere Dokument.
index	Gibt ein Dokument, das einen Index der Webseite enthält, an.
search	Stellt einen Link zu einer Seite her, die die Suche innerhalb des Dokuments ermöglicht.
help	Führt zu einem Dokument, das zusätzliche Hilfe-Information über die Webseite enthält.
copyright	Zeigt ein Dokument an, das besondere Copyright Information enthält.
author	Ein Dokument über den Autor der Seite wird angegeben.
alternate	Verweist auf ein Dokument, das den Inhalt des aktuellen Dokuments in einer anderen Sprache oder in einem anderen Format enthält.
appendix	Führt zum Appendix des aktuellen Dokuments.
bookmark	Gibt die Seite an, die sich besser für eine Bookmark eignet als die aktuelle.
chapter	Führt zur Startseite des Kapitels, zu dem die aktuelle Seite gehört.
glossary	Zeigt auf das Glossar des Dokuments.
section	Gibt ein Dokument an, das das Sub-Kapitel zum aktuellen Dokument enthält.
subsection	Verweist auf ein Dokument, das das „Sub-Sub-Kapitel“ zum aktuellen Dokument ist.

Die Dokumente, die mit einem `<link>`-Tag angegeben werden, müssen nicht unbedingt HTML-Seiten sein. So kann man beispielsweise auch auf Dokumente im Postscriptformat mit demselben Inhalt verweisen. Mit dem `title`-Attribut kann das Ziel des Links näher beschrieben werden. Handelt es sich dabei um ein anderes Format als HTML, ist es sinnvoll, dies mit dem `type`-Attribut anzugeben.

Abbildung 5.10.: Ein anhand der Meta-Information erstelltes Menü im Opera-Browser.

```
<link media="print"
      title="Dokument in Postscript"
      type="application/postscript"
      rel="alternate"
      href="../ps/dok.ps">
```

Ist das Dokument in mehreren Sprachen verfügbar, wird darauf mit dem `rel`-Attribut „alternate“ hingewiesen, die Sprache des Dokumentes wird mit dem `hreflang`-Attribut angegeben:

```
<link title="Dokument in Holländisch"
      type="text/html"
      rel="alternate"
      hreflang="nl"
      href="../nl/dok.html">
```

5.7. Farben

Die Wahl der Farben spielt bei der Gestaltung von Webseiten eine große Rolle. Dabei sollten nicht nur ästhetische Gesichtspunkte betrachtet werden. Damit die Seiten auch von Menschen mit Sehbehinderungen verwendet werden können, sollte zum Beispiel der Kontrast zwischen angrenzenden Farben vor allem von Vordergrund und Hintergrund möglichst groß sein.

Eine Farbkombination, die für eine Person mit gutem Sehvermögen leicht unterscheidbar ist, kann für Personen mit Sehbehinderungen nahezu keinen Kontrast haben. Zur

richtigen Farbwahl muss man sich einiger im Folgenden erläuterter Faktoren bewusst sein.

Es gibt verschiedene Modelle um Farben zu beschreiben. Neben dem RGB-Modell (Rot-, Grün-, Blau-Modell), das zur Spezifikation von Farben in HTML verwendet wird, gibt es ein intuitiveres und für die Farbwahl besser geeignetes Farbmodell namens HSB. Die drei Buchstaben stehen für die Größen „Farbton (Hue)“, „Sättigung (Saturation)“ und „Helligkeit (Brightness)“.

Der Farbton ist das, was als Farbe empfunden wird, und wird im HSB-Modell als Winkel in einem Farbkreis angegeben. Abbildung 5.11 zeigt einen Farbkreis ohne die möglichen Abstufungen zwischen den Grundfarben. Die Sättigung wird auf einer Skala von Grau bis zur Reinfarbe gemessen. Ein auf Null reduzierter Sättigungsgrad führt zu Grau, ein hoher Sättigungsgrad lässt Farben leuchtend wirken. Helligkeit gibt die sichtbare Helligkeit verglichen mit einer Grauskala an, anders ausgedrückt, den Anteil an Licht, den wir bei einer Farbe wahrnehmen. 0% Helligkeit ergibt Schwarz, 100% den vollen Farbton. Abbildung 5.12 zeigt, in welchem Farbbereich sich Sättigung und Helligkeit auswirken.

Helligkeit

Um einen möglichst guten Kontrast zu erhalten, sollte sich die Helligkeit von Vordergrund- und Hintergrundfarbe deutlich unterscheiden. Es genügt nicht, Unterschiede im Farbton und in der Sättigung zu haben, denn besonders farbenblinde Personen könnten dadurch Schwierigkeiten bei der Wahrnehmung der einzelnen Farben haben.

Ob eine Webseite genügend Kontrast bietet, lässt sich am einfachsten testen, indem man sie sich nur in Graustufen ansieht. Farben mit ähnlicher Helligkeit werden dadurch fast ununterscheidbar.

Farbton

Abbildung 5.11 zeigt eine Unterteilung von Farbtönen in helle und dunkle Farben. Für eine gute Lesbarkeit ist es erforderlich, entweder für den Hintergrund dunkle Farben aus der unteren Kreishälfte und für den Vordergrund helle Farben aus der oberen Hälfte zu verwenden, oder umgekehrt. Ähnliche Farbwerte geben zumeist einen sehr schlechten Kontrast. Daher sollten Kombinationen von aneinander angrenzenden Farben vermieden werden.

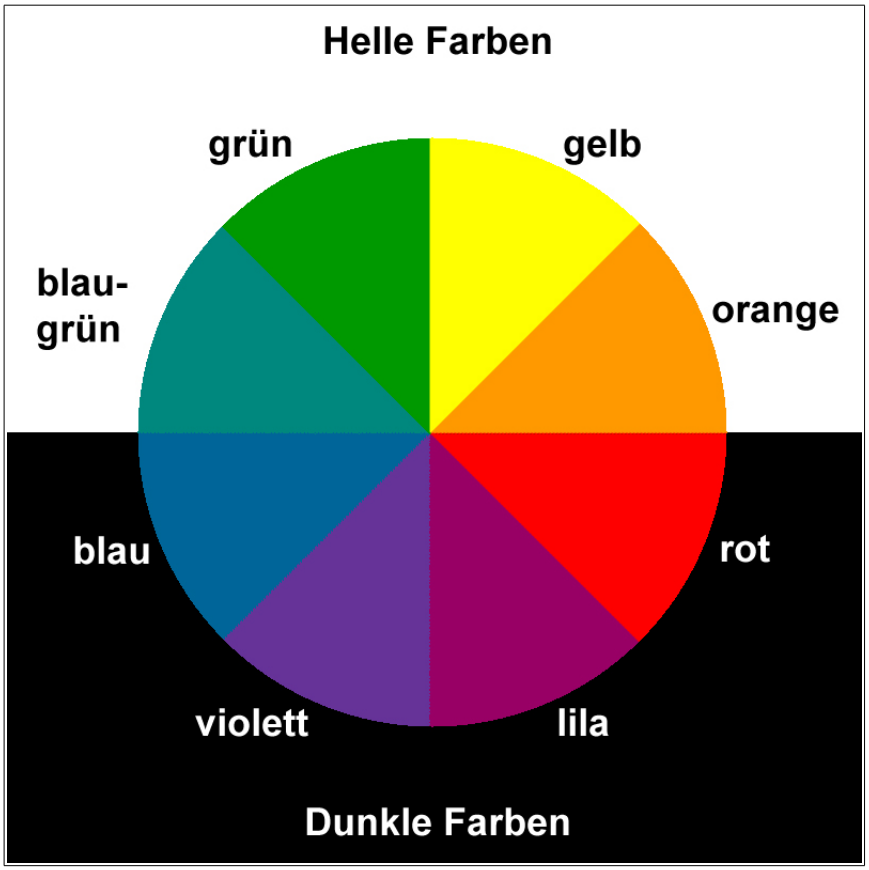


Abbildung 5.11.: Farbspektrum.

Abbildung 5.12.: Helligkeit und Sättigung.

Effekte

Weiters muss darauf geachtet werden, dass durch Bewegungseffekte die Lesbarkeit von Seiten nicht eingeschränkt wird. Dies gilt im Besonderen für flackernde Graphiken und Texte. Diese wirken sich nicht nur nachteilig auf die Lesbarkeit aus, sondern können sogar, wenn sie im Bereich von 4 bis 59 Hertz flackern oder aufblitzen, bei Menschen mit photosensitiver Epilepsie Anfälle auslösen.

5.8. Schriftarten

Ebenso wichtig wie die richtige Wahl der Farben ist die Wahl der Schriftarten. Für Dokumente im Papierformat haben sich Schriften mit Serifen durchgesetzt, da sie besser lesbar sind. Ob sich Schriften mit Serifen oder serifenlose Schriftarten für das Lesen am Bildschirm besser eignen, ist umstritten, auch wenn die Mehrheit für serifenlose Schriftarten plädiert.

Auf jeden Fall sollten „condensed“ (verdichtete) Schriftarten und Schriftarten mit ungewöhnlichem Buchstabensatz, wie zum Beispiel Schreibschriften, vermieden werden. In Abbildung 5.13 ist ersichtlich, dass diese schlechter lesbar sind als Standardschriftarten.

Abbildung 5.13.: Verschiedene, unterschiedlich gut lesbare Schriftarten.

Außerdem muss darauf geachtet werden, dass jede Schriftart entsprechend groß dargestellt werden muss, oder dass dem Benutzer ermöglicht wird, die Schriftart zu vergrößern. Dazu dienen, wie im Kapitel 4 „Markup & Stylesheets II“ erwähnt, relative Größenangaben.

6. Links & Image-Maps

6.1. Links

Das Internet unterscheidet sich von anderen Informationsmedien unter anderem durch die Möglichkeit, Informationen miteinander zu vernetzen. Daher stellen Links ein zentrales Gestaltungsmittel von Webseiten dar und deren Gestaltung erfordert besondere Sorgfalt. Neben der passenden Auswahl der zur Verfügung gestellten Links spielen auch die Textwahl und das Layout von Links eine wichtige Rolle, um eine optimale Lesbarkeit von Webseiten zu gewährleisten.

Der Text, der als Link erscheint, sollte sowohl kurz und bündig als auch aussagekräftig sein. Zu lange Texte können die Übersichtlichkeit beeinträchtigen, durch aussagekräftige Links ist es leichter, sich auf einer Seite zurechtzufinden und zur gesuchten Information zu gelangen.

Das folgende Beispiel zeigt drei unterschiedlich übersichtliche Möglichkeiten einen Link zu gestalten:

- Hier klicken für weitere Bilder zum aktuellen Thema.
- Weitere Bilder zum aktuellen Thema.
- Weitere Bilder zum aktuellen Thema.

Der Linktext „Hier klicken“ bzw. „click here“ ist sehr verbreitet, aber dennoch eine schlechte Wahl. Der Text hat keine Bedeutung außerhalb des Kontextes und wäre in einer Liste von Links, wie man sie sich von manchen Browsern zu einer Webseite anzeigen lassen kann, unbrauchbar. Der zweite Linktext ist unnötig lang und kann zum Beispiel durch Umbrechen am Ende einer Zeile die Lesbarkeit beeinträchtigen. Der dritte Linktext hingegen ist kurz und aussagekräftig, und die Bedeutung bleibt auch außerhalb des Kontextes erhalten.

Front Loading

Ein Prinzip, das auch auf die Wahl von Linktexten angewendet werden kann, ist „Front Loading“. Es hat zum Ziel, die Kernaussagen oder die Hauptinformation eines Textab-

schnittes möglichst früh darzubieten. Dies verkürzt Suchzeiten und erleichtert Benutzern, die mit Thema oder Sprache nicht vertraut sind, die Orientierung. Dies betrifft im Besonderen auch Menschen, die auf Webseiten nur einen serialisierten Zugriff haben, wie z.B. durch die Verwendung von Text- oder Voice-Browsern. Auch für die Gestaltung von Linktexten ist es wichtig, die Hauptinformation frühstmöglich darzubieten. Praktisch bedeutet das, dass vor allem in Linklisten der Link seiner genaueren Beschreibung vorausgeht:

- Internet-Standards können unter W3C nachgeschlagen werden.
- W3C stellt Internet-Standards zur Verfügung.

Der zweite Link ist besser, da die wichtige Information am Anfang steht. Dies ist vor allem bei Linklisten wichtig.

Das Title-Attribut

Eine nähere Beschreibung von Links kann mit Hilfe des `title`-Attributes vorgenommen werden. Dieser Text wird von den meisten Browsern als Tooltip dargestellt und kann zusätzlich Information zur besseren Orientierung enthalten.

```
Bilder in
<a href="lores.html" title="10 Bilder zu je 10kB">kleiner</a> oder
<a href="hires.html" title="10 Bilder zu je 80kB">großer</a> Auflösung.
```

Kann die Bedeutung eines Links nur aus dem Kontext geschlossen werden, ist es sinnvoll, den Kontext mit dem `title`-Attribut in Abbildung 6.1 mit einzubeziehen. Ohne die zusätzliche Information, um welches Kapitel es sich bei den Dokumenten, von denen nur das Format bekannt ist, handelt, wären die Links in einer aus der Webseite generierten Linkliste kaum brauchbar.

Links gruppieren

Ist es notwendig, eine größere Gruppe von Links darzustellen, sollte ein Skip-Link eingebaut werden um es Benutzern von Voice- und Textbrowsern zu ermöglichen diesen Abschnitt zu überspringen. Eine weitere Möglichkeit ist auch, die Links mit dem `<map>`-Tag zu gruppieren, um Browsern automatisches Überspringen dieser Linkgruppen zu ermöglichen, auch wenn die aktuellen Browser diese Möglichkeit nicht unterstützen (Abbildung 6.3).

```
<table>
   <tr>
      <th>Kapitel 1</th>
      <td><a href="doc1.html" title="Kapitel 1 in HTML">HTML</a></td>
      <td><a href="doc1.pdf" title="Kapitel 1 als PDF">PDF</a></td>
      <td><a href="doc1.txt" title="Kapitel 1 im Text-Format">Text</a>
   </tr>
   <tr>
      <th>Kapitel 2</th>
      <td><a href="doc2.html" title="Kapitel 2 in HTML">HTML</a></td>
      <td><a href="doc2.pdf" title="Kapitel 2 als PDF">PDF</a></td>
      <td><a href="doc2.txt" title="Kapitel 2 im Text-Format">Text</a>
   </tr>
   <tr>
      <th>Kapitel 3</th>
      <td><a href="doc3.html" title="Kapitel 3 in HTML">HTML</a></td>
      <td><a href="doc3.pdf" title="Kapitel 3 als PDF">PDF</a></td>
      <td><a href="doc3.txt" title="Kapitel 3 im Text-Format">Text</a>
   </tr>
</table>
```

Abbildung 6.1.: Angabe des Kontextes von Links durch das title-Attribut.

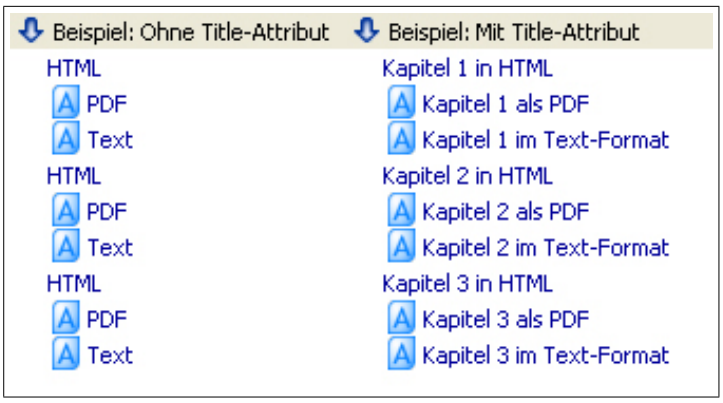


Abbildung 6.2.: Linkliste einer Webseite, vom Opera-Browser automatisch generiert, links ohne und rechts mit Title-Attribut.

```
<a href="#skip-navi"><img src="blank.gif"
                          alt="Skip Navigation" width="1"></a>
<map title="Navigation">
   <a href="doc1.html">Kapitel 1 |
   <a href="doc2.html">Kapitel 2 |
   <a href="doc3.html">Kapitel 3 |
   <a href="doc4.html">Kapitel 4 |
   <a href="doc5.html">Kapitel 5
</map>
<a name="skip-navi"/>
```

Abbildung 6.3.: Gruppieren von Links und Verwendung eines Skip-Links.

Links gestalten

Für die Navigation auf Webseiten ist es hilfreich, wenn bereits besuchte Links in einer anderen Farbe dargestellt werden als noch nicht besuchte Links. Es hat sich eingebürgert, Links unterstrichen darzustellen. Diese Art, Links zu kennzeichnen, hat sich bewährt, da die User sich daran gewöhnt haben.

Die Gestaltung der Links und der „visited“ Links kann im Stylesheet angegeben werden:

```
a          { font-decoration: underline; color: blue; }
a:visited  { font-decoration: underline; color: red; }
```

6.2. Image-Maps

Eine Image-Map ist ein Bild, dem mehrere Links hinterlegt sind. Je nachdem, wo genau auf das Bild geklickt wird, wird ein anderer Link aufgerufen. In HTML wird zwischen zwei Arten von Image-Maps unterschieden, Client-Side Image-Maps, bei denen die Zuordnung des Links beim Browser geschieht, und Server-Side Image-Maps, bei denen der Server diese Aufgabe übernimmt.

Client-Side

Bei Client-Side Image-Maps wird innerhalb der HTML-Seite angegeben, in welchem Bereich welcher Link aktiviert werden soll. Wird ein Bild als Image-Map verwendet, wird dieses in einzelne Regionen unterteilt. Jede einzelne Region wird mit Hilfe des `<area>`-Tags geometrisch beschrieben. Die geometrischen Beschreibungen aller Regionen

bezeichnet man als „Map“. Wenn der Benutzer auf das Bild klickt, kann der Browser bestimmen, welcher Link dieser Region zugeordnet ist und eine Anfrage an den Server senden.

In Abbildung 6.4 sieht man den Sourcecode einer Image-Map. Die geometrische Definition der Regionen wird jeweils innerhalb des `<area>`-Tags vorgenommen. Hier werden außerdem die Ziel-Adresse und eine äquivalente Textinformation zum Link angegeben.

```
<img usemap="#map1" src="images/imagemap2.jpg"/>

<map name="map1">
    <area href="chapter1.html"
          alt="Kapitel 1" title="Kapitel 1"
          shape="rect"    coords="0,0,94,35"/>
    <area href="chapter2.html"
          alt="Kapitel 2" title="Kapitel 2"
          shape="rect"    coords="95,0,189,35"/>
    <area href="chapter3.html"
          alt="Kapitel 3" title="Kapitel 3"
          shape="rect"    coords="190,0,285,35"/>
</map>
```

Abbildung 6.4.: Beispiel einer Client-Side Image-Map.

Wenn die einzelnen Regionen mit `alt`-Attributen versehen werden, lässt sich die Seite auch mit Text- oder Sprach-Browsern verwenden. Bei dem Programm Lynx wählt man zuerst die Image-Map aus, und bekommt anschließend eine Liste der Regionen mit den angegebenen `alt`-Attributen.

Der Code in Abbildung 6.5 stellt eine andere, für Textbrowser bessere Möglichkeit vor, Image-Maps zu definieren. Zur Spezifikation der einzelnen Regionen werden Textlinks verwendet, die als zusätzliche Angaben die entsprechenden Koordinaten der Region enthalten. In Voice- und Textbrowsern können so anstatt des Bildes die Links direkt ausgegeben werden.

Eine Verbesserung dieser Methode kann noch erfolgen, indem die Links der Image-Map zusätzlich außerhalb dieser auf der Seite angegeben werden. Dies kann erreicht werden, indem die Beschreibung, die in Form von Links mit zusätzlichen Koordinatenangaben außerhalb der Image-Map platziert und von der Image-Map referenziert wird (Abbildung 6.6). Dadurch können nicht nur Voice- und Textbrowser, sondern auch graphische Browser ohne die Anzeige von Bildern die Links darstellen (Abbildung 6.7).

```
<object usemap="#map1" type="image/jpg" data="images/imagemap2.jpg">
   <map name="map1">
       <a href="chapter1.html"
             shape="rect"    coords="0,0,94,35">Kapitel 1</a> |
       <a href="chapter2.html"
             shape="rect"    coords="95,0,189,35">Kapitel 2</a> |
       <a href="chapter3.html"
             shape="rect"    coords="190,0,285,35">Kapitel 3</a>
   </map>
</object>
```

Abbildung 6.5.: Client-side Imagemap mit Alternative für Text- und Voicebrowser.

```
<object usemap="#map2" type="image/jpg" data="images/imagemap2.jpg">
</object>
...
beliebiger Text
...
<map name="map2">
    <a href="chapter1.html"
          shape="rect"    coords="0,0,94,35">Kapitel 1</a> |
    <a href="chapter2.html"
          shape="rect"    coords="95,0,189,35">Kapitel 2</a> |
    <a href="chapter3.html"
          shape="rect"    coords="190,0,285,35">Kapitel 3</a>
</map>
```

Abbildung 6.6.: Client-side Imagemap, bei der die Linksliste in jedem Fall angezeigt wird.

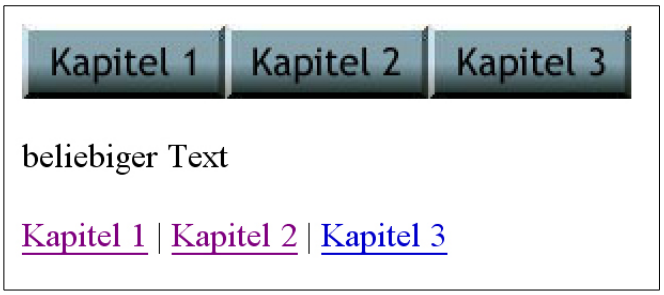


Abbildung 6.7.: Image-Map mit zusätzlichen Textlinks.

Server-Side

Bei Server-Side Image-Maps wird im Gegensatz zu Client-Side Image-Maps nur eine Seite aufgerufen, und die Koordinaten des Maus-Klicks werden an den Server übertragen. Der Server berechnet dann, welcher Link aufgerufen wurde und leitet auf die ermittelte Seite weiter.

```
<a href="http://www.somewhere.com">
   <img src="imagemap.gif" ismap="1" alt="Somewhere">
</a>
```

Server-Side Image-Maps sind nicht im Sinne des barrierefreien Webdesigns, da die Information, welcher Link ausgeführt wird, in der HTML-Seite nicht existiert. Die einzige Möglichkeit, diese Information bereitzustellen, wäre eine zusätzliche Link-Liste, ähnlich wie bei Client-Side Image-Maps.

Wenn die Liste weit von der Image-Map entfernt ist, macht es Sinn, im Text auf die Position dieser Liste zu verweisen (Abbildung 6.8).

```
<a href="http://www.somewhere.com">
   <img src="imagemap.gif" ismap="1"
         alt="Image-Map (Navigation am Ende der Seite)">
</a>
...
Hauptteil des Dokumentes
...
Navigation: <a href="chapter1.html">Kapitel 1</a> |
            <a href="chapter2.html">Kapitel 2</a> |
            <a href="chapter3.html">Kapitel 3</a>
```

Abbildung 6.8.: Server-side Image-Map mit dem Verweis auf eine textuelle Navigation.

Nahezu jede Server-Side Image-Map kann durch eine Client-Side Image-Map ersetzt werden, da die einzelnen Regionen durch beliebig komplexe Polygone spezifiziert werden können. Daher sollte grundsätzlich zugunsten barrierefreien Webdesigns auf Server-Side Image-Maps verzichtet werden.

7. Tabellen

Nahezu jede Seite im Internet enthält eine oder mehrere Tabellen. Dabei kann zwischen „Daten-Tabellen“ und „Layout-Tabellen“ unterschieden werden. In den Anfängen des Internets wurden Tabellen ausschließlich dafür verwendet, tabellarische Daten darzustellen. Heute werden Tabellen überwiegend zur graphischen Gestaltung eingesetzt, häufig sind sie verschachtelt.

Die Linearisierung von „Layout-Tabellen“ durch nicht-graphische Browser und die damit verbundenen Probleme wurden bereits im Kapitel 5 „Layout & Navigation“ beschrieben. Im folgenden Kapitel geht es um die Gestaltung von „Daten-Tabellen“ im Sinne des barrierefreien Webdesigns.

Wenn „Layout-Tabellen“ richtig eingesetzt werden, ist die linearisierte Version meist gut zu lesen – bei Daten-Tabellen ist dies oft nicht der Fall. Abbildung 7.1 und Abbildung 7.2 zeigen ein Beispiel einer Daten-Tabelle.

```
<table cellspacing="1">
   <tr><th rowspan="5" class="big">
       <img src="../images/schedule.gif"
             alt="[Stundenplan erste Klasse]"></th>
       <th>Tag</th>
       <th>7:00</th><th>8:00</th><th>9:00</th>
       <th>10:00</th><th>11:00</th><th>12:00</th></tr>

   <tr><td class="day">Montag</td>
       <td class="topic1" colspan="2">Mathematik</td>
       <td class="topic2" colspan="2">Deutsch</td>
       <td class="topic3" colspan="1">Geschichte</td></tr>
   ...
   <tr><td class="day">Donnerstag</td>
       <td class="topic2" colspan="1">Deutsch</td>
       <td colspan="2"></td>
       <td class="topic5" colspan="3">Englisch</td></tr>
</table>
```

Abbildung 7.1.: Sourcecode eines Stundenplans.

Stundenplan 1. Klasse	Tag	7:00	8:00	9:00	10:00	11:00	12:00
	Montag	Mathematik		Deutsch		Geschichte	
	Dienstag	Mathematik		Biologie	Englisch		
	Mittwoch	Biologie	Deutsch			Geschichte	
	Donnerstag	Deutsch			Englisch		

Abbildung 7.2.: Stundenplan, von einem graphischen Browser dargestellt.

Sehende Benutzer können problemlos die Zusammenhänge dieser einfachen Tabelle erfassen. Um zu erkennen, wann genau eine Unterrichtseinheit stattfindet, muss der Benutzer nur die Spalten bis zur obersten Zelle weiterverfolgen. Probleme treten auf, wenn die Seite mit einem Text- oder Voicebrowser dargestellt wird.

Wie in Abbildung 7.3 ersichtlich, werden die Zelleninhalte ohne entsprechende Abstände direkt hintereinander dargestellt. Das geschieht, da eine tabellarische Darstellung die maximale Zeilenbreite von 80 Zeichen übersteigen würde. Entfernt man den `alt`-Text „Stundenplan der ersten Klasse“, so wird man diese maximale Zeilenbreite nicht überschreiten, die Tabelle kann auch von Textbrowsern tabellarisch dargestellt werden.

```
                                                                    Stundenplan

   [Stundenplan erste Klasse] Tag 7:00 8:00 9:00 10:00 11:00 12:00
   Montag Mathematik Deutsch Geschichte
   Dienstag Mathematik Biologie Englisch
   Mittwoch Biologie Deutsch Geschichte
   Donnerstag Deutsch Englisch

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 7.3.: Stundenplan, von einem Textbrowser dargestellt. Die Zusammenhänge sind nicht mehr zu erkennen.

7.1. caption-Tags und summary-Attribute

Werden Tabellen mit einer Überschrift versehen, ist dies vor allem für Text- und Voicebrowser praktisch, um einen ersten Eindruck über den Inhalt der Tabelle zu vermitteln. Dazu kann das `<caption>`-Tag verwendet werden. Die Position und die Formatierung dieser Überschrift kann mittels Stylesheet festgelegt, oder wie in Abbildung 7.4 für graphische Browser komplett ausgeblendet werden.

Um die Position der Überschrift innerhalb einer Tabelle zu verändern gibt es die CSS-Property `caption-side` mit den möglichen Werten `top`, `bottom`, `left` und `right`. Leider unterstützen die aktuellen graphischen Browser diese CSS-Property nur bedingt, die Anzeige oberhalb der Tabelle wird allerdings meistens unterstützt.

Mit dem `summary`-Attribut kann ähnlich wie mit dem `longdesc`-Attribut für Bilder ein längerer Erklärungstext für Tabellen mit angegeben werden. Dieser Text wird von Text- oder Voicebrowsern dargestellt. Dieser Text kann den Inhalt der Tabelle entweder zusammenfassen, oder sogar komplett wiedergeben (Abbildung 7.4).

```
<table cellspacing="1" summary="Die Tabelle zeigt den Stundenpan der
   ersten Klasse von Montag bis Donnerstag
   zwischen 7:00 und 12:00 Uhr.
   Montag: 2h Mathematik, 2h Deutsch, 1h Geschichte.
   Dienstag: 2h Mathematik, 1h Biologie, 2h Englisch.
   Mittwoch: 1h Biologie, 2h Deutsch, 1h frei, 1h Geschichte.
   Donnerstag: 1h Deutsch, 2h frei, 3h Englisch.">
   <caption style="display:none;">Stundenplan erste Klasse</caption>
   <tr><th rowspan="5" class="big">
       <img src="../images/schedule.gif" alt=""></th>
        ...
</table>
```

Abbildung 7.4.: Stundenplan unter Verwendung des caption- und das summary-Tags.

Um Freistunden auch mit Text- und Voicebrowsern darzustellen, werden transparente Bilder mit entsprechenden `alt`-Attributen eingefügt. Diese werden anstelle der leeren Zellen angezeigt. Um Fächer, die länger als eine Stunde dauern, zu kennzeichnen, werden in die Zellen transparente Bilder eingefügt, anstatt derer die `alt`-Attribute mit der Angabe der Dauer ausgegeben werden. Wie die Tabelle mit den Änderungen aus Abbildung 7.4 aussieht, ist in Abbildung 7.5 zu sehen.

```
                                                                Stundenplan

   CAPTION: Stundenplan erste Klasse

      Tag        7:00   8:00    9:00   10:00  11:00   12:00
   Montag     Mathematik 2h  Deutsch 2h    Geschichte
   Dienstag   Mathematik 2h  Biologie Englisch 2h
   Mittwoch   Biologie Deutsch 2h     frei Geschichte
   Donnerstag Deutsch  frei 2h        Englisch 3h
Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 7.5.: Stundenplan in einem Textbrowser mit Caption und Freistunden.

7.2. Das headers-Attribut

Voicebrowser können die Tabelle aus Abbildung 7.5 immer noch nicht korrekt wiedergeben, da sie den Text nur zeilenweise vorlesen. Dadurch geht die Information der Tabelle verloren. Der Zusammenhang der einzelnen Zellen mit deren Überschriften ist somit nicht mehr gegeben. Mit dem `headers`-Attribut kann genau dieser Zusammenhang erhalten bleiben.

Die einzelnen Zellen, die eine Überschrift enthalten, werden jeweils mit einem `id`-Attribut versehen, das dieser Zelle einen eindeutigen Namen zuweist. Bei jeder Datenzelle kann dann, wie in Abbildung 7.6 gezeigt, im `headers`-Attribut eine Liste der zugehörigen Überschriften-IDs angegeben werden.

```
<table cellspacing="1" summary="...">
   <caption style="display:none;">Stundenplan erste Klasse</caption>
   <tr><th rowspan="5" class="big">
       <img src="../images/schedule.gif" alt=""></th>
       <th>Tag</th>
       <th id="h7">7:00</th><th id="h8">8:00</th>
       <th id="h9">9:00</th><th id="h10">10:00</th>
       <th id="h11">11:00</th><th id="h12">12:00</th></tr>

   <tr><td class="day" id="mon">Montag</td>
       <td class="topic1" colspan="2" headers="mon h7">Mathematik
           <img src="../images/blank.gif" alt="2h"></td>
       <td class="topic2" colspan="2" headers="mon h9">Deutsch
           <img src="../images/blank.gif" alt="2h"></td>
       <td class="topic3" colspan="1" headers="mon h11">Geschichte
       </tr>
   ...
   <tr><td class="day" id="thu">Donnerstag</td>
       <td class="topic2" colspan="1" headers="thu h7">Deutsch</td>
       <td colspan="2" headers="thu h8">
          <img src="../images/blank.gif" alt="frei 2h"></td>
       <td class="topic5" colspan="3" headers="thu h10">Englisch
           <img src="../images/blank.gif" alt="3h"></td>
</table>
```

Abbildung 7.6.: Sourcecode des Stundenplans unter Angabe der Zusammenhänge mit Hilfe von headers-Attributen.

Die Ausgabe eines Voicebrowsers könnte dann in etwas so aussehen:

```
...
Montag 7:00 Mathematik 2h Montag 9:00 Deutsch 2h Montag 11:00 Geschichte
...
Donnerstag 7:00 Deutsch Donnerstag 8:00 frei 2h
Donnerstag 10:00 English 3h
```

Bei komplexen Tabellen kann eine Zelle auch mehr als zwei Überschriften haben. Abbildung 7.7 und Abbildung 7.8 zeigen eine Tabelle mit dreidimensionalen Daten, bei der für die einzelnen Datenzellen je drei IDs im `headers`-Attribut eingetragen sind.

```
<table>
   <tr><th></th>
       <th id="eur">Europa</th>
       <th id="usa">USA</th>
       <th id="jap">Japan</th>

   <tr><th class="year" colspan="4" id="y1">2001</th></tr>
   <tr><th id="h1">1. Halbjahr</th>
      <td class="data" headers="y1 h1 eur">34.897,-</td>
      <td class="data" headers="y1 h1 usa">17.660,-</td>
      <td class="data" headers="y1 h1 jap">22.098,-</td></tr>
   <tr><th id="h2">2. Halbjahr</th>
      <td class="data" headers="y1 h2 eur">45.184,-</td>
      <td class="data" headers="y1 h2 usa">22,504,-</td>
      <td class="data" headers="y1 h2 jap">19.548,-</td></tr>

   <tr><th class="year" colspan="4" id="y2">2002</th></tr>
   <tr><th id="i1">1. Halbjahr</th>
      <td class="data" headers="y2 i1 eur">53.287,-</td>
      <td class="data" headers="y2 i1 usa">22.388,-</td>
      <td class="data" headers="y2 i1 jap">23.900,-</td></tr>
   <tr><th id="i2">2. Halbjahr</th>
      <td class="data" headers="y2 i2 eur">45.387,-</td>
      <td class="data" headers="y2 i2 usa">30,345,-</td>
      <td class="data" headers="y2 i2 jap">24.119,-</td></tr>
</table>
```

Abbildung 7.7.: Sourcecode einer Tabelle mit dreidimensionalen Daten.

7.3. Relative Größenangaben

Auch für Tabellen sollten im Sinne des barrierefreien Webdesigns relative Größenangaben verwendet werden. Dadurch können diese einerseits mit verschiedenen Bildschirmauf-

	Europa	USA	Japan
2001			
1. Halbjahr	34.897,-	17.660,-	22.098,-
2. Halbjahr	45.184,-	22,504,-	19.548,-
2002			
1. Halbjahr	53.287,-	22.358,-	23.900,-
2. Halbjahr	45.357,-	30,345,-	24.119,-

Abbildung 7.8.: Sourcecode einer „dreidimensionalen Tabelle“.

lösungen und Fenstergrößen optimal dargestellt werden, andererseits kann auch die Schriftgröße verstellt werden. Auch wenn beispielsweise durch das Verändern der Schriftgröße die Seite nicht mehr dem geplanten Design entspricht, ein Grund, warum die „äußersten Tabellen“ oft mit absoluten Größenangaben festgelegt werden, sollte auf absoluten Größenangaben zugunsten sehbehinderter Personen verzichtet werden.

8. Formulare

Viele Webseiten enthalten Formulare, um Benutzerdaten an den Server zu schicken. Bei der Gestaltung von Formularen muss auf die speziellen Anforderungen von Text- und Voicebrowsern geachtet werden. Die beiden Hauptaspekte, um Formulare barrierefrei zu gestalten, sind das Gruppieren von Informationen und die Zuordnung von Beschriftungen zu Eingabefeldern.

Ein weiterer Aspekt, der oft vergessen wird, ist, dass nicht jeder Benutzer eine Maus als (primäres) Eingabegerät verwendet. Auf manchen Seiten ist die Navigation unter ausschließlicher Verwendung mit der Tastatur sehr schwierig oder sogar unmöglich.

8.1. Gruppieren von Informationen

Gerade bei umfangreicheren Formularen ist es wichtig, zusammengehörende Elemente auch als solche zu kennzeichnen. Dafür wird in HTML das `<fieldset>`-Tag zur Verfügung gestellt. Damit lassen sich mehrere Elemente gruppieren und mit einer Überschrift, die innerhalb des `<fieldset>`-Tags durch ein `<legend>`-Tag definiert wird, versehen. In Abbildung 8.1 und Abbildung 8.2 ist ein Formular dargestellt, bei dem eine Gruppierung vorgenommen wurde.

Ein Vorteil der Gruppierung mittels `<fieldset>`-Element gebenüber einer graphischen Lösung ist, dass diese Gruppierung mit der Überschrift auch bei Text- und Voicebrowsern dargestellt werden kann. Soll die Gruppierung dennoch mit graphischen Elementen durchgeführt werden, kann trotzdem das `<fieldset>`-Tag verwendet, und der Rahmen und die Überschrift ausgeblendet werden:

```
<fieldset style="border: hidden;">
   <legend style="display:none">
      <b>Name &amp;amp; Adresse</b>
   </legend>
   <input  type="text" name="name">
   ...
</fieldset>
```

Auch wenn man diese Überschrift in graphischen Browsern ausblendet, kann es sinnvoll sein, diese trotzdem zu formatieren, da Textbrowser Formatierungen wie fett, kursiv und unterstrichen darstellen.

```
<fieldset>
   <legend>Personaldaten</legend>
   Bitte geben Sie ihre Personaldaten bekannt:<br>
   <table>
      <tr><td>Nachname:</td>
          <td><input name="lastname" type="text"></td></tr>
      <tr><td>Vorname:</td>
          <td><input name="firstname" type="text"></td></tr>
      <tr><td>Adresse:</td>
          <td><input name="address" type="text"></td></tr>
   </table>
</fieldset>

<fieldset>
   <legend>Haus &amp; Freizeit</legend>
      Hat Ihnen das Haus und die Umgebung gefallen?<br>
      <input type="radio" name="house" value="1"> ja<br>
      <input type="radio" name="house" value="2"> eher ja <br>
      <input type="radio" name="house" value="3"> eher nein <br>
      <input type="radio" name="house" value="4"> nein <br>
      <br>
      Hat Ihnen das Freizeitangebot gefallen?<br>
      <input type="radio" name="free" value="1"> ja<br>
      <input type="radio" name="free" value="2"> eher ja <br>
      <input type="radio" name="free" value="3"> eher nein <br>
      <input type="radio" name="free" value="4"> nein <br>
</fieldset>
```

Abbildung 8.1.: Sourcecode eines Feedbackformulars mit Gruppierung durch das fieldset-Element.

Feedback Formular

Personaldaten

Bitte geben Sie ihre Personaldaten bekannt:

Nachname:

Vorname:

Addresse:

Haus & Freizeit

Haben Ihnen das Haus und die Umgebung gefallen?

- ja
- eher ja
- eher nein
- nein

Hat Ihnen das Freizeitangebot gefallen?

- ja
- eher ja
- eher nein
- nein

Abschicken

Abbildung 8.2.: Feedbackformular, von einem graphischen Browser dargestellt, mit Gruppierung durch das fieldset-Element.

Wie bei jeder Überschrift ist die Wahl des Textes wichtig, damit die Gruppierung beim Lesen der Webseite hilfreich ist. Bei komplexen Formularen können zur besseren Übersicht `<fieldset>`-Elemente auch verschachtelt werden.

Eine weitere, inzwischen recht gut unterstützte Gruppierungsmöglichkeit besteht für Auswahlboxen. Hier können mit dem `<optgroup>`-Tag einzelne Optionspunkte gruppiert werden. Es ist den Browsern überlassen, wie genau diese Gruppierung dargestellt wird. In Abbildung 8.3 und Abbildung 8.4 ist zu sehen, dass die meisten graphischen Browser eine Überschrift vor die einzelnen Optionspunkte stellen.

```
<select name="sel">
   <optgroup label="Österreich">
      <option>Wien</option>
      <option>Linz</option>
      <option>Graz</option>
      <option>Salzburg</option>
   </optgroup>
   <optgroup label="Deutschland">
      <option>Berlin</option>
      <option>Hamburg</option>
      <option>München</option>
   </optgroup>
   <optgroup label="Italien">
      <option>Triest</option>
      <option>Rom</option>
   </optgroup>
</select>
```

Abbildung 8.3.: Sourcecode einer Auswahlbox mit Gruppierung durch das optgroup-Element.

8.2. Das Anordnen von Formularfeldern

Formularfelder können auf unterschiedliche Weise innerhalb eines Formulars angeordnet werden. Manche Anordnungen sind vor allem für Voice-Browser ungeeignet. Abbildung 8.5 zeigt eine solche schlechte Anordnung.

Probleme tauchen auf, wenn dieses Formular von einem Voicebrowser vorgelesen wird. Die Beschriftung des Formelementes und die Eingabefelder „stehen“ dann nicht mehr hintereinander. Die Ausgabe eines Voicebrowsers könnte sich etwa folgendermaßen anhören:

Abbildung 8.4.: Auswahlbox, von einem graphischen Browser dargestellt, mit Gruppierung durch das optgroup-Element.

Abbildung 8.5.: Eine horizontale Anordnung von Formelementen verursacht Probleme bei Voicebrowsern.

```
Formular: Horizontale Anordnung
Vorname
Nachname
Alter
[Eingabefeld]
[Eingabefeld]
[Eingabefeld]
```

In diesem Fall besteht die Schwierigkeit darin, dass die Beschriftung nicht direkt vor dem Eingabefeld steht. Für Voicebrowser eignet sich die vertikale oder die einzeilige Anordnung wesentlich besser. Diese werden in Abbildung 8.6 und Abbildung 8.7 dargestellt.

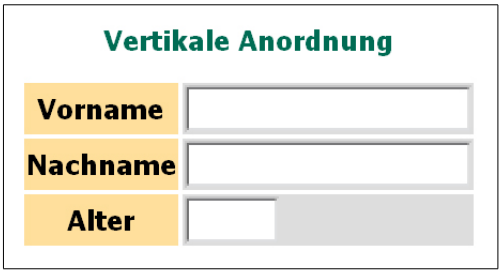


Abbildung 8.6.: Die vertikale Anordnung von Formelementen lässt sich besser vorlesen.

Abbildung 8.7.: Die einzeilige Anordnung lässt sich ebenso gut vorlesen wie die vertikale.

Da Beschriftung und Formelement in diesen Fällen nacheinander stehen, ist die Ausgabe eines Voicebrowsers besser zu verstehen:

```
Formular: Vertikale Anordnung
Vorname
[Eingabefeld]
Nachname
[Eingabefeld]
Alter
[Eingabefeld]
```

8.3. Labels

Mit dem `<Label>`-Tag gibt es eine explizite Möglichkeit, Beschriftungen und Formelemente zu verknüpfen. Diese Beziehung kann von Text- oder Voicebrowsern ausgewertet werden. Es gibt zwei Verwendungsmöglichkeiten für das `<Label>`-Tag. Für die erste Variante müssen Beschriftung und Formularfeld direkt hintereinander stehen:

```
<label>Vorname:  <input type="text" name="name"> </label>
<label>Nachname: <input type="text" name="lastname"> </label>
<label>Alter:    <input type="text" name="age"> </label>
```

Zumeist sind aber Beschriftung und Formelement durch Tabellenzellen o.ä. getrennt. Dann weist man, wie in Abbildung 8.8, jedem Formelement eine ID zu, auf die dann das `for`-Attribut des `<Label>`-Tags verweist.

```
<tr>
   <th><label for="name">Vorname</label></th>
   <th><label for="last">Nachname</label></th>
   <th><label for="age">Alter</label></th>
</tr>
<tr>
   <td><input type="text" id="name" name="firstname"></td>
   <td><input type="text" id="last" name="lastname"></td>
   <td><input type="text" id="age" name="age"></td>
</tr>
```

Abbildung 8.8.: Verknüpfung von Beschriftung und Formelementen mit Hilfe des label-Tags.

8.4. Titles

Da es nur ein Label für jedes Formularfeld geben darf, braucht es eine andere Strategie, um Eingabefelder mit mehr als einer Beschriftung zu handhaben. Ein Beispiel dafür ist in Abbildung 8.9 zu sehen.

Als Alternative zu Labels kann für mehrdimensionale Anordnungen von Formularelementen das `title`-Attribut verwendet werden, Abbildung 8.10 zeigt ein Beispiel, bei dem `title`-Attribute zur näheren Bestimmung von Eingabefeldern eingesetzt wurden.

Zweidimensionale Anordnung

	Vorname	Nachname	Alter
Spieler 1			
Spieler 2			
Spieler 3			

Abbildung 8.9.: Eine zweidimensionale Anordnung von Formelementen.

```
<tr>
   <th>Spieler 1</th>
   <td><input type="text" title="Vorname Spieler 1" name="firstname1">
   <td><input type="text" title="Nachname Spieler 1" name="lastname1">
   <td><input type="text" title="Alter Spieler 1" name="age1" size="5">
</tr>
<tr>
   <th>Spieler 2</th>
   <td><input type="text" title="Vorname Spieler 2" name="firstname2">
   <td><input type="text" title="Nachname Spieler 2" name="lastname2">
   <td><input type="text" title="Alter Spieler 2" name="age2" size="5">
</tr>
```

Abbildung 8.10.: Verwendung des Title-Attributes für zur Beschriftung von Eingabefeldern in Formularen.

8.5. Navigation mit der Tastatur

Die Benutzer von Voice- und Textbrowsern, aber auch Benutzer graphischer Browser, können nur die Tastatur als Eingabegerät verwenden. Daher muss beim Design von Formularen darauf geachtet werden, dass diese auch mit der Tastatur alleine verwendbar sind. Das betrifft vor allem Access-Keys, Tab-Reihenfolge und Javascript.

Access-Keys

Um in größeren Formularen schnell von einem Element zu einem anderen Element zu springen, kann einem Element ein Access-Key zugewiesen werden. Mit einer bestimmten Tastenkombination (Alt+Buchstabe) kann zu dem entsprechenden Element gesprungen werden, bei dem das `accesskey`-Attribut gesetzt ist. Wichtig ist, dass die Access-Keys auch entsprechend gekennzeichnet werden. In Abbildung 8.11 und Abbildung 8.12 werden sie unterstrichen dargestellt.

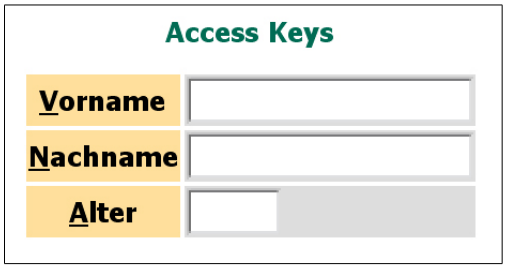


Abbildung 8.11.: Durch Unterstreichen gekennzeichnete Access-Keys in einem graphischen Browser.

Im Rahmen folgender Elemente kann das `accesskey`-Attribut angewendet werden: `<a>`, `<area>`, `<button>`, `<input>`, `<label>`, `<legend>` und `<textarea>`.

Leider unterstützen noch nicht alle (graphischen) Browser diese Möglichkeit, dennoch ist es sinnvoll, dieses Feature zu verwenden.

Tab-Reihenfolge

Um von einem Eingabeelement zum nächsten zu springen, gibt es in jedem Browser eine Taste oder eine Tastenkombination (zumeist die Tabulator-Taste). Um die einzelnen Elemente in eine sinnvolle Reihenfolge zu stellen, gibt es das `tabindex` Attribut.

Dem Attribut kann eine Zahl zwischen 0 und 32767 zugewiesen werden. Elemente mit kleinen Indizes stehen in der Reihenfolge vor Elementen mit großen Indizes. Haben zwei

```
<tr>
   <th><label for="name" accesskey="v"><u>V</u>orname</label></th>
   <td><input type="text" id="name" name="name"></td>
</tr>
<tr>
   <th><label for="lastname" accesskey="n"><u>N</u>achname</label></th>
   <td><input type="text" id="lastname" name="lastname"></td>
</tr>
<tr>
   <th><label for="age" accesskey="a"><u>A</u>lter</label></th>
   <td><input type="text" id="age" name="age" size="5"></td>
</tr>
```

Abbildung 8.12.: Sourcecode zur Verwendung von Access-Keys.

Elemente die gleiche (oder keine) Zahl, hängt die Reihenfolge von deren Position innerhalb des Quelltextes ab. Deaktivierte Elemente (`disabled=true`) werden ausgelassen.

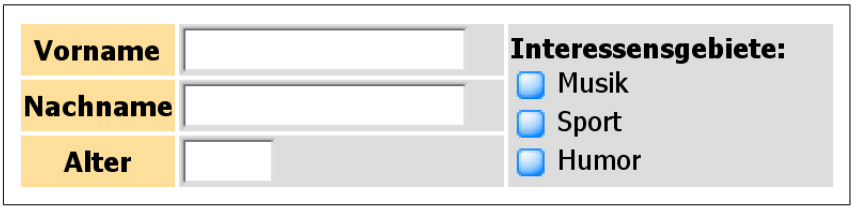


Abbildung 8.13.: Zweispaltige Anordnung eines Formulars.

Abbildung 8.13 zeigt eine zweispaltige Anordnung von Formularelementen. Ohne Angabe der Tab-Reihenfolge werden die Elemente aufgrund der Tabellenstruktur in der folgenden Reihenfolge ausgewählt:

```
Vorname-Eingabefeld
Musik-Checkbox
Sport-Checkbox
Humor-Checkbox
Nachname-Eingabefeld
Alter-Eingabefeld
```

Damit zuerst die drei Textfelder und anschließend die drei Checkboxen fokussiert werden, muss die Tab-Reihenfolge wie in Abbildung 8.14 dargestellt, explizit angegeben werden.

```
<tr>
   <th>Vorname</th>
   <td><input type="text" name="name" tabindex="1"></td>
   <td rowspan="3">
      <b>Interessensgebiete: </b><br>
      <input type="checkbox" name="interests" value="1" tabindex="4">
         Musik<br>
      <input type="checkbox" name="interests" value="2" tabindex="5">
         Sport<br>
      <input type="checkbox" name="interests" value="3" tabindex="6">
         Humor<br>
   </td>
</tr>
<tr>
   <th>Nachname</th>
   <td><input type="text" name="lastname" tabindex="2"></td>
</tr>
<tr>
   <th>Alter</th>
   <td><input type="text" name="age" size="5" tabindex="3"></td>
</tr>
```

Abbildung 8.14.: Angabe der Tab-Reihenfolge in einem Formular.

Javascript

In vielen Fällen sind mit Javascript gestaltete Interaktionen mit der Tastatur alleine schlecht oder gar nicht zu bedienen. In Abbildung 8.15 wird eine Sprachauswahl mittels Javascript dargestellt, bei der zwischen „Deutsch“, „Englisch“ und „Spanisch“ gewählt werden kann. Da aber auf die erste Änderung reagiert wird, d.h. auf den Mausklick oder auf das erste Nach-unten-Navigieren mit der Tastatur, kann die unterste Sprache ohne Maus nicht ausgewählt werden.

Dadurch wird auch den Usern die Kontrolle über den Ablauf genommen, da sie nicht selbst entscheiden können, wann sie die aktuelle Seite verlassen wollen. Ein weiteres Problem ist, dass besonders blinde Personen nicht erkennen können, dass die Seite gewechselt wird. Auf die genaueren Details im Umgang mit Javascript wird im Kapitel 11 „Browser-Kompatibilität“ eingegangen.

```
<form ...>
   <select name="lang"  onchange="submit();">
      <option>Bitte wählen Sie eine Sprache</option>
      <option value="de">Deutsch</option>
      <option value="en">English</option>
      <option value="sp">Spanisch</option>
   </select>
</form>
```

Abbildung 8.15.: Bei der Verwendung einer Auswahlbox mit Hilfe von Javascript kann das letzte Element nicht mit der Tastatur ausgewählt werden.

9. Frames

Mit Hilfe von Frames lassen sich Webseiten in mehrere Seiten aufteilen, die dann nebeneinander oder übereinander dargestellt werden. Dies wird oft zu Navigationszwecken eingesetzt. Abbildung 9.1 zeigt, wie eine Frameseite aufgebaut ist, Abbildung 9.2 zeigt die Darstellung der Seite mit einem graphischen Browser.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">

<html lang="ge">
   <head>
      <title>Schlechtes Beispiel einer Frameseite</title>
      <meta http-equiv="Content-Type"
            content="text/html; charset=ISO-8859-1">
   </head>
   <frameset rows="80px,*">
      <frame src="frame_navi1.html">
      <frame src="frame_main1.html" name="main">
   </frameset>
</html>
```

Abbildung 9.1.: Sourcecode einer einfachen Frameseite.

Frameseiten stellen für barrierefreies Webdesign eine Hürde dar, da nicht jeder Browser Frames darstellen kann. Manche Browser können nur einen Frame anzeigen. Von diesen Browsern wird zuerst eine Liste der verfügbaren Frames angezeigt, von denen der User einen Frame auswählen kann. Das führt dazu, dass sich das Verwenden solcher Seiten sehr mühsam gestalten kann.

Aus diesem Grund sollten Frameseiten zu Gunsten des barrierefreien Webdesigns vermieden werden. Werden Frameseiten dennoch verwendet, sollten einige Designgrundsätze beachtet werden, um Benutzern von Voice- und Textbrowsern das Surfen zu erleichtern.

Auf jeden Fall sollten stark verschachtelte Frameseiten und Frameseiten, die eine parallele Darstellung der einzelnen Seiten benötigen, vermieden werden.

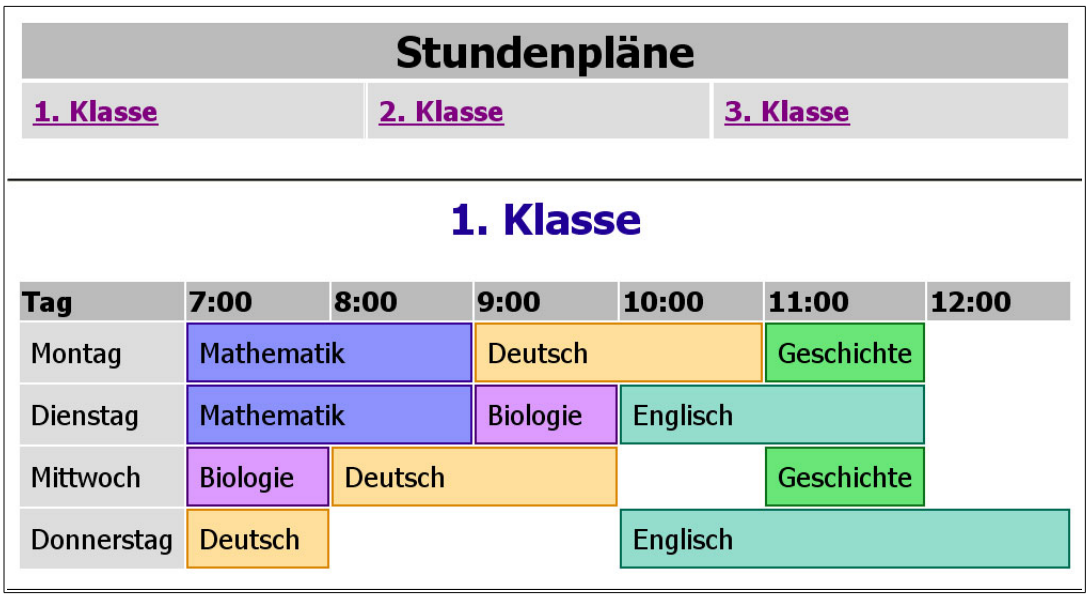


Abbildung 9.2.: Ein Frameset mit zwei Frames.

9.1. Das noframes-Tag

Mit Hilfe des `<noframes>`-Tags kann eine zusätzliche Textinformation angegeben werden, die bei Browsern, die keine Frames darstellen können, angezeigt wird. Dadurch ist es möglich, dem User eine Erklärung zum Inhalt der einzelnen Frames geben.

Eine andere Möglichkeit wird in Abbildung 9.3 gezeigt. Dort wird das `<noframes>`-Tag verwendet, um den direkten Zugriff auf die Stundenpläne zu ermöglichen. Dadurch wird der Frame mit den Links auf die Stundenpläne „überflüssig“ und den Usern wird die Verwendung der Seite erleichtert.

Je nachdem, ob das `<noframes>`-Tag vor oder nach den `<frame>`-Tags geschrieben wird, erscheint der zusätzliche Text auch vor oder nach der Liste der verfügbaren Frames. Abbildung 9.4 zeigt die Frameseite aus Abbildung 9.3 mit dem nach den `<frame>`-Tags eingefügten `<noframes>`-Tag in einem Text-Browser.

9.2. Das name- und das title-Attribut

Um den Benutzern von Browsern, die keine Frames anzeigen können, die Verwendung von Frameseiten zusätzlich zu erleichtern, ist es möglich, analog zur Beschreibung von Links und Bildern das `title`- und das `longdesc`-Attribut zu verwenden (siehe Kapitel 2 „Äquivalente Text-Information“ und Kapitel 6 „Links & Image-Maps“).

```
<frameset rows="80px,*">
   <frame src="frame_navi1.html">
   <frame src="frame_main1.html" name="main">
</frameset>
<noframes>
   <p>
      Stundenpläne für
      <ul>
         <li><a href="frame_main1.html" target="main">1. Klasse</a></li>
         <li><a href="frame_main2.html" target="main">2. Klasse</a></li>
         <li><a href="frame_main3.html" target="main">3. Klasse</a></li>
      </ul>
   </p>
</noframes>
```

Abbildung 9.3.: Sourcecode einer Frameseite mit noframes-Tag.

```
                                        Schlechtes Beispiel einer Frameseite

   FRAME: frame_navi1.html
   FRAME: main

   Stundenpläne für
     * 1. Klasse
     * 2. Klasse
     * 3. Klasse


Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Abbildung 9.4.: Darstellung einer Frameseite mit zusätzlicher Navigationsinformation in einem Text-Browser.

Wird nun von einem Textbrowser die Frameseite aufgerufen, sollten in der Liste der verfügbaren Frames deren Titel dargestellt werden. Leider ist dies nicht in allen Browsern der Fall. Es kommt auch vor, dass anstatt der Titel die `name`-Attribute angezeigt werden, die nur für den „internen Gebrauch“ spezifiziert sind. Daher sollten die `name`-Attribute der Frames ebenso aussagekräftig wie die `title`-Attribute sein.

In Abbildung 9.5 ist der Sourcecode einer Frameseite zu sehen, bei der sowohl ein `<noframes>`-Tag zur besseren Verwendbarkeit in Voice- und Textbrowsern als auch `title`-Attribute zur genaueren Beschreibung der Frames eingesetzt wurden.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">

<html lang="ge">
<head>
   <title>Frameseite mit noframes und title</title>
   <meta http-equiv="Content-Type" content="text/html;
         charset=ISO-8859-1">
</head>
<frameset rows="80px,*">
   <frame src="frame_navi1.html" name="Navigation"
          title="Navigation, Auswahl der Klasse">
   <frame src="frame_main1.html" name="main"
          title="Stundenplan der gewählten Klasse">
   <noframes>
      <p>
         Stundenpläne für
         <ul>
            <li><a href="frame_main1.html" target="main"
                   title="Stundenplan der 1. Klasse">1. Klasse</a></li>
            <li><a href="frame_main2.html" target="main"
                   title="Stundenplan der 2. Klasse">2. Klasse</a></li>
            <li><a href="frame_main3.html" target="main"
                   title="Stundenplan der 3. Klasse">3. Klasse</a></li>
         </ul>
      </p>
   </noframes>
</frameset>
</html>
```

Abbildung 9.5.: Sourcecode einer Frameseite mit noframes-Tag und title-Attributen.

Um eine gute Verwendbarkeit auch für Voice- und Textbrowser sicherzustellen, sollten nicht nur im Frameset, sondern auch in den einzelnen Seiten Titel mit Hilfe des `<title>`-Tags angegeben werden.

9.3. Alternative Seiten

Eine weitere Möglichkeit dafür, dass Frameseiten für Voice- und Textbrowser keine unüberwindbare Hürde darstellen, ist, eigene Seiten ohne Frames zur Verfügung zu stellen und als Einstiegsseite entsprechend zwei Links anzugeben. Wichtig dabei ist, dass auch die alternativen Seiten aktualisiert werden, da ansonsten der zusätzliche Aufwand nutzlos ist und die User wieder auf die Frameseite zugreifen müssen.

Da nicht sichergestellt werden kann, dass sich der User immer zuerst auf der Startseite befindet, da er die Seite beispielsweise von einer Suchmaschine aus aufgerufen hat, sollte es auch auf jeder Seite einen Link auf die frame-losen Webseiten geben.

10. Sprache

Auf Webseiten sollte eine möglichst einfache Sprache verwendet werden, um die Seiten für eine möglichst große Gruppe von Leuten zugänglich zu machen. Dies hilft vor allem Personen mit geistigen Einschränkungen, obwohl letztlich jeder Leser davon profitiert. Gerade im Internet ist die Sprache, in der eine Webseite verfasst ist, nicht unbedingt Muttersprache der einzelnen Leser.

In HTML gibt es insbesondere für mehrsprachige Webseiten einige technische Aspekte zur Verwendung von Sprache.

10.1. Spezifikation der Sprache

Es ist nötig, auf Webseiten die Sprache anzugeben, in der der Großteil des Inhalts verfasst ist. Zusätzlich kann für einzelne Ausdrücke oder Abschnitte eine Sprache explizit angegeben werden.

Die Angabe der Sprache hat mehrere Vorteile:

- Sprach-Synthesizer können die Sprache „erkennen“ und die Texte richtig „aussprechen“.
- Suchmaschinen verwenden die Sprachinformation, um die Seite richtig zu indizieren.
- Zukünftige Browserversionen könnten Typographie und Abteilungsregeln an die angegebene Sprache anpassen.
- Die Sprachangabe kann für automatische Übersetzung oder Rechtschreibkontrollen eingesetzt werden.

Die Angabe der Sprache erfolgt über das `lang`-Attribut. Die Hauptsprache eines Dokumentes wird innerhalb des `<html>`-Tags angegeben.

```
<html lang="de">
   ...
</html>
```

Sprachcodes bestehen aus einem primären, zumeist zweistelligen Kurzzeichen und beliebig vielen Subcodes, die, durch Bindestriche getrennt, angefügt werden können.

Die Subcodes sind meist Länderkurzzeichen, und werden in der Regel groß geschrieben, obwohl zwischen Groß- und Kleinschreibung nicht unterschieden werden muss.

Einige der wahrscheinlich häufigsten Sprachcodes sind in der folgenden Liste angegeben:

de	Deutsch
de-AT	Österreichisches Deutsch
de-DE	„Deutsches“ Deutsch
fr	Französisch
en	Englisch
en-US	U.S. Englisch
en-GB	Britisches Englisch

Eine vollständige Liste der zweistelligen Sprachcodes findet sich unter: `http://www.laurentia.com/iso639/lang-en.htm#table`

Auch bei nahezu jedem Element kann das `lang`-Attribut angegeben werden, um die Sprache anzugeben:

```
Wählen Sie eine der folgenden Sprachen:
   <a href="deutsch.html" lang="de">Deutsch</a><br>
   <a href="english.html" lang="en">English</a><br>
   <a href="spanish.html" lang="es">Español</a><br>
```

Bei Links kann zusätzlich noch die Sprache des Zieldokumentes angegeben werden, denn das `lang`-Attribut gibt nur die Sprache des Linktextes an:

```
<a href="spanish.html" lang="es" hreflang="es">Español</a>
```

10.2. Abkürzungen und Akronyme

Mit Hilfe von `<abbr>`- und `<acronym>`-Tags können Ausdrücke explizit als Abkürzungen oder Akronyme gekennzeichnet werden. Zusätzlich können die Bedeutung der Abkürzung, die Bestandteile des Akronyms und die zugrunde liegende Sprache angegeben werden.

```
<acronym lang="en" title="World Wide Web">WWW</acronym>
<abbr lang="de" title="Herr">Hr.</abbr>
```

Dies hilft nicht nur Lesern der Webseite beim Verständnis, sondern wird auch von Suchmaschinen, Voice-Browsern, Rechtschreibkontrollen und Übersetzungsprogrammen verwendet.

Sind Wörter als Abkürzung oder als Akronym gekennzeichnet, werden sie von manchen Browsern markiert. Die Erklärungen werden als Tooltips angezeigt, wenn der Mauszeiger über dem Ausdruck steht. Mit Hilfe von Stylesheets können Akronyme und Abkürzungen beliebig hervorgehoben werden:

```
<style> acronym { color: red; } </style>
...
<acronym lang="en" title="World Wide Web">WWW</acronym>
```

Es ist möglich, mit Hilfe von Stylesheets anzugeben, ob Abkürzungen und Akronyme als ein Wort ausgesprochen werden (wie z.B. TWAIN) oder buchstabiert werden sollen (z.B. WWW). Dies ist insbesondere für Voicebrowser eine Hilfe, den Text verständlich auszugeben. Außerdem ist es möglich, die Lesestimme von Voicebrowsern in Geschwindigkeit und Höhe zu verändern.

An dieser Stelle sei noch erwähnt, dass es für ein korrektes Wiedergeben durch die Leserstimme hilfreich ist, wenn nicht nur Sätze, sondern auch Listenelemente, Bildunterschriften, Überschriften und Ähnliches mit Punkten beendet werden. Bei Überschriften ist es unwahrscheinlich, dass sich „Schlusspunkte“ durchsetzen, da sie für sehende Leser irritierend wirken.

```
<style type="text/css">
   acronym, abbr   { color: red; cursor: help; }
   .normal         { speak: normal; }
   .spell          { speak: spell-out; }
</style>

...

<p>Die
<acronym class="spell" lang="en"
         title="Optical Character Recognition">
OCR</acronym>-Software benötigt einen Scanner mit einem
<acronym class="normal" lang="en"
         title="Technology Without Any Interesting Name">
TWAIN</acronym>-Treiber.</p>
```

Abbildung 10.1.: Sourcecode mit Verwendung von Akronymen und auditiven Stylesheets.

10.3. Political Correctness

Im Zuge der Political Correctness hat sich im deutschsprachigen Raum die weibliche Form von Hauptwörtern mit angefügtem, großgeschriebenem „In“ eingebürgert. Diese Form hat jedoch zwei große Nachteile im Hinblick auf Lesbarkeit und Verständlichkeit von Texten.

Einerseits ist für Personen mit Sehbehinderungen eine serifenlose Schriftart besser zu lesen als eine Schriftart mit Serifen. In diesen Schriftarten steht jedoch das große `I` zumeinst wie ein kleines `L` aus, was die Lesbarkeit deutlich verringert.

Andererseits tritt ein Problem auf, wenn Texte vorgelesen werden. Dabei stolpern Leser über diese Ausdrücke, und auch Programme machen Fehler, wenn innerhalb eines Wortes ein Großbuchstabe steht und somit wird das Zuhören sehr mühsam. Daher sollte auf diese Schreibweise gänzlich verzichtet werden und stattdessen keine Mühe gescheut werden entweder beide Formen gleichberechtigt nebeneinander zu stellen oder in einer Einleitung (disclaimer) auf die Gleichberechtigung beider Formen hinzuweisen, auch wenn nur eine Form Verwendung findet.

11. Browserkompatibilität

Im Zuge der vielfältigen Gestaltungsmöglichkeiten durch neue Technologien wie Javascript, Flash und Applets ist es wichtig, sich über Abwärtskompatibilität von Webseiten Gedanken zu machen. Jede Webseite muss auch mit einfachen Browsern noch verwendbar sein.

Die einfachste Testmöglichkeit dafür ist, die Webseite mit einem älteren Browser oder einem Textbrowser auszuprobieren. Gerade dadurch, dass Javascript nicht unterstützt wird, können manche Seiten nicht mehr verwendet werden.

Problematisch sind auch Webseiten, die browserspezifische Tags enthalten. Hier gilt, dass die Grundfunktionalität der Seite auch ohne diese Tags gegeben sein muss.

Es ist ebenfalls zu bedenken, dass nicht jeder Benutzer die Webseite mit einer Maus bedient. Deshalb muss darauf geachtet werden, dass die Webseite auch ausschließlich mit der Tastatur verwendet werden kann.

11.1. Javascript

In modernen Webseiten wird immer öfter Client-side Javascript eingesetzt. Dabei muss darauf geachtet werden, dass die Bedienung der Seite auch ohne Javascript möglich ist, um diese barrierefrei zu gestalten.

Darauf weisen auch die W3C-Guidelines hin:

> Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page.
> *Web Content Accessibility Guidelines Checkpoint 6.3*

Javascript soll dementsprechend nur so eingesetzt werden, dass die notwendige Funktionalität der Seite auch ohne Javascript erhalten bleibt. Praktisch bedeutet das, dass Anzeigeeffekte die zum Beispiel Menüpunkte mit einem Bild hinterlegen, demnach kein Problem sind, da sie für die Grundfunktionalität der Seite nicht wichtig sind. Popupfenster hingegen, die ohne Javascript nicht zugängliche Information enthalten, widersprechen den W3C Grundsätzen und sind für eine Vielzahl von Usern nicht zugänglich.

Im Wesentlichen gibt es zwei Einsatzbereiche für Javascript. Entweder wird die vorhandene Funktionalität der Webseite um Farbeffekte, Formularüberprüfungen, etc. erweitert, oder es wird zusätzliche Funktionalität wie Popupfenster, dynamische Texte, etc. zur Verfügung gestellt.

Im Folgenden werden vier typische Javascriptanwendungen behandelt und auf ihre Barrierefreiheit überprüft.

1. Textausgabe mit Javascript

Abbildung 11.1 zeigt ein Script, das am Ende der Seite das Datum der letzen Änderung hinzufügt. Da diese Information zum Verständnis und der Verwendbarkeit der Seite (wahrscheinlich) nicht unbedingt nötig ist, erfüllt dieser Script die W3C-Richtlinien, und ist somit barrierefrei.

```
<script language="JavaScript">
<!-- verbergen vor Browsern ohne Javascript
   document.write("<p style='align:center'>Zuletzt geändert am "
                  + document.lastModified + "</p>");
// -->
</script>
```

Abbildung 11.1.: Sourcecode zur Textausgabe mit Javascript.

2. Visuelle Änderungen mit Javascript

Javascript wird oft dafür eingesetzt, durch User-Interaktion das Aussehen von Elementen zu verändern. In Abbildung 11.2 ändert sich die Hintergrundfarbe der Tabellenzelle, wenn sich der Mauszeiger über die Zelle bewegt.

Auch dieses Beispiel ist für barrierefreies Webdesign unbedenklich. Graphische Browser mit aktiviertem Javascript geben dem Benutzer dadurch ein zusätzliches Feedback, das für Browser ohne Javascript irrelevant ist.

Webseiten, in denen Scripts nicht für zusätzliche Funktionalitäten oder zur Vermittlung zusätzlicher Information verwendet werden, sind im Sinne der W3C als barrierefrei anzusehen.

```
<style type="text/css">
   .white   { background: white;      }
   .red     { background: lightcoral; }
   .green   { background: lightgreen; }
   .blue    { background: lightblue;  }
</style>
...
<table>
   <tr>
      <td onMouseOver="className='red'"
          onMouseOut ="className='white'">Rot</td>
      <td onMouseOver="className='green'"
          onMouseOut ="className='white'">Grün</td>
      <td onMouseOver="className='blue'"
          onMouseOut ="className='white'">Blau</td>
   </tr>
</table>
```

Abbildung 11.2.: Sourcecode zur Änderung der Hintergrundfarbe mittels Javascript.

3. Validieren von Formularen mittels Javascript

Mit Hilfe von Javascript können Formulardaten auf ihre Gültigkeit überprüft werden. Abbildung 11.3 zeigt den Sourcecode eines Formulars, das nur dann abgeschickt wird, wenn Username und Passwort zumindest 6 Zeichen haben, und das Passwort zweimal gleich angegeben wurde. Ansonsten wird eine Fehlermeldung ausgegeben.

Die Validierung mittels Javascript hat gegenüber der serverseitigen Validierung mehrere Vorteile. Die serverseitige Validierung ist langsamer, da der User warten muss, bis die Anfrage am Server bearbeitet und die Fehlermeldung angezeigt wird. Bei der Verwendung von Javascript geschieht hingegen alles innerhalb des Browsers und nur gültige Angaben werden an den Server geschickt. Anschließend muss der User wieder auf die Formularseite zurückkehren.

Ein weiterer Vorteil ist, dass der Server ungültige Anfragen nicht bearbeiten muss, und somit weniger belastet ist. Dadurch wird weniger Bandbreite benötigt.

Ist Javascript jedoch nicht verfügbar oder deaktiviert, muss die Validierung trotzdem durchgeführt werden. Das kann dann nur am Server geschehen. Nur so kann gewährleistet werden, dass die Seite auch ohne Javascript barrierefrei ist. Wird die serverseitige Überprüfung unterlassen, kann das bei der Verwendung von Browsern ohne Javascript zu unerwarteten Fehlern führen. Daher sollte immer sowohl Javascript als auch serverseitige Validierung verwendet werden.

```
<script language="Javascript">
   <!--
      function checkForm()
      {
         if (document.form1.name.value.length<6)
         {
            alert("Der Username muss mindestens 6 Buchstaben haben.");
            return false;
         }
         if (document.form1.password.value.length<6)
         {
            alert("Das Passwort muss mindestens 6 Buchstaben haben.");
            return false;
         }
         if (document.form1.password.value !=
            document.form1.password2.value)
         {
            alert("Geben Sie zweimal das selbe Passwort an.");
            return false;
         }
         return true;
      }
   //-->
</script>
...
<form method="get" action="validateform.html" name="form1"
      onSubmit="return checkForm()">
   <table>
      <tr><th>Username</th>
         <td><input type="text" name="name" size="20"></td>
      <tr><th>Passwort</th>
         <td><input type="password" name="password" size="20"></td>
      <tr><th>Bestätigung</th>
         <td><input type="password" name="password2" size="20"></td>
   </table>
   <p><input type="submit" value="Abschicken"></p>
</form>
```

Abbildung 11.3.: Sourcecode zur Formvalidierung mit Javascript.

4. Nur durch Javascript zugänglicher Inhalt

Javascript wird oft für die Gestaltung der Navigation von Webseiten eingesetzt. Dabei sind Teile der Navigation oder der Information nur durch von Javascript unterstützter Interaktion zugänglich. Abbildung 11.4 zeigt ein Beispiel, bei dem durch das Klicken auf das „+“ Symbol ein Submenü sichtbar wird.

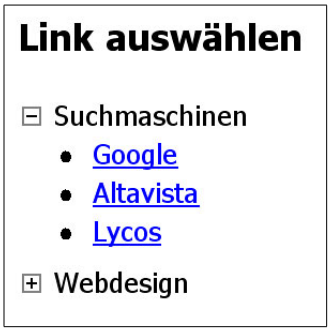


Abbildung 11.4.: Mittels Javascript gestaltetes Menü, dessen Unterpunkte nur mit Hilfe von Javascript erreichbar sind.

Bei einem Browser ohne Javascript ist es allerdings nicht möglich, die einzelnen Submenüpunkte auszuwählen, da das Submenü nur mit Hilfe von Javascript aufgerufen werden kann. Eine solche Navigation entspricht daher nicht den Anforderungen des barrierefreien Webdesign.

Dies bedeutet nicht, dass es notwendig ist, ganz auf Javascript zu verzichten, es muss allerdings eine Möglichkeit gefunden werden, die Information auch ohne Javascript zu erreichen. Dazu gibt es im Wesentlichen zwei Möglichkeiten, die anhand der nächsten Beispiele erläutert werden sollen.

Abbildung 11.5 und Abbildung 11.7 zeigen die etwas erweiterte Navigation aus Abbildung 11.4. Klickt man hier auf den neu hinzugefügten Link „Suchmaschinen“, öffnet sich eine eigene Seite, die die Links des ausgewählten Submenüs enthält. Dadurch ist es möglich, die Submenüpunkte auch ohne Javascript auszuwählen.

Eine andere Möglichkeit ist, die ohne Javascript nicht erreichbaren Submenüs in Browsern, die Javascript nicht unterstützen, mit Hilfe des `<noscript>`-Tags einzublenden. Browser, die Javascript unterstützen, zeigen den Inhalt des `<noscript>`-Tags nicht an, Browser, die Javascript nicht unterstützen, hingegen schon.

Diese Methode hat den Vorteil, dass keine zusätzlichen Seiten erzeugt werden müssen. Man nimmt aber in Kauf, dass die angezeigte Liste sehr lang werden kann.

Wann immer Javascript zu Navigationszwecken verwendet wird, muss es auch einen alternativen, unter Umständen längeren Weg zum gleichen Ziel geben.

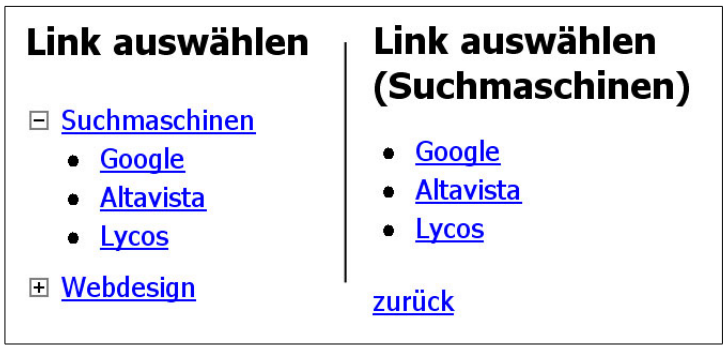


Abbildung 11.5.: Links: Die Hauptmenüpunkte sind mit einem Link hinterlegt, der auch ohne die Verwendung von Javascript zu den Unterpunkten führt. Rechts: Eigene Seite mit Submenü, die auch ohne Javascript zugänglich ist.

```
<img id="img1" src="../images/plus.gif" onClick="expand(1);" class="box">
<a href="hiddencontent_1.html">Suchmaschinen</a><br>
<div id="div1" style="display:none" class="menu">
   <ul>
      <li><a href="http://www.google.at">Google</a></li>
      <li><a href="http://www.altavista.at">Altavista</a></li>
      <li><a href="http://www.lycos.at">Lycos</a></li>
   </ul>
</div>
<img id="img2" src="../images/plus.gif" onClick="expand(2);" class="box">
<a href="hiddencontent_2.html">Webdesign</a><br>
<div id="div2" style="display:none" class="menu">
   <ul>
      <li><a href="http://www.w3.org">W3C</a></li>
      <li><a href="http://www.w3schools.com">W3C Schools</a></li>
      <li><a href="http://www.w3.org/TR/html4/">HTML 4.01 spec</a></li>
   </ul>
</div>
```

Abbildung 11.6.: Sourcecode eines Menüs mit Javascript, und mit Links hinterlegten Hauptmenüpunkten.

```
<img id="img1" src="../images/plus.gif" onClick="expand(1);" class="box">
Suchmaschinen<br>
<div id="div1" style="display:none" class="menu">
   <ul>
      <li><a href="http://www.google.at">Google</a></li>
      <li><a href="http://www.altavista.at">Altavista</a></li>
      <li><a href="http://www.lycos.at">Lycos</a></li>
   </ul>
</div>
<noscript>
   <div class="menu">
   <ul>
      <li><a href="http://www.google.at">Google</a></li>
      <li><a href="http://www.altavista.at">Altavista</a></li>
      <li><a href="http://www.lycos.at">Lycos</a></li>
   </ul>
   </div>
</noscript>
...
```

Abbildung 11.7.: Sourcecode eines Menüs mit Javascript bei dem ein noscript-Tag eingesetzt wurde.

Alle vier soeben vorgestellten Anwendungen von Javascript sind im Sinne des barrierefreien Webdesigns zugänglich, wenn Alternativen zur Verfügung gestellt werden, sodass die volle Funktionalität auch ohne Javascript verfügbar ist.

In den beiden abschließenden Abschnitten soll noch auf zwei grundlegende Prinzipien zur Verwendung von Javascript eingegangen werden, um Fehlerquellen zu verringern.

Interaktion ohne Maus

Bei der Erstellung von Webseiten wird oft von einer Standardausstattung der User ausgegangen, die jedoch nicht der Realität entspricht. So verwendet nicht jeder Benutzer eine Maus als Eingabegerät. Das bedeutet, dass Webseiten so gestaltet werden müssen, dass sie auch mit der Tastatur alleine zu bedienen sind.

Die Interaktion auf Webseiten geschieht aufgrund bestimmter, vom User ausgelöster Ereignisse. Diese lösen Eventhandler aus, die den weiteren Ablauf steuern. Diese Ereignisse können in zwei Gruppen eingeteilt werden:

- **Für ein Eingabegerät spezifische Events**
  Ereignisse, die durch eine Aktion eines bestimmten Eingabegerätes hervorgerufen werden. So kann man beispielsweise nur mit einer Maus „klicken“ und mit einer Tastatur „eine Taste drücken“.
  Eventhandler Maus: `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`
  Eventhandler Tastatur: `onkeypress`, `onkeydown`, `onkeyup`
- **Logische Events**
  Diese Events sind unabhängig vom Eingabegerät und haben eine Funktion innerhalb eines logischen Ablaufs.
  Logische Events: `onload`, `onunload`, `onfocus`, `onblur`, `onsubmit`, `onreset`, `onselect`, `onchange`

Da für ein Eingabegerät spezifische Events die Benutzbarkeit einer Seite zwangsläufig einschränken, sollten, wenn möglich, logische Events verwendet werden. Das Beispiel in Abbildung 11.8 und Abbildung 11.9 zeigt zwei Möglichkeiten, auf die Änderung in einer Listbox zu reagieren.

Die Änderung der Vordergrundfarbe geschieht mit Hilfe des logischen `onchange`-Eventhandlers, die der Hintergrundfarbe hingegen mit dem für die Maus spezifischen `onclick`-Eventhandler. Die Vordergrundfarbe lässt sich dementsprechend auch mit der Tastatur verändern und lässt den Usern die Wahl des Eingabegerätes offen.

Ähnliches gilt auch bei der im vorigen Abschnitt behandelten Validierung von Formularen. In Abbildung 11.3 wurde zur Kontrolle der Eingabe der `onselect`-Eventhandler

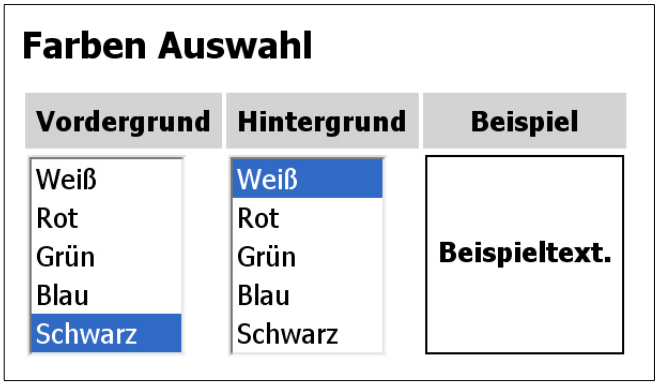


Abbildung 11.8.: Farbe und Hintergrundfarbe mit Javascript ändern.

```
<td>
   <select name="color" size="5"
      onChange="document.all.sample.style.color=this.value">
      <option value="#ffffff" selected>Weiß</option>
      <option value="#ff0000">Rot</option>
      <option value="#00ff00">Grün</option>
      <option value="#0000ff">Blau</option>
      <option value="#000000">Schwarz</option>
   </select>
</td>
<td>
   <select name="back" size="5"
      onClick="document.all.sample.style.background=this.value">
      <option value="#ffffff" selected>Weiß</option>
      <option value="#ff0000">Rot</option>
      <option value="#00ff00">Grün</option>
      <option value="#0000ff">Blau</option>
      <option value="#000000" selected>Schwarz</option>
   </select>
</td>
<td>
   <div id="sample">
      <b>Beispieltext.</b>
   </div>
</td>
```

Abbildung 11.9.: Sourcecode, in dem onchange und onclick Verwendung finden.

verwendet. Weniger geschickt wäre es, stattdessen den `onclick`-Eventhandler des Submit-Buttons zu verwenden, da `onclick` sich nur auf Mausklicks bezieht (auch wenn die meisten Browser auch beim Auswählen mit der Tastatur den `onclick`-Eventhandler aktivieren).

Links mit Javascript

Es existiert die Möglichkeit, ein Javascript anstelle von Linkadressen einzusetzen:

```
<a href="javascript:openwindow('new.html')">...</a>
```

Dies wird oft dazu verwendet, Fenster mit bestimmten Seiten in einer vorgegebenen Größe zu öffnen, oder andere Aktionen auszuführen. Das gerade gezeigte Beispiel zeigt einen Fall, der nur bei Browsern, die Javascript unterstützen, funktioniert. Es ist aber möglich, diese Links auch für Browser ohne Javascript zugänglich zu machen:

```
<a href="new.html"
   onClick="openwindow('new.html'); return false;">...</a>
```

In diesem Fall führt ein javascriptfähiger Browser die Funktion `openwindow` aus. Der Aufruf „`return false;`“ verhindert, dass der Link selbst auch ausgeführt wird. Browser ohne Javascript hingegen linken einfach auf die Seite `new.html`.

11.2. Alternative Seiten

Ist es unmöglich oder zu aufwendig, Seiten, die Technologien wie Flash oder Java Applets beinhalten, barrierefrei zu gestalten, so bietet sich die Möglichkeit, den Inhalt dieser Seiten auch für Voice- und Textbrowser in zugänglichen, alternativen Seiten darzustellen.

Da die Wartung und Aktualisierung dann aber parallel auf beiden Seiten geschehen muss, was einen gewissen Mehraufwand darstellt, sollte dies aber nur der „letzte Ausweg“ sein.

Die Auswahl so einer alternativen Seite kann entweder auf einer eigenen Willkommenseite oder durch eine Browserweiche geschehen. Es ist jedoch vorzuziehen, dem User diese Entscheidungsmöglichkeit zu geben.

Da entsprechende Webseiten nicht nur über die Hauptseite erreichbar sind, sollte sich zusätzlich auf jeder Seite ein Link auf die alternativen Seiten befinden.

12. Hilfsmittel

Es existiert eine Reihe von Programmen, die für das Erstellen barrierefreier Webseiten hilfreich sind. Dabei handelt es sich einerseits um Validatoren, die Webseiten auf ihre Korrektheit überprüfen oder um Programme, die zusätzlich noch Korrekturen vornehmen und andererseits um Browser, die Webseiten auf unterschiedliche Weise darstellen. Nicht alle diese Programme wurden speziell für barrierefreies Webdesign entwickelt, doch da sie das Erstellen barriererfreier Seiten unterstützen, sollen sie im Folgenden kurz vorgestellt werden.

12.1. Validatoren

Bei Validatoren handelt es sich um Programme, die Webseiten auf ihre Korrektheit überprüfen. Da jedoch nur ein kleiner Teil der Aspekte des barrierefreien Webdesigns automatisch überprüft werden kann, dienen diese Programme vor allem zur Erstkontrolle und zur regelmäßigen Überprüfung entstehender Webseiten.

W3C HTML Validation Service

Mit dem im Rahmen dieses Kurses bereits verwendeten W3C HTML Validation Service[1] kann man eine Seite auf HTML-Validität überprüfen. Als Grundlage dieses Tests dient die HTML-Spezifikation[2].

Alle Abweichungen von der HTML-Spezifikation der zu überprüfenden Seite werden dem Benutzer angezeigt. So kann die Seite Schritt für Schritt HTML-konform gestaltet werden.

Zur Überprüfung einer Seite kann dem Validation Service die URL der zu überprüfenden Seite übergeben werden, wenn diese online ist. Alternativ kann auch auf Webseiten von einem lokalen Computer zugegriffen und diese können von dort aus validiert werden.

Der Opera Browser[3] kann das W3C HTML Validation Service auch direkt mit einer

[1] `http://validator.w3.org/`
[2] `http://www.w3.org/TR/html4/`
[3] `http://www.opera.com/`

Tastenkombination (Strg+Alt+V) für die aktuell angezeigte Seite aufrufen. Bei anderen Browsern kann mit Hilfe eines „Lesezeichens“ mit dem folgenden Script jede Seite, die sich online im Internet befindet, einfach überprüft werden:

```
javascript:void(window.open('http://validator.w3.org/check?doctype=
%28detect+automatically%29&charset=%28detect+automatically%29
&uri='+escape(window.location)))
```

Die Validierung der in Abbildung 12.1 und Abbildung 12.2 gezeigten Seite würde (Schritt für Schritt) die folgenden Punkte beanstanden:

- Fehlende Angabe der Codierung der Seite. Ohne diese Angabe kann der Test nicht durchgeführt werden.
- Das schließende `<h2>`-Tag passt nicht zum öffnenden `<h1>`-Tag. Mehrere Fehlermeldungen werden für diesen einen Fehler generiert.
- Das `<img>`-Tag hat kein `alt`-Attribut.
- Das `<input>` zum Absenden des Formulars darf nicht an dieser Stelle stehen. Es kann z.B. in eine Tabellenzelle verschoben werden, oder in ein `<p>`-Tag eingeschlossen werden.

Jede Seite sollte zumindest mit dem W3C HTML Validation Service überprüft werden, bevor sie ins Internet gestellt wird. Damit kann man zumindest gewährleisten, dass die Seite von einem Großteil der Browser gut dargestellt werden kann. Manche Browser würden das fehlerhafte Schließen der Überschrift ignorieren, andere hingegen nicht.

Die korrigierte Seite könnte wie in Abbildung 12.3 aussehen.

Bobby

Bobby[4] ist ein Service der Watchfire Corporation[5]. Es überprüft, ob Webseiten den W3C Richtlinien entsprechen. Dies kann entweder online erfolgen, oder über eine gekaufte Version auf einem lokalen Computer.

Das Service findet einerseits Fehler der angegebenen Seite und weist andererseits auf Aspekte des barrierefreien Webdesigns hin, die vom Benutzer selbst überprüft werden sollen, da sie nicht automatisch überprüft werden können. Manche dieser Hinweise werden durch die Verwendung gewisser Tags ausgelöst, andere Meldungen erscheinen hingegen jedes Mal.

[4] `http://bobby.watchfire.com/bobby/html/en/index.jsp`
[5] `http://www.watchfire.com/`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head><title>Testseite</title></head>

<h1>Newsletter abonnieren</h2>

<form action="validator.html" method="get">
<table>
   <tr>
      <td rowspan="2"><img src="../images/newsletter.gif" width="100">
      <td>Name
      <td><input type="text" name="name">
   </tr>
   <tr>
      <td>Email
      <td><input type="text" name="email" id="email">
   </tr>
</table>
<input type="submit" value="Newsletter abonnieren">
</form>

</html>
```

Abbildung 12.1.: Sourcecode einer fehlerhafte HTML-Seite.

Abbildung 12.2.: Darstellung der fehlerhaften Webseite in einem graphischen Browser.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
   <title>Testseite</title>
   <meta http-equiv="Content-Type" content="text/html;
         charset=ISO-8859-1">
</head>

<h1>Newsletter abonnieren</h1>

<form action="validator.html" method="get">
<table>
   <tr>
      <td rowspan="2"><img src="../images/newsletter.gif" width="100"
                          alt="">
      <td>Name
      <td><input type="text" name="name">
   </tr>
   <tr>
      <td>Email
      <td><input type="text" name="email" id="email">
   </tr>
</table>
<p><input type="submit" value="Newsletter abonnieren"></p>
</form>

</html>
```

Abbildung 12.3.: Sourcecode der ausgebesserten HTML-Seite.

Die einzelnen Hinweise und Fehlermeldungen werden nach ihrer Wichtigkeit geordnet ausgegeben und entsprechen den drei Prioritätsstufen der WAI Richtlinien.

Bei der Validierung des Beispiels in Abbildung 12.1 liefert das Service unter anderem die folgenden Meldungen:

- Allen Bildern muss ein `alt`-Attribut zugewiesen werden.
- Wenn eine der verwendeten Tabellen nicht für Layoutzwecke, sondern als Datentabelle verwendet wird, sollten die Header entsprechend gekennzeichnet werden.
- Für alle Formularelemente sollen Labels verwendet werden.
- Für die Tabelle soll eine Zusammenfassung mit Hilfe des `<summary>`-Tags angegeben werden.
- Für die Webseite soll die Sprache angegeben werden.

Insgesamt gibt Bobby über 30 Meldungen aus, die teilweise eine Aktion nach sich ziehen sollten. Der Grund für die zahlreichen Meldungen ist, dass viele Punkte des barrierefreien Webdesigns nicht maschinell überprüft werden können. Ein Vorteil dieses Services ist, dass es auf diese kritischen Aspekte hinweist, und zum Nachdenken anregt.

Bobby gilt als Standard für barrierefreies Webdesign. Für Webseiten steht ein eigenes Symbol zur Verfügung, das anzeigt, dass eine Seite den W3C Richtlinien entspricht und von Bobby darauf überprüft wurde.

Abbildung 12.4.: Bobby Symbol.

WAVE

WAVE[6] ist ein Onlinetool von WebAIM (Web Accessibility in Mind), einem Projekt der Universität in Utah. Ähnlich wie Bobby ist WAVE ein Tool, das Webseiten auf die Einhaltung der W3C Richtlinien über barrierefreies Webdesign überprüft.

Im Gegensatz zu den beiden vorherigen Tools stellt WAVE die Auswertung nicht textuell, sondern graphisch dar. Es zeigt die Webseite erneut an und fügt Fehlermeldungen und Hinweise in Form von Grafiken ein. Die Auswertung der Webseite aus dem Beispiel in Abbildung 12.1 ist in Abbildung 12.5 dargestellt.

[6] `http://wave.webaim.org`

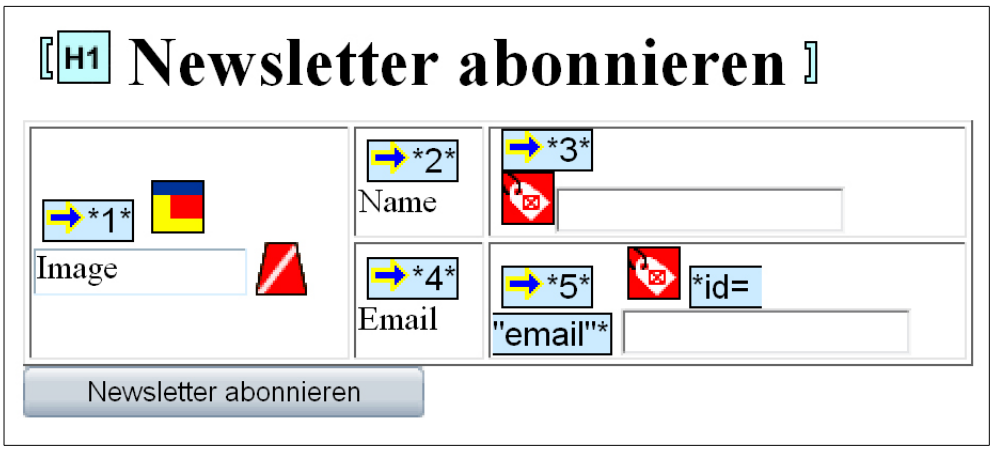


Abbildung 12.5.: Validierung mit WAVE.

Das trapezförmige rote Symbol zeigt an, dass für ein Bild kein `alt`-Attribut gesetzt wurde. Das gelb-blau-rote Symbol zeigt an, dass überprüft werden soll, ob eine Tabelle aus Layoutgründen notwendig ist. Die zwei roten Symbole rechts zeigen an, dass `label`-Attribute für die beiden Formularelemente fehlen.

Die Zahlen hinter den blauen Pfeilen zeigen an, in welcher Reihenfolge die Tabellenzellen von einem Voicebrowser vorgelesen werden würden, bzw. wie die Reihenfolge der Zellen der linearisiert Tabelle sein wird. Überschriften werden markiert und es wird angegeben, um welche Überschrift es sich handelt. Somit kann man auch die korrekte Strukturierung der Überschriften kontrollieren.

WAVE verwendet über 70 Zeichen, um die Struktur und die Problemfelder von Webseiten anzuzeigen. Dabei wird auch versucht, schlampige `alt`-Texte oder Vorschläge für Überschriften zu finden.

Die verschiedenen Zeichen existieren für die Bereiche Errors, HTML Alerts, Script Alerts, Media Alerts, Partial Alerts, Accessibility Features, Problematic Accessibility Features, Structural Elements und Semantic Elements. Die komplette Liste mit den Erklärungen[7] kann auf der WAVE Seite im Internet nachgelesen werden.

12.2. Korrigierende Tools

Neben validierenden Programmen existiert auch eine Reihe an Tools, die mit und ohne Userinteraktion Webseiten korrigieren, indem sie verbesserte Kopien der Seiten anlegen. Einige dieser Programme bieten zum Einfügen spezieller Texte wie z.B. `alt`-Texte eine Interaktion mit dem User an.

[7] `http://wave.webaim.org/wave/explanation.htm`

HTML Tidy

HTML Tidy[8] ist ein Tool, das automatisch Fehler in HTML-Seiten findet, ausbessert, und die korrigierten Seiten neu formatiert in eine Datei schreibt. Ursprünglich von der W3C entwickelt, ist HTML Tidy nun ein Open Source Projekt auf der SourceForge.net[9] Plattform.

Gerade unsaubere Verwendung von Markup wird von HTML Tidy entsprechend ausgebessert:

- Fehlende oder falsch schließende Tags werden korrigiert.

```
<h1>heading
<h2>subheading</h3>

<h1>heading</h1>
<h2>subheading</h2>
```

- Schließende Tags werden in die richtige Reihenfolge gebracht.

```
<p>Paragraph mit <b>fett, <i>fett kursiv,</b> fett?</i> normal?

<p>Paragraph mit <b>fett, <i>fett kursiv,</i> fett?</b> normal?
```

- Falsch platzierte `<hr>`-Tags werden ausgebessert.

```
<h1><hr>Überschrift</h1>
<h2>Unter-<hr>Überschrift</h2>

<hr>
<h1>Überschrift</h1>
<h2>Unter-</h2>
<hr>
<h2>Überschrift</h2>
```

- Listen werden korrigiert.

```
<body>
<li>1st list item
<li>2nd list item

<body>
<ul>
<li>1st list item</li>
<li>2nd list item</li>
</ul>
```

- Anführungszeichen werden zu Attributen hinzugefügt.

[8] http://tidy.sourceforge.net/
[9] http://www.sourceforge.net/

```
<img src=picture.gif>

<img src="picture.gif">
```

- Warnungen über unbekannte oder proprietäre Tags und Attribute werden ausgegeben.
- ...

Auf das Beispiel in Abbildung 12.1 angewendet liefert HTML Tidy das in Abbildung 12.6 abgebildete Ergebnis. HTML Tidy hat schließende Tags hinzugefügt, den Fehler in der Überschrift ausgebessert und ein `<body>`-Tag eingefügt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
   <head>
      <meta name="generator" content=
      "HTML Tidy for Windows (vers 1st April 2002), see www.w3.org">
      <title>Testseite</title>
   </head>
   <body>
      <h1>Newsletter abonnieren</h1>
      <form action="validator.html" method="get">
         <table>
            <tr>
               <td rowspan="2"><img src="../images/newsletter.gif"
               width="100"></td>
               <td>Name</td>
               <td><input type="text" name="name"></td>
            </tr>
            <tr>
               <td>Email</td>
               <td><input type="text" name="email" id="email"></td>
            </tr>
         </table>
         <input type="submit" value="Newsletter abonnieren">
      </form>
   </body>
</html>
```

Abbildung 12.6.: Von HTML Tidy korrigierter Sourcecode.

Das Verhalten und die Form des Layouts von HTML Tidy kann individuell eingestellt werden. So kann z.B. ganz auf Einrückungen verzichtet oder angegeben werden, an welchen Stellen Zeilenumbrüche eingefügt werden sollen.

Zusätzlich wird angemerkt, dass die Tabelle kein `<summary>`-Tag, und das Bild kein `alt`-Attribut verwendet. Von automatisch generierten `alt`-Texten sollte jedoch im Sinne des barrierefreien Webdesigns Abstand genommen werden.

A-Prompt

Das Programm A-Prompt[10] wird an der Universität Toronto am Adaptive Technology Resource Centre[11] entwickelt. Im Gegensatz zu den bisherigen Programmen stellt das Programm die Fehler und Probleme einer Webseite in einer graphischen Benutzeroberfläche (Abbildung 12.7) dar.

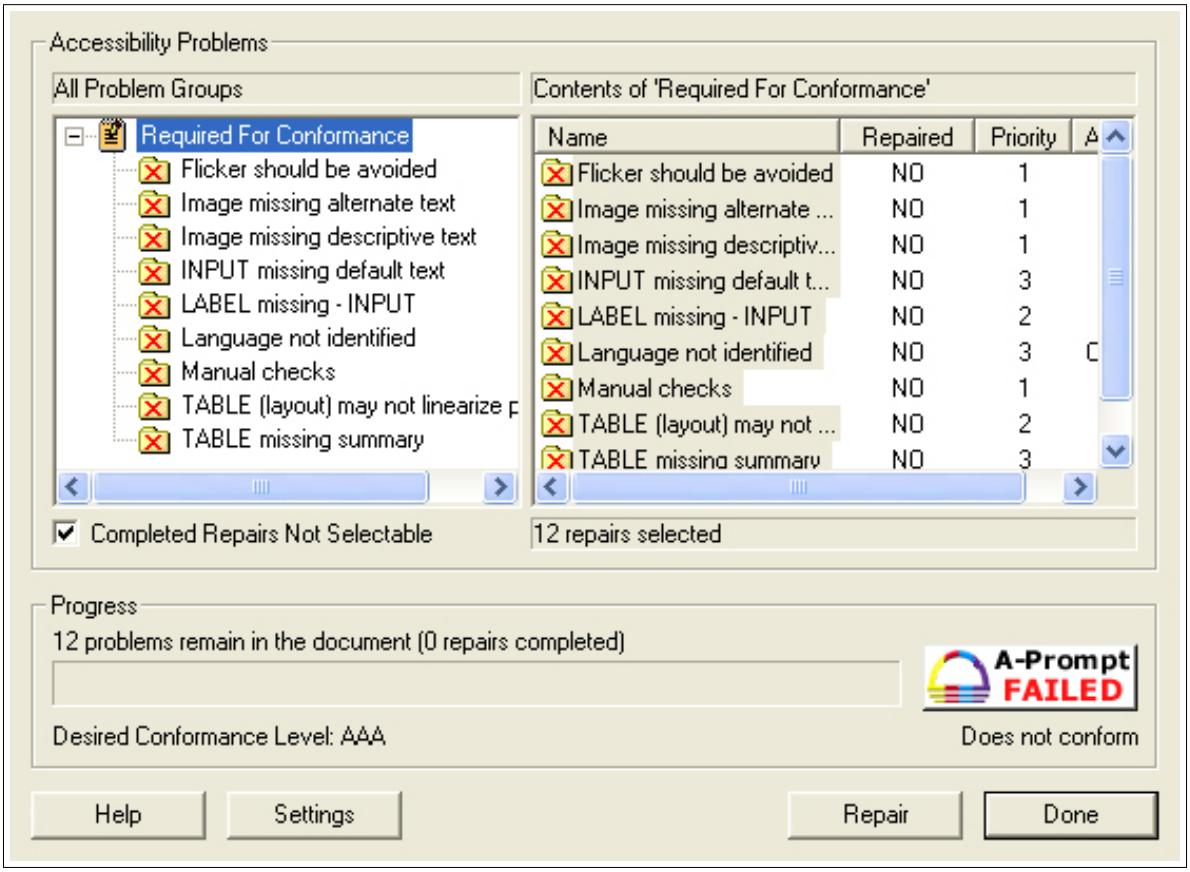


Abbildung 12.7.: Graphische Benutzeroberfläche von A-Prompt.

Die einzelnen Kritikpunkte können interaktiv behandelt und ausgebessert werden. Abbildung 12.8 zeigt den Dialog zur Eingabe eines `longdesc`-Attributes. Hier kann,

[10] `http://aprompt.snow.utoronto.ca/index.html`
[11] `http://www.utoronto.ca/atrc/`

wenn nötig, entweder ein Beschreibungstext eingegeben oder auf eine Datei verwiesen werden. Zusätzlich kann auf Wunsch ein d-Link mit einer neuen Datei erzeugt werden.

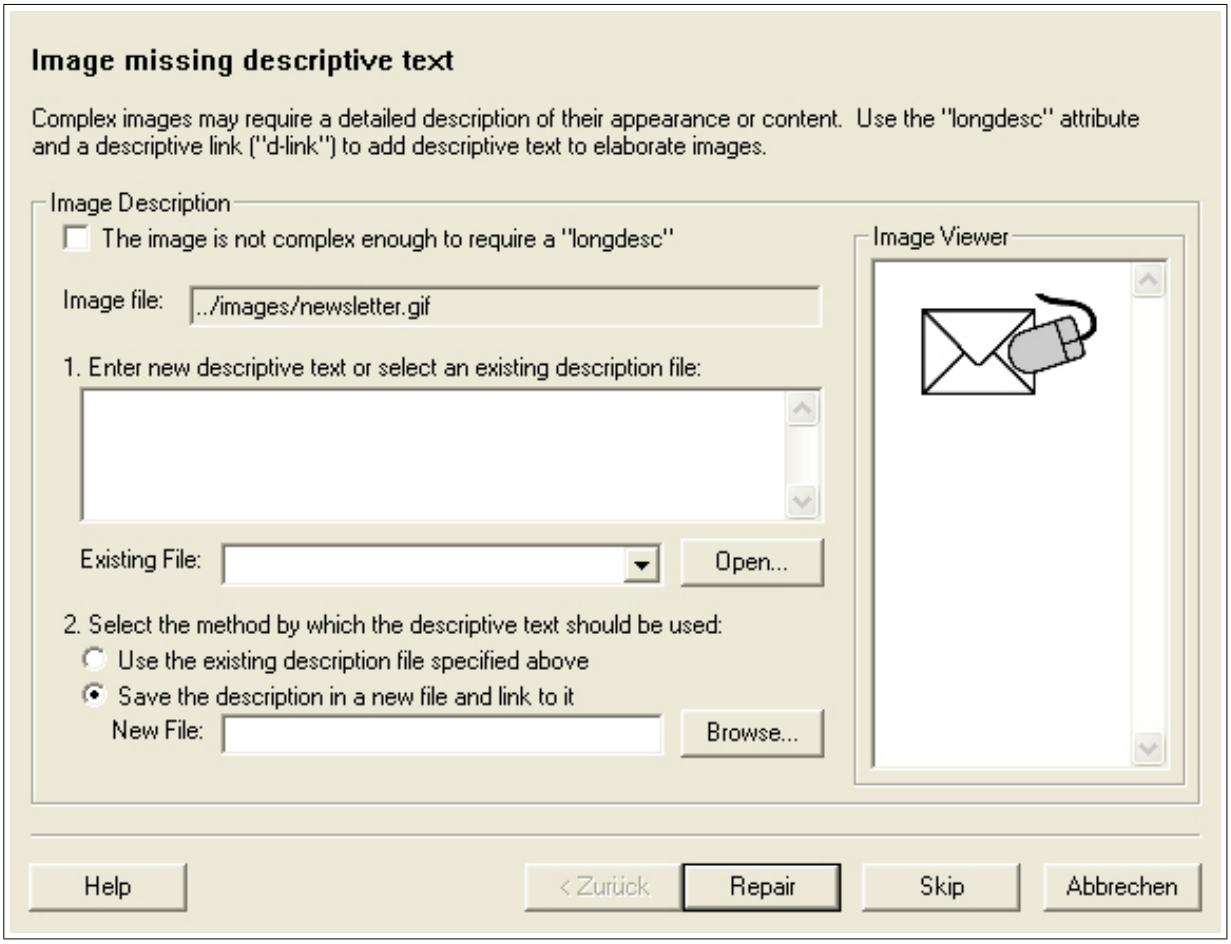


Abbildung 12.8.: Dialogbox zur Eingabe von Beschreibungstexten.

12.3. Browser

Es gibt eine Vielzahl von Browsern, die entweder für das Design barrierefreier Webseiten oder für Benutzer mit speziellen Bedürfnissen konzipiert sind. Diese eignen sich natürlich besonders gut zum Erstellen bzw. zur Kontrolle von Webseiten.

Lynx

Lynx[12] ist der am weitesten verbreitete Textbrowser. Entwickelt von der Universität in Kansas, ist er unter der GNU General Public License verfügbar. Ursprünglich für die Verwendung an Terminals entwickelt, ist der Browser für eine Vielzahl von Plattformen verfügbar.

Lynx ist kein Browser, der extra für die speziellen Bedürfnisse behinderter Menschen entwickelt wurde, ist aber durch sein Textinterface und die für die Tastatur ausgelegte Bedienung, z.B. für die Verwendung in Kombination mit einer Braillezeile gut geeignet.

Für die Entwicklung barrierefreier Internetseiten hilft ein Textbrowser – wie bereits in den Kapiteln zuvor gezeigt – Probleme und Fehler in Webseiten zu finden. Jede barrierefreie Webseite muss auch ohne Einschränkung mit einem Textbrowser verwendet werden können. Fehlende `alt`-Attribute, schlechte Tabellenstrukturen und auf Mausinteraktion basierende Seiten sind Probleme, die bei der Verwendung von Lynx auftreten und dadurch am besten durch die Verwendung eines Textbrowsers aufgedeckt werden.

Opera

Der Opera[13] Browser ist ein graphischer Browser, der einige hilfreiche Funktionen für Webdesign im Allgemeinen, und barrierefreies Webdesign im Speziellen zur Verfügung stellt:

- Opera zeichnet sich durch ziemlich striktes Einhalten der HTML Spezifikationen aus.
- Er ermöglicht über eine Toolbar oder die Taste G einfaches Ein- und Ausschalten von Bildern.
- Über ein Quickmenü ist einfaches Ein- und Ausschalten von Javascript oder Cookies möglich.
- `<link>`-Zusammenhänge von Webseiten können als Navigationshilfe angezeigt werden (Kapitel 5 „Layout & Navigation“).
- Vergrößern und Verkleinern der ganzen Webseite samt Bildern und anderen Elementen ist möglich.
- Die Möglichkeit zur Anzeige aller Links einer Seite mit den entsprechenden `title`-Attributen wird geboten.

[12] `http://lynx.browser.org/`
[13] `http://www.opera.com/`

Eine weitere angenehme Funktion dieses Browsers ist der „User Modus“. Damit können z.B. ein Textbrowser emuliert, die Seite in schwarz/weiß angezeigt, Tabellen deaktiviert, oder, wie in Abbildung 12.9 gezeigt, strukturierende Tags und Elementbegrenzungen angezeigt werden.

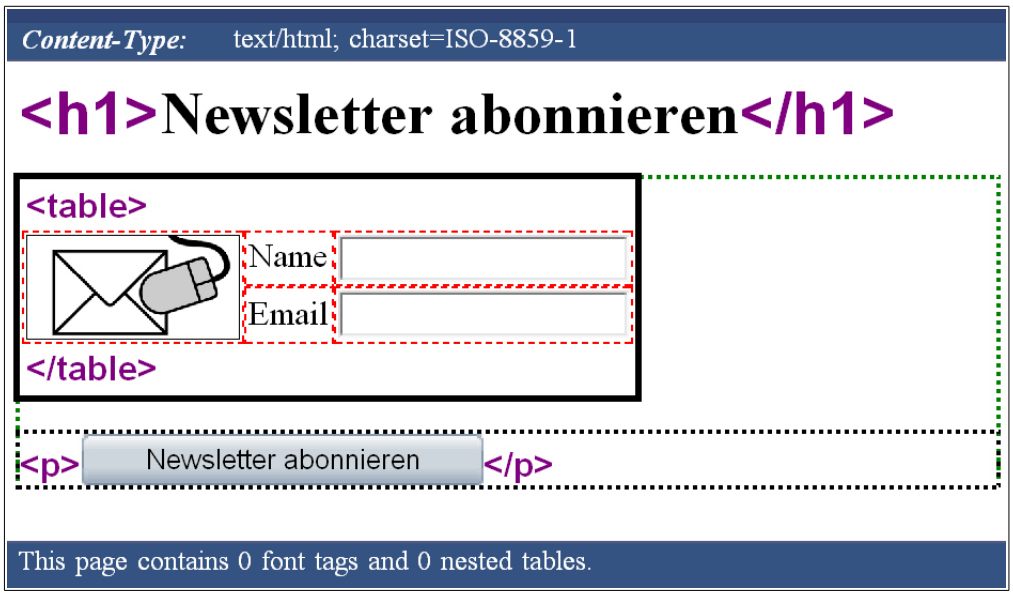


Abbildung 12.9.: Konfigurierbarer User Modus des Browsers Opera.

IBM Home Page Reader und pwWebSpeak

Der IBM Home Page Reader[14] und pwWebSpeak[15] sind Voicebrowser, mit denen sich z.B. blinde Personen Webseiten vorlesen lassen können. Die Erfahrung, einmal mit einem Voicebrowser gearbeitet zu haben, ist für einen Webdesigner sehr lehrreich, denn es hilft die Probleme jener User, die solche Programme zum Surfen verwenden, zu verstehen.

[14] http://www.ibm.com/de/accessibility/hpr.html
[15] http://www.soundlinks.com/pwgen.htm

Teil III.

Anhang

A. Der Onlinekurs auf CD-ROM

Ein wesentlicher Teil dieser Diplomarbeit ist ein Onlinekurs über barrierefreies Webdesign. Er umfasst denselben Inhalt wie das vorhergegangene Skriptum und einen Aufgabenteil mit Lösungsvorschlägen und Erklärungen. Der Kurs ist einerseits unter `http://www.rettinger.cc/bfwd` online zu finden und andererseits dieser Arbeit auf einer CD beigefügt.

Der Onlinekurs eignet sich besonders gut für das Selbststudium, da er sich in dem Medium befindet, das behandelt wird. Die einzelnen Beispiele können somit live ausprobiert und mit den neu vermittelten Techniken kann experimentiert werden.

Als Ergänzung zur Theorie dienen im Onlinekurs Aufgaben, anhand derer der vermittelte Stoff ausprobiert und eingeübt werden kann. Zur Kontrolle der eigenen Arbeit liegen dazu auch Musterlösungen vor. Im Folgenden wird der Onlinekurs in einigen Bildern vorgestellt, die seine Hauptteile repräsentieren sollen.

Auf Abbildung A.1 ist ein Screenshot des Onlinekurses zu sehen. Sie zeigt einen Ausschnitt aus Kapitel 8 „Formulare“. Zu allen Abbildungen ist zumindest der im Zusammenhang relevante Ausschnitt aus dem Sourcecode abgebildet, mit dem Link unter dem Sourcecode kommt man auf eine eigene Seite, die das Beispiel enthält, und kann sich so auch den gesamten Sourcecode ansehen und gegebenenfalls damit experimentieren.

Die Aufgaben, die sich am Ende des Kurses befinden, behandeln alle die eigens entworfene Webseite des “Fiktiven Instituts für Barrierefreies Webdesign“. Die dazugehörige Startseite ist auf Abbildung A.2 zu sehen. Sie wird im Zuge von fünf Aufgaben schrittweise barrierefrei umgestaltet und erweitert.

Auf einer eigenen Seite werden den Lesern die Aufgaben näher gebracht. Diese bestehen aus mehreren Schritten, die sich auf die im Kurs behandelten Themen beziehen. Für jede Aufgabe wird eine eigene Version der Webseite zur Verfügung gestellt, die verbessert werden soll. Abbildung A.3 zeigt einen Ausschnitt der Aufgabenseite.

Für jede Aufgabe wird eine Musterlösung vorgestellt, die einen der möglichen Lösungswege darstellt. Zu jeder Musterlösung gibt es eine Seite, die die vorgenommenen Änderungen erklärt und theoretisch untermauert. Abbildung A.4 zeigt die Erklärung zur Musterlösung der Aufgabe 3.

Titles

Da es nur ein Label für jedes Formularfeld geben darf, braucht es eine andere Strategie um Eingabefelder mit mehr als einer Beschriftung zu handhaben. Ein Beispiel dafür ist in Abbildung 8.9 zu sehen.

Zweidimensionale Anordnung

	Vorname	Nachname	Alter
Spieler 1			
Spieler 2			
Spieler 3			

Abbildung 8.9: Eine zweidimensionale Anordnung von Formelementen.

Als Alternative zu Labels kann für mehrdimensionale Anordnungen von Formularelementen das `title`-Attribut verwendet werden, Abbildung 8.10 zeigt ein Beispiel, bei dem `title`-Attribute zur näheren Bestimmung von Eingabefeldern eingesetzt wurden.

```
<tr>
   <th>Spieler 1</th>
   <td><input type="text" title="Vorname Spieler 1" name="firstname1">
   <td><input type="text" title="Nachname Spieler 1" name="lastname1">
   <td><input type="text" title="Alter Spieler 1" name="age1" size="5">
</tr>
<tr>
   <th>Spieler 2</th>
   <td><input type="text" title="Vorname Spieler 2" name="firstname2">
   <td><input type="text" title="Nachname Spieler 2" name="lastname2">
   <td><input type="text" title="Alter Spieler 2" name="age2" size="5">
</tr>
```

Abbildung 8.10: Verwendung des Title-Attributes für zur Beschriftung von Eingabefeldern in Formularen.

Navigation mit der Tastatur

Die Benutzer von Voice- und Textbrowsern, aber auch Benutzer graphischer Browser, können nur die Tastatur als Eingabegerät verwenden. Daher muss beim Design von Formularen darauf geachtet werden, dass diese auch nur mit der Tastatur verwendbar sind. Das betrifft vor allem Access-Keys, Tab-Reihenfolge und Javascript.

Abbildung A.1.: Skriptum.

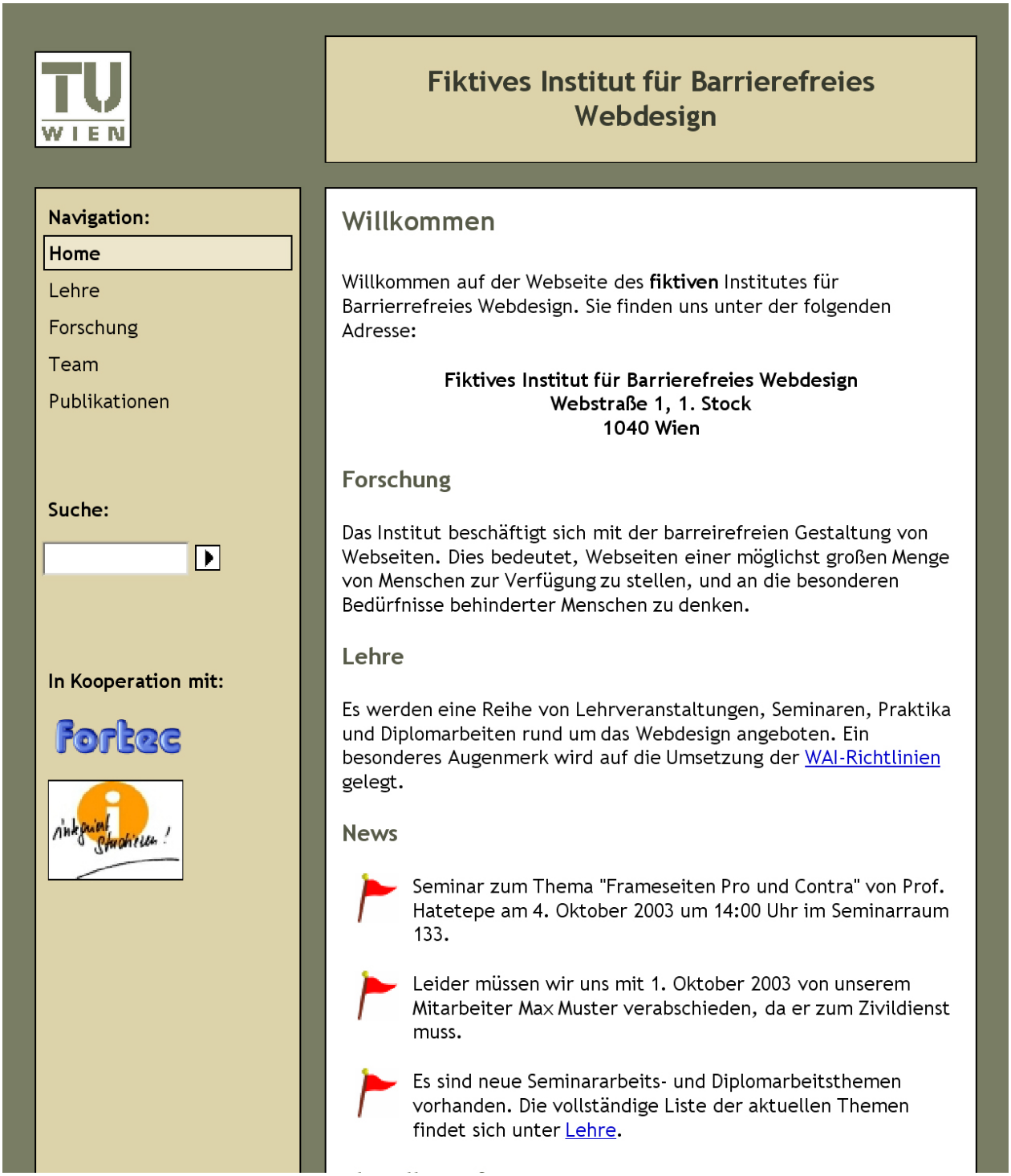


Abbildung A.2.: Institutsseite.

Aufgaben

Einleitung

Im Zuge dieser Aufgaben soll die Webseite des **Fiktiven Institut für Barrierefreies Webdesign** Schritt für Schritt barrierefrei gestaltet werden. Versuchen Sie, in den Aufgaben die Mängel der entsprechenden Version zu finden und zu beheben. Nach jeder Aufgabe gibt es eine Musterlösung als Vorschlag, wie man die jeweiligen Probleme lösen könnte.

Natürlich gibt es zu den einzelnen Aufgaben mehrere mögliche Lösungen, und so ist die Musterlösung vor allem als Anregung zu verstehen.

Um die Aufgaben überschaubar zu halten, ist die Funktionalität der Musterseite eingeschränkt. Daher führen manche Links nicht auf die angegebenen Seiten, sondern wieder zurück auf die Hauptseite. Dynamische Funktionalitäten wie die Suche sind auch nicht implementiert. Der Umfang der Seite nimmt im Laufe der Aufgaben zu und gegen Ende ist die Seite beinahe komplett.

Die kompletten Aufgabenseiten sind als **zip-Datei** verfügbar und können zum einfacheren Editieren heruntergeladen werden.

Aufgabe 1

Schritt 1: Wie in Kapitel 1 erklärt wurde, soll jede Webseite HTML-konform sein. Machen Sie die notwendigen Änderungen, und überprüfen Sie die HTML-Konfirmität mit Hilfe des **W3C HTML Validation Services**.

Schritt 2: Achten Sie im Besonderen auf den richtigen Einsatz von Techniken im Bezug auf Graphiken, die in Kapitel 2 vorgestellt wurden.

Ausgangsversion	**Startversion**
Detailseiten	Home
Stoff	**Kapitel 1: Einführung**, **Kapitel 2: Äquivalente Text-Information**
Musterlösung	**Musterlösung zur Aufgabe 1**
Kommentare	**Kommentare zur Musterlösung zu Aufgabe 1**

Abbildung A.3.: Aufgabenseite.

Erklärungen zur Musterlösung zu Aufgabe 3

Um die **Ausgangsversion** der **Aufgabe 3** zu verbessern, sind die im Folgenden erläuterten Änderungen notwendig. Sie können sich entweder die **Musterlösung** direkt ansehen, oder alle Aufgaben und Lösungen als **zip-Datei** herunterladen.

Schritt 1

Tabellenstruktur: Das Ziel der folgenden Änderungen ist, dass der Haupttext in linearisierter Form, wie sie von Textbrowsern wie zum Beispiel Lynx angezeigt wird, vor der Navigation erscheint. Dadurch muss der Text der linken Spalte nicht auf jeder Seite erneut gelesen werden.

Um dies zu erreichen, muss die linke Tabellenzelle nach der rechten Zelle im Code stehen. Das bedeutet, dass man in der linken Spalte eine neue Zelle hinzufügt, die genau 1px hoch ist, und darunter (und somit in der Linearisierungsreihenfolge hinter der Hauptzelle) die Navigation erscheint. Die Hauptzelle erstreckt sich somit über zwei Tabellenzeilen.

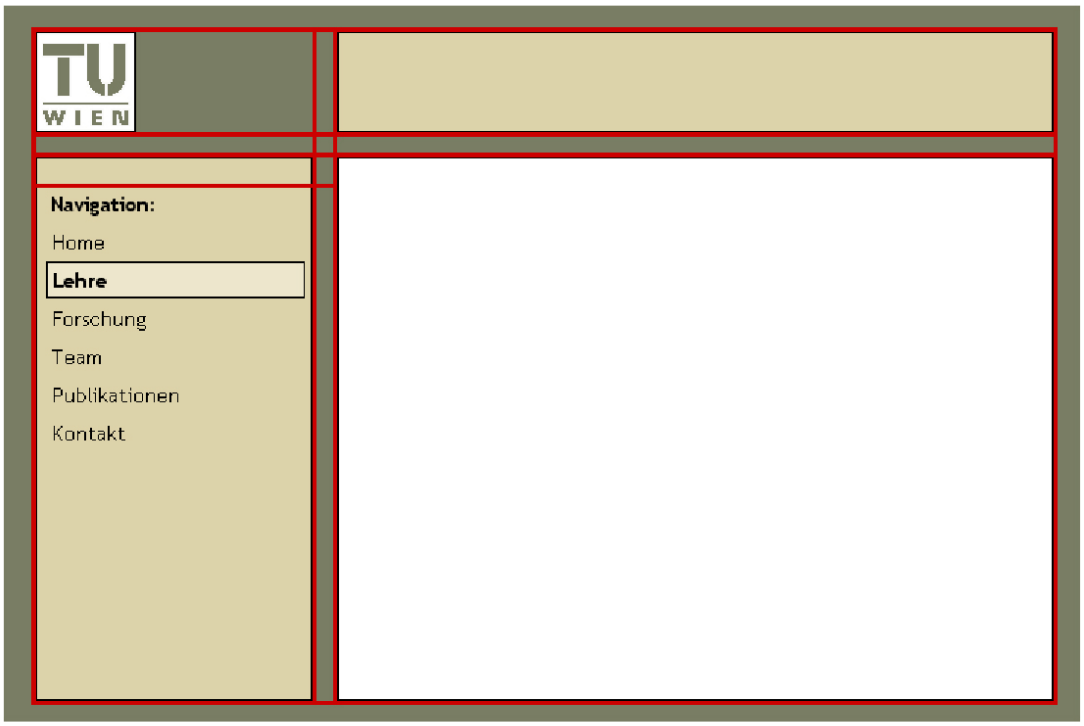


Abbildung A.4.: Kommentar zur Musterlösung zu Aufgabe 3.