Du kannst nicht vorbei…!

Barrierefreie Websites – Vortrag vom 01.04.2004
auf der PHP usergroup Hannover

Von Ulrich Hacke und David Maciejewski

Inhaltsverzeichnis

1 Einleitung

„Barrierefreie Websites" oder „Entwicklung eines barrierefreien Kommunikationsdesigns im Internet" – das sind Phrasen, die uns seit Anfang 2003 immer wieder im Netz-Alltag begegnen. Was genau darunter zu verstehen ist (sowohl in technischer als auch in gesellschaftlich-soziologischer Hinsicht), darüber gibt das vorliegende Dokument in Form eines Vortrages in der hannoverschen PHP usergroup Aufschluss.

Als Leser (oder Hörer) angesprochen sind Menschen, die schon mehr als die ersten Gehversuche im Internet unternommen haben. Gemeint sind vor allem Webmaster, Programmierer, Projektmanager und Entscheider in Webprojekten. Sie alle werden früher oder später mit der Barrierefreiheit in Berührung kommen. Der Vortrag beginnt mit einer einleitenden Betrachtung über Sinn und Zweck des Ganzen und vertieft dies mit unfangreichen Informationen über gesellschaftlich-juristische sowie technische Hintergründe. Im Bereich Technik werden HTML und CSS vorgestellt und es wird auf die Besonderheiten eingegangen, die Webdesigner bedenken müssen, wenn Ihre Werke auch für sehschwache Menschen nutzbar sein sollen. Kapitel 4 umfasst mehrer Beispiel und soll als Anreiz dienen, selber in der Materie nachzuforschen und eigene Lösungen zu finden. Abgerundet wird der technische Part mit Kapitel 0; hier werden verfahren zu technischen Validierung von Websites beschrieben. Alle weiterführenden Informationen, die nicht unmittelbar zum Verständnis des Vortrages nötig sind, wurden im Anhang zusammengestellt.

1.1 Aller Anfang ist schwer. Oder doch nicht?

Eines ist sicher: Barrierefreie Internetseiten kann niemand auf Anhieb ohne Probleme umsetzen. Das heißt aber auch nicht, dass die Programmierung nur einem elitären Kreis vorbehalten ist.

1.1.1 Warum ist Barrierefreiheit für uns PHP-Leute wichtig?

Die Frage ist sehr leicht zu beantworten: mit Hilfe von PHP werden (dynamische) Webseiten in HTML/XHTML erzeugt (Hintergrundinformationen zur Entwicklung des Webs zum HTML-basierten Informationsmedium finden sich im Anhang im Kapitel 7.1). Inzwischen liegt PHP mit über 15 Millionen Installationen (Quelle: php.net) weltweit mit an der Spitze der erfolgreichsten scriptbasierten Websprachen überhaupt. Im Klartext: eine unübersehbare Anzahl von Webseiten im globalen Netz wird durch PHP-Code zur Laufzeit erstellt (sprich: gerendert). Da auch eine große Anzahl an "major sites" (Portale, internationale Firmen etc.) auf PHP setzen, ist es für PHP-Programmierer wichtig, sich mit den Grundlagen der Barrierefreiheit auszukennen.

1.1.2 Was heißt das eigentlich: Barrierefreiheit?

Für Blinde und Sehbehinderte soll die moderne Welt nicht am Bildschirm aufhören - Blindenschrift für den Computer heißt das Stichwort; denn das Internet eröffnet neue Dimensionen. Ein Lesegerät für Blinde übersetzt Buchstaben von Computerbildschirmen in Braille, die erhabene Schrift aus sechs Punkten. Der Apparat lässt die Punktschrift an einer rotierenden Walze wie ein Reifenprofil unter dem Finger des Lesers entlang ziehen.

Dennoch ist für viele blinde Menschen die schöne neue Welt des Internets immer noch verschlossen (zumal Braille-Zeilen teuer sind). Akustische Übersetzungsprogramme (Screen-Reader) können längst nicht mit Braille mithalten. Schlimmer noch: Programmierer und Webdesigner tun ihr übriges mit ihren Sites, um Blinden und anderen (Seh)behinderten das Leben so richtig schön schwer zu machen.

Was heißt das? Tabellen beispielsweise, die zur Positionierung von Bildern und Logos verwendet werden, werden vom Lesegerät Zeile für Zeile vorgelesen. Das kann für den User sehr mühsam sein, zumal <img>-Tags nicht gerade informativ sind. Statt Informationen erhält der Blinde lediglich einen unverständlichen Wort- und Buchstabensalat.

Aber es gibt auch eine Reihe anderer Behinderungen, wie die Farbblindheit (vergl. Kapitel 3.4.1), dessen Nachteile man mit sauberer Programmierung minimieren kann.

Die einzige Lösung kann daher nur sein: sämtliche Layouttabellen und allen überflüssigen Code zu verbannen. Damit man genau weiß, was überflüssig ist, gibt es Standards auf die wir hier näher eingehen werden. Die grafische Ausgestaltung der Seiten übernimmt fortan CSS (die nötigen Informationen zur Anwendung stehen in Kapitel 3.3).

1.1.3 Was ändert sich und was bleibt gleich?

Die Arbeits- und Herangehensweise bei der Programmierung von barrierefreien und validen Webseiten wird eine andere sein. Es ist mehr Know How notwendig, logisches Umdenken ist gefragt. Damit solche Webseiten aber überhaupt richtig angezeigt werden können, benötigt man die neueren Browserversionen. Ältere Browser, wie z.B. der immer noch eingesetzte Netscape Navigator 4.7 und andere gehören damit endlich der Vergangenheit an, weil diese schlicht und ergreifend nicht fähig sind, modernere Websites darzustellen.

1.1.4 Mehr Arbeit…?

Wer glaubt, dass mit der Umsetzung von barrierefreien Websites viel Arbeit auf einen zukommt, irrt gewaltig. Die Aussagen mancher großer Agenturen, eigens Taskforces eingerichtet zu haben, um diese Aufgaben bewältigen zu können, ist einzig allein ein Marketingtrick, oder Zeugnis von zu wenig Erfahrung im Umgang mit CSS.

In der Praxis zeigt sich sehr schnell, dass die Programmierung von validen (sprich: „grammatikalisch korrekten") und gleichzeitig barrierefreien Seiten lediglich eine andere Herangehensweise darstellt als bisher. Und ganz nebenbei gesagt: exakt dazu wurden HTML-Standards (vergl. Kapitel 3.1) verabschiedet. Regeln, an die man sich als Webdesigner halten sollte, sind nun wirklich nichts Neues. Vielmehr sollten Designer dankbar sein, dass Browser in der Regel auch mit dem wüstesten Code („HTML-Suppe") bislang fast immer klar gekommen sind.

Im Übrigen gibt es genügend Werkzeuge, welche uns „Web-Leute" bei der Entwicklung barrierefreier Sites unterstützen (mehr dazu in Kapitel 0).

1.1.5 …und außerdem

Seit dem Inkrafttreten des „Gesetzes zur Gleichstellung behinderter Menschen" ist Barrierefreiheit nicht mehr nur eine nette Geste, sondern für öffentliche Anbieter Pflicht. Ab dem 1. August 2002 müssen alle Seiten, die von Bundesbehörden ins Netz gestellt werden, bestimmten Kriterien an die Barrierefreiheit genügen. Entsprechende Gesetze und Verordnungen auf Landes– und kommunaler Ebene sind in Vorbereitung und werden sich an die Vorgaben der Barrierefreie Informationstechnik-Verordnung (BITV) anlehnen. Mehr dazu im Kapitel 2.1)

Die Anzahl behindertengerecht programmierter Websites wird in den nächsten Jahren rapide zunehmen. Noch ist Zeit, sich die Techniken dahinter genauer anzusehen.

1.2 Vorteile, oder: warum das alles?

Welches sind die Top-Argumente, die für eine Umstellung von Websites nach Kriterien der Barrierefreiheit sprechen? Hier eine kurze Liste:

1. Kunden

   Je mehr Menschen die eigene Website betrachten können, desto besser. Eine möglichst hohe Breitenwirksamkeit (nicht zu verwechseln mit Massenkompatibilität) ist unverzichtbar. Website-Besucher sind Kunden, die z.B. Produkte kaufen, sich Meinungen aneignen, Werbung konsumieren, eigene Beiträge leisten und und und…

2. Geschwindigkeit

   Barrierefreie Seiten zeichnen sich fast immer durch schlankeren Code aus und können dadurch schneller übertragen und auf Clientseite gerendert werden. Je schneller eine Site beim Konsumenten zu sehen ist, umso mehr wird er sich darauf umsehen. Seiten, die lange brauchen, bis sie sich aufgebaut haben, steigern die Ungeduld und die Unzufriedenheit – was nicht schnell da ist, wird bald wieder weggeklickt.

3. Geschwindigkeit

   Dies ist ein Argument, das nicht allzu lange Bestand haben wird. Noch sind barrierefreie Seiten (so sie denn als solche ausgezeichnet oder klar erkennbar sind), genießen derzeit noch einen recht hohen „Marktwert", sei es aus technischer oder sei es aus politischer Sicht. Wer eine gut gestaltete barrierefreie Site sein eigen nennt und das entsprechend kommuniziert, kann mit guten Besucherzahlen rechnen. Und ganz nebenbei kann man eine Vorreiterrolle sprechen. Für Firmen, die im Web ihre Geschäfte machen oder für Webagenturen kann es eigentlich gar kein besseres Argument geben. Aber Vorsicht: eine Website wird nicht allein per definitionem barrierefrei!

4. Plattformunabhängigkeit

   Viele Webdesigner entwickeln ihre Sites für genau einen Browser auf genau einem Betriebssystem. Wer mit einer anderen Konfiguration vorbeischaut, hat das Nachsehen. Barrierefreie und standardisierte Sites dagegen werden auf allen gängigen Browsern korrekt dargestellt. Natürlich gibt es immer Unterschiede – aber das geht vollkommen in Ordnung. Eine Bach-Sinfonie klingt auch unterschiedlich, wenn man sie auf einem kleinen Küchenradio, einer HiEnd-Anlage oder live im Konzertsaal hört, obwohl die zugrunde liegenden Noten doch immer dieselben sind.

5. Flexibilität

   Was tun, wenn eine Site auf ganz unterschiedlichen Ausgabemedien angezeigt werden soll, z.B. einem Handy oder einem PDA anstelle des Arbeitsplatzrechners? Eine barrierefreie Site ist auch dafür gerüstet und passt sich selbständig an die Begebenheiten des Ausgabegerätes an. Und durch den schlanken Code sind Weiterentwicklungen viel schneller umgesetzt, als wenn man sich erst durch einen ellenlangen spaghettiartigen HTML-Code wühlen muss.

2 Grau ist alle Theorie: Die Vorschriften

2.1 Die BITV

2.1.1 Hintergründe

Am ersten Mai 2002 ist in Deutschland das Behindertengleichstellungsgesetz (BGG) in Kraft getreten. Ziel ist es, „die Benachteiligung von behinderten Menschen zu beseitigen und zu verhindern sowie die gleichberechtigte Teilhabe von behinderten Menschen am Leben in der Gesellschaft zu gewährleisten und ihnen eine selbstbestimmte Lebensführung zu ermöglichen." (BGG, §1).

Damit bot das BGG quasi die Grundlage für die zwei Monate später vom Bundesinnenministerium und dem Bundesministerium für Arbeit und Soziales verabschiedete Barrierefreie Informationstechnik-Verordnung (BITV) dar. In ihr ist näher geregelt, welche Bedingungen für eine barrierefreie Website zu erfüllen sind, bis wann das umzusetzen ist und für wen diese Verordnung zunächst gilt – zunächst nämlich für alle Internetauftritte und –angebote sowie „mittels Informationstechnik realisierte graphische Programmoberflächen, die öffentlich zugänglich sind", der Behörden der Bundesverwaltung. Die dafür nö-

tigen Umbauarbeiten der betroffenen Informationsangebote sollen bis zum 31. Dezember 2004 abgeschlossen sein.

Inhaltlich basieren die Anforderungen und Bedingungen der BITV grundsätzlich auf den Zugänglichkeitsrichtlinien für Web-Inhalte 1.0 (Web Content Accessibility Guidelines 1.0) des World Wide Web Consortiums (W3C) vom 5. Mai 1999.

2.1.2 Inhalte

Damit eine Website der BITV entspricht, muss sie jede der 14 Anforderungen erfüllen:

1. Für jeden Audio– oder visuellen Inhalt sind geeignete äquivalente Inhalte bereitzustellen, die den gleichen Zweck oder die gleiche Funktion wie der originäre Inhalt erfüllen.
2. Texte und Grafiken müssen auch dann verständlich sein, wenn sie ohne Farbe betrachtet werden.
3. Markup–Sprachen (insbesondere HTML) und Stylesheets sind entsprechend ihrer Spezifikationen und formalen Definitionen zu verwenden.
4. Sprachliche Besonderheiten wie Wechsel der Sprache oder Abkürzungen sind erkennbar zu machen.
5. Tabellen sind mittels der vorgesehenen Elemente der verwendeten Markup–Sprache zu beschreiben und in der Regel nur zur Darstellung tabellarischer Daten zu verwenden.
6. Internetangebote müssen auch dann nutzbar sein, wenn der verwendete Benutzeragent neuere Technologien nicht unterstützt oder diese deaktiviert sind.
7. Zeitgesteuerte Änderungen des Inhalts müssen durch die Nutzerin / den Nutzer kontrollierbar sein.
8. Die direkte Zugänglichkeit der in Internetangeboten eingebetteten Benutzerschnittstellen ist sicherzustellen.
9. Internetangebote sind so zu gestalten, dass Funktionen unabhängig vom Eingabegerät oder Ausgabegerät nutzbar sind.
10. Die Verwendbarkeit von nicht mehr dem jeweils aktuellen Stand der Technik entsprechenden assistiven Technologien und Browsern ist sicherzustellen, so weit der hiermit verbundene Aufwand nicht unverhältnismäßig ist.
11. Die zur Erstellung des Internetangebots verwendeten Technologien sollen öffentlich zugänglich und vollständig dokumentiert sein, wie z.B. die vom World Wide Web Consortium entwickelten Technologien.
12. Der Nutzerin / dem Nutzer sind Informationen zum Kontext und zur Orientierung bereitzustellen.
13. Navigationsmechanismen sind übersichtlich und schlüssig zu gestalten.
14. Das allgemeine Verständnis der angebotenen Inhalte ist durch angemessene Maßnahmen zu fördern.

2.1.3 Standards

Die BITV schreibt die Nutzung von Technologien vor, wie die vom W3C. Mit dieser „Öffnungsklausel" schließt die BITV auch andere De–Facto–Standards ein, die nicht vom W3C kontrolliert werden: JavaScript ist als ECMA–262 vollständig dokumentiert und normiert worden. Auch das Flash–Format ist vom Hersteller Macromedia vor einiger Zeit offen gelegt und dokumentiert worden, so dass es hier wohl ebenfalls eingeschlossen ist.

Aus dieser Anforderung folgt unter anderem auch, dass der Programmierer bei seinen Seiten auf proprietäres, d.h. herstellerspezifisches Markup grundsätzlich verzichten soll-

te. Allerdings ist es für eine Verordnung aus formaljuristischen Gründen nicht möglich, sich direkt auf Standards zu beziehen, die nicht der regierungsamtlichen Kontrolle unterliegen. Grundsätzlich fährt man dann genau richtig, wenn man sich an die Standards des W3C hält. Das umfasst (X)HTML und CSS. Validatoren des W3C helfen bei der Entwicklung und Kontrolle (vergl. Kapitel 0).

3 Grundlagen

3.1 HTML-Standards

Bei der Programmierung von Websites sollte man sich stets an Standards halten, um gewährleisten zu können, dass die meisten Browser und Medien mit der neuen Seite auf Anhieb klarkommen.

3.1.1 Die richtige Wahl des Doctypes

Damit Software in der Lage ist, ein Dokument korrekt darzustellen, muss sie wissen, um was für eine Art von Dokument es sich handelt. Dazu hat das WWW-Konsortium (W3C) in den HTML-Spezifikationen die Regel aufgenommen, dass jedem Dokument eine "doctype"-Angabe voranzustellen ist, die Aufschluss über die verwendete Dokumentensprache und Version gibt. Die "doctype"-Angabe verweist dabei auf eine "Document Type Definition" (DTD) und steht immer am Anfang eines jeden Dokumentes.

Ein gültiger Doctype mit voller Angabe der URI (universal resource identifier, die komplette Webadresse) weist die Browser an, die Webseite im Standard-Modus anzuzeigen, also das (X)HTML, CSS und DOM (document object model) wie erwartet umzusetzen.

Ein fehlender, veralteter oder ohne URI versehener Doctype weist die Browser hingegen an, die Webseite im sogenannten „Quirks"-Modus anzuzeigen. Der Browser nimmt dann also an, dass es sich um altmodischen, ungültigen Code handelt (wie in den späten Neunzigern fast generell üblich). So wird der Browser die Webseite in rückwärtskompatiblem Modus rendern und das CSS so umsetzen wie im IE4 üblich und sich auf proprietäres DOM stützen. Das ist natürlich nicht das Ziel.

Die Standards HTML 4.01 und XHTML 1.0 sehen drei DTDs vor: „Strict" umfasst den „reinen" Standard. „Transitional" bezieht aber auch die als „veraltet" - aber noch erlaubt - geltenden Elemente mit ein. „Frameset" ist das DTD für „Transitional" in Dokumenten, die Frames benutzen. Der XHTML 1.1-Standard umfasst lediglich „Strict".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//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">
```

Abb. 1 HTML 4.01 Strict, Transitional, Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Abb. 2 XHTML 1.0 Strict, Transitional, Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Abb. 3 XHTML 1.1

3.1.2 Valide Websites

Barrierefreie Websites sind grundsätzlich immer valide, entsprechen also den seitens des W3C definierten Standards und sind fehlerfrei ("grammatikalisch korrekt"). Anders gesagt: Achtet man bei der Programmierung auf die Validität hat man schon den Grundstein für eine barrierefreie Website gelegt.

Zur Validierung von Webseiten stellt der W3C kostenlose Onlineformulare zur Verfügung. Jeder gefundene Fehler wird inklusive Beschreibung explizit aufgeführt und kann so im Nu entfernt werden.

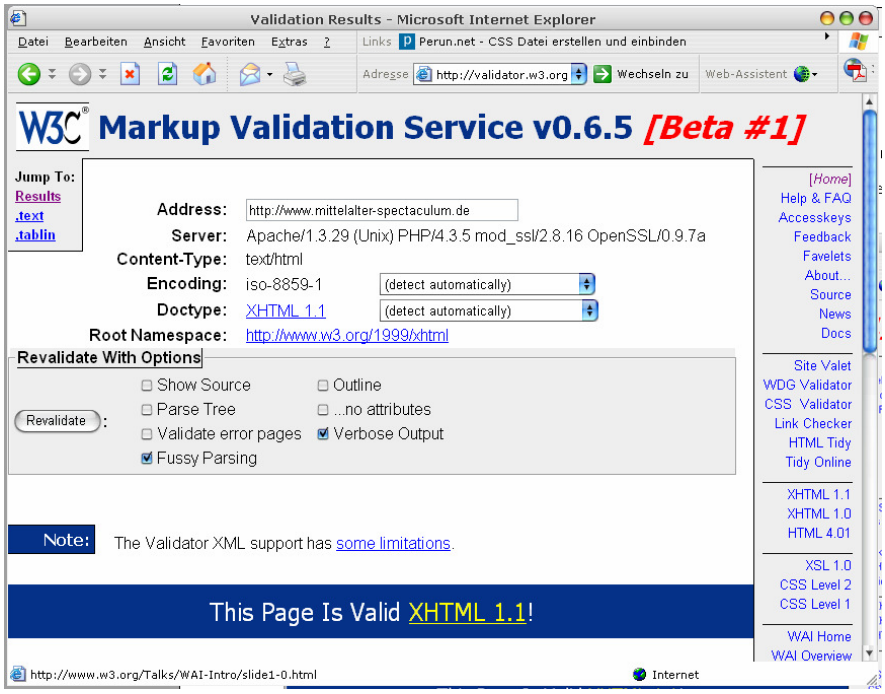

Abb. 4 Markup Validator des W3C nach erfolgreicher Prüfung einer Internetseite

3.2 Browser

Jedem dürfte noch der wehleidige Browserkrieg in den Ohren liegen, wo jeder seinen eigenen Browser favorisierte – einen (historischen Rückblick haben wir im Anhang in Kapitel 7.2 zusammengestellt). Damals gab es nicht wirklich triftige Argumente für oder gegen einen Mitbewerber. Nun scheint es so, dass der alte Kaffee erneut angewärmt wird, doch so ist es nicht. Dank fest definierten Regelwerken gibt es nur einen Maßstab für einen guten Browser: Entweder dieser kann die Standards korrekt umsetzen oder eben nicht. Doch die heutigen Browsergenerationen halten sich nicht vollständig an die Regeln, was zwangsläufig immer noch zu Darstellungsunterschieden führt.

Dazu ein einfaches – aber folgenschweres - Beispiel: Das Regelwerk des W3C für HTML und XHTML sagt aus, dass ein „padding" (innerer Abstand des Inhaltes zum Container, z.B. der Text innerhalb eines Paragraphen) und „margin" (äußerer Abstand vom Container) außerhalb des Innenelementes (hier im Beispiel der Text zum Paragraphen) liegen. Mozilla interpretiert dies richtig, der Internet Explorer 6 nicht, was zwangsläufig dazu führt, dass beide Browser unterschiedliche Darstellungen liefern.

Doch auch nicht alle Regeln sind logisch. Setzt der Programmierer in einem Paragraphen ein padding von 2px und einen margin von 5px, so ist ein Element von einer festen Breite von 100% genau 100% + 4px (für beide paddings, links und rechts) + 10 px für die margins breit. Printdesigner und Layouter werden sich hier die Haare raufen. Dies sollte dennoch niemanden davon abhalten, sich an die Standards zu halten, denn gerade bei behindertengerechten Websites steht nicht das Design, sondern die Nutzbarkeit im Vordergrund. Gute Websites zeichnen sich dadurch aus, wenn beides Berücksichtigung gefunden hat.

Hier eine Liste der aktuellen Browser, die mehr oder weniger die erforderlichen Techniken unterstützen:

- Internet Explorer 6
- Mozilla 1.6
- Firefox 0.8 (abgespeckte Version Mozilla)
- Opera 7
- Netscape 7

3.3 Webdesign mit CSS

Auf den ersten Blick scheint CSS nicht allzu viel mit Barrierefreiheit zu tun zu haben – geht es doch zunächst um validen HTML-Code. Stellt man aber die Trennung von Layout und Inhalt in den Vordergrund, dann erscheint CSS gleich in einem ganz anderen Licht. Während die HTML-Codeblöcke für ein stabiles Grundgerüst sorgen, ist CSS für das Erscheinungsbild der Inhalte zuständig.

Ein sehr passende (wenn auch flapsige Beschreibung) der technischen Herangehensweise findet sich in einem Artikel von Joe Clark: er verwendet die Metapher des Ingenieurs namens HTML, der mit einer angesehenen und trendigen Innenausstatterin verheiratet ist: CSS. Der Ingenieur zimmert Stützbalken so zusammen, dass niemand auf abscheuliche Weise beim Betreten eines Gebäudes ums Leben kommt, sobald man aber anfängt, über Gardinen zu reden, schickt man ihn am besten in den Keller, wo er mit seiner Modelleisenbahn spielen kann. Für die wirklich wichtigen Dinge lässt man besser die Innenausstatterin ans Werk. Kurz: nachdem Skelett und Muskeln einer Website (HTML) stehen, kann man diesen „Körper" (nicht umsonst haben verwenden wir das <body>-Tag) in jeden glamourösen, glitzernden und tiefausgeschnittenen Fummel stecken, den man sich vorstellen kann.

3.3.1 Wie wendet man CSS an?

CSS kann man entweder im Kopf einer (X)HTML-Datei platzieren, in eine externe Datei auslagern (empfohlen) oder in den Tags integrieren (embedded CSS).

```
<head>
  <title>Untitled</title>
  <style type="text/css">
    p.text { color: #f00; }
  </style>
</head>
```

Abb. 5 CSS-Definitonen im Kopf einer Seite

```
<head>
    <title>Untitled</title>
    <link rel="stylesheet" type="text/css" href="style.handheld.css">
</head>
```

Abb. 6 Auslagerung der CSS-Definitionen in eine externe Datei

```
<p style="color: #f00;">
Das ist mein Text
</p>
```

Abb. 7 embedded CSS

Eigene Stylesheets zu erstellen ist simpel: man braucht eigentlich nur ein paar grundlegende HTML-Kenntnisse, um loszulegen. Ein Beispiel: um die Textfarbe einer Überschrift 1. Ordnung <h1> festzulegen, benutzt man diese Regel:

```
h1 { color: red; }
```

Abb. 8 CSS-Beispiel: Überschrift farbig formatieren

Jede CSS-Regel besteht aus zwei Hauptteilen, dem Selektor (h1) und der Deklaration (color:red). Die Deklaration besteht ebenfalls aus zwei Teilen, der Eigenschaft „color" und dem Wert „red". Daraus ergibt sich folgende Syntax:

```
SELEKTOR { Eigenschaft: Wert; }
```

Abb. 9 Grundlegende Syntax für CSS-Regeln

Der Selektor h1 ist die Verbindung zwischen HTML und dem Stylesheet, und alle gültigen HTML Elemente sind mögliche Selektoren.

Die Eigenschaft 'color' ist nur eine von vielen Eigenschaften, mit denen man das Aussehen des HTML Dokumentes beeinflussen kann.

Um Styles Sheets möglichst klein zu halten, kann man Selektoren in Gruppen zusammenfasen, indem man sie mit Kommata vor der Deklaration trennt:

```
h1, h2, h3, div { font-family: Arial; }
```

Abb. 10 Selektoren zusammenfassen

Genau so kann man auch mehrere Eigenschaften zusammenfassen:

```
h1 {
  font-weight: bold;
  font-size: 1em;
  line-height: 105%;
  font-family: Arial;
}
```

Abb. 11 Eigenschaften zusammenfassen

Zusätzlich haben einige Eigenschaften ihre eigene Gruppierungssyntax:

```
h1 { font: bold 12px/14px Arial }
```

Abb. 12 Eigenschaften gruppieren

Dieses Beispiel ist mit dem oberen identisch.

Um den Einsatz von Styles möglichst flexibel zu gestalten, gibt es das HTML-Attribut `class`. Für alle Elemente innerhalb von `body` kann das Attribut definiert werden, und die Klasse (class) kann im Stylesheet deklariert werden:

```
.gruen { color: green }
```

Abb. 13 Klassen deklarieren

Das HTML Attribut „id" kann ebenfalls als Selektor verwendet werden. Der Unterschied zu „class" besteht darin, das ein „id" ein Unikat in dem Dokument sein muss, also jeder Wert für „id" darf nur einmal auftauchen. Auf diese Wiese ausgezeichnete Elemente er-

halten daher eine besondere Bedeutung. Deklariert werden „id"-Selektoren mit einem vorangestelltem „#"

```
#wide { letter-spacing: 0.3em }
h1#wide { letter-spacing: 0.5em }
```

Abb. 14 Elemente mit ID auszeichnen

In (X)HTML sieht die Anwendung der obigen Beispiele dann so aus:

```
<div id="wide"><p class="green">Das ist mein Text</p></div>
```

Abb. 15 Codebeispiel in (X)HTML

CSS-Anweisungen kann man sehr gut dafür nutzen, um immer wiederkehrende Elemente aus dem (X)HTML-Dokument zu verbannen. Das bei image-Tags übliche `border="0"` kann in CSS so definiert werden:

```
img { border: 0px; }
```

Abb. 16 Wiederkehrende Eigenschaft per CSS definieren

Auch Farbdefinition RGB (rot, grün, blau) mit `color` kann man mit CSS abkürzen:

```
color: #f00; /* ist das gleiche wie: */ color: #ff0000;
color: #000; /* ist das gleiche wie: */ color: #000000;
color: #a37; /* ist das gleiche wie: */ color: #aa3377;
```

Abb. 17 Farbwerte abkürzen

Bekanntlich besteht eine Farbdefinition in (X)HTML aus drei hexadezimal codierten Bytewerten, die für die Farbanteile rot, grün und blau stehen (RGB) und umfasst daher sechs Zeichen. Kürzt man eine Farbe auf drei Zeichen ab, interpretiert der Browser diese Zeichen als den höherwertigen Anteil (Nibble) jedes Farbbytes und das niederwertige Nibble wird diesem gleichgesetzt. Daher lassen sich nur solche Farbwerte abkürzen, deren Farbbytes aus gleich großen Nibbles bestehen.

3.3.2 Von Boxen und fließenden Containern

Haben Programmierer und Webdesigner bisher gestalterische Merkmale wie Logos oder Bilder per Tabelle entsprechend auf der Webseite platziert, kann man mit CSS durchaus effizientere Methoden anwenden, die zudem bei richtiger Anwendung auch valide sind.

Angst um die lieb gewonnenen Tabellen braucht aber niemand zu haben. In Form von reinen Datentabellen bleiben sie uns erhalten. Alle anderen Tabellen fliegen raus und werden beispielsweise durch DIV-Container ersetzt. Abstriche im Design braucht man ebenfalls nicht zu machen – auch komplexe Webseiten lassen sich mit dieser Technik problemlos umsetzen. Container lassen sich durch die Nutzung der `float`-Eigenschaft fließend nebeneinander platzieren:

```
<div style="float: left; width: 100px;">Der 1. Container</div>
<div style="float: left;">Dieser Container erschein auf Grund von
float direkt neben dem 1. Container</div>
<br style="clear: both;" /> /* Erzwingt einen Umbruch für den
nachfolgenden Inhalt*/
```

Abb. 18 Zwei -Container nebeneinander

Container lassen sich auch absolut, also pixelgenau platzieren. Als Referenz dient entweder der Seitenrahmen (`position: absolute;`) oder das Mutterelement (`position: relative;`). Mit dem z-index kann man die Reihenfolge bei Überlappungen von anderen Containern bestimmen. Ein Beispiel:

```
<div style="position: absolute; top: 50px; left: 130px; z-index:
5;">Mein positionierter Container</div>
```

Abb. 19 Container platzieren

In unseren Beispiel einer Miniwebsite zeigen wir die Anwendung solcher Container. (siehe Kapitel 4.1).

3.3.3 Semantisch statt visuell

Was ist denn eigentlich der Unterschied zwischen den Tags `<b>bold</b>` im Vergleich zu `<strong>strong</strong>` und `<i>italic</i>` im Vergleich zu `<em>emphasis</em>`?

Zwar werden beide Paare von Tags in den allermeisten Browsern gleich dargestellt: „`bold`“ und „`strong`“ ergeben beide Fettdruck: „italic" und „emphasis" kursive Schrift. Der Unterschied besteht darin, dass es sich bei „`bold`“ und „`italic`“ um Tags zur visuellen Darstellung handelt, während „`strong`“ und „`em`“ hingegen semantische, logische Tags sind. Die visuelle Darstellung legt also lediglich fest, wie ein Browser den so formatierten Text darstellt. Die semantische Darstellung hingegen beinhaltet stattdessen tatsächlich eine Hervorhebung bzw. Betonung - stellt dem Lesegerät oder Browser also exakte Befehle bereit. Die semantische Kennzeichnung mittels „strong" und „emphasis" ist somit immer sinnvoller, als die Kennzeichnung mittels bold und italic.

Anderes Beispiel: Mit den Aural-Definitionen von CSS kann man die Lesefunktionen beeinflussen. Das sieht dann so aus:

```
volume: loud;
speak-numeral: continuous;
```

Abb. 20 Vorleseoptionen kontrollieren

Je genauere Definitionen man anlegt, desto größer ist die Chance, dass der eigene Text genauso verstanden wird, wie er gemeint war.

3.3.4 Unterschiedliche Ausgabemedien

Einer der größten Vorteile von CSS besteht darin festzulegen kann, wie ein Dokument auf unterschiedlichen Ausgabeeinheiten jeweils dargestellt werden soll. Man kann also getrennte Stylesheets für die Bildschirmdarstellung und für die Druckausgabe festlegen.

`all`	Passend für alle Ausgabegeräte.
`aural`	Sprachausgabe des Dokuments
`braille`	Ausgabe auf einem Tastgerät für Blinde
`embossed`	Ausgabe auf einem Drucker für Blindenschrift
`handheld`	Gedacht für Ausgabeeinheiten, die in der Hand gehalten werden (normalerweise kleiner Bildschirm, geringe Bandbreite)
`print`	Ausgabe auf nicht durchsichtigem Material (in der Regel ein normaler Papierausdruck)
`projection`	Projektion, z.B. mit einem Video-Beamer
`screen`	Darstellung auf einem farbigen Computer Bildschirm
`tty`	Ausgabe in einem Terminalfenster, d.h. nur Text verfügbar, keine Grafiken, keine Pixelmaße (z.B. Lynx)
`tv`	Ausgabe auf einem Fernseher (kleine Auflösung, Farbe, nur bedingt scrollfähig, Sound immer verfügbar)

Abb. 21 CSS-Medientypen

Die Einbindung solcher Stylesheets könnte so aussehen:

```
<link rel="stylesheet" type="text/css" media="print"
href="style.print.css" />
<link rel="stylesheet" type="text/css" media="screen"
href="style.screen.css" />
<link rel="stylesheet" type="text/css" media="handheld"
href="style.handheld.css" />
<link rel="stylesheet" type="text/css" media="tty"
href="style.tty.css" />
```

Abb. 22 Mehrere Styles für unterschiedliche Medientypen einbinden

Da ältere Browser (wie der Netscape 4.7) diverse Formatierungen einfach nicht versteht, könnte man Usern eines solchen Browsers ein `<div>` mit einem Warnhinweis auf die alte Technologie einblenden.

```
@import url(„style.basic.css“);
```

Abb. 23 Basis-Style einbinden

Mit der oberen Anweisung importieren wir im screen-Stylesheet unsere Basis-Styles. Dort stellen wir unser `div` mit dem Hinweistext auf „anzeige“:

```
div.hinweis { visibility: visible; display: block; }
```

Abb. 24 Hinweistext für ältere Browser

In allen anderen Ausgabemedien und auch später im „`screen`“ verstecken wir hingegen den Hinweis:

```
div.hinweis { visibility: hidden; display: none; }
```

Abb. 25 Der gleiche Hinweis: diesmal für neuere Browser

Auf diese Weise kann man auch im Medium `screen` aufwändige Grafiken, Logos und Hintergründe anzeigen und diese im Medium `print` ganz einfach ausblenden. So entfallen die bisher aufwändig programmierten „Druckversionen“. Auch dies lässt sich am Beispiel der weiter unten vorgestellten Miniwebsite ausprobieren (siehe Kapitel 4.1).

3.4 Behinderungen und/oder Einschränkungen

Die meisten Menschen denken beim Stichwort „Behinderung“ sofort an körperliche Behinderungen (z.B. den „klassischen“ Rollstuhlfahrer) oder Sehbehinderungen. Dabei gibt es letztendlich eine ganze Reihe von Beeinträchtigungen für den Menschen; ihre Relevanz für die Erstellung barrierefreier Seiten wird jedoch nur für die Sehbehinderungen und die Farbenblindheit detaillierter besprochen (weitergehende Informationen zu Behinderungen sind im Anhang im Kapitel 7.3 aufgeführt).

3.4.1 Farbblindheit

Landläufig werden unter der Bezeichnung „Sehbehinderte“ Blinde verstanden; Webdesigner sollten sich jedoch viel mehr Sorgen um diejenigen Menschen machen, die noch etwas Sehvermögen besitzen und solche, die Probleme mit bestimmten Farbkombinationen haben. Nahezu alle Farbenblinden sehen immer noch Farben – die einzige nennenswerte Ausnahme wird Achromatopsie genannt und betrifft Schätzungen zufolge 0,03% der Weltbevölkerung. Diese Gruppe sieht (soweit sie das selbst beurteilen kann) alles in Graustufen.

Welches sind nun die Farbkombinationen, die bei Farbenblinden für Verwirrungen sorgen? Die klassische Kombination ist Rot und Grün (zum Glück spielt bei Verkehrsampeln auch noch die Position einer Farbe eine Rolle), aber auch viele rötliche bzw. grünliche

Abstufungen bereiten Probleme. Die medizinischen Termini für diese Formen der Fehlsichtigkeit lauten im übrigen Protanopie und Deuteranopie (falls sie in abgeschwächter Form auftreten Protanomalie resp. Deuteranomalie).

Weitaus weniger Menschen können Blau und Grün nicht auseinander halten – hier lautet der Fachausdruck Tritanopie. Eine solche Fehlsichtigkeit kann sich im Laufe der Zeit entwickeln, wohingegen Protanopie und Deuteranopie meistens erblich sind.

Die Zahl der Betroffenen beträgt (in westlichen Ländern) zwischen 4-8% der Bevölkerung. Die meisten Farbenblinden sind Männer.

4 Case studies: von Fall zu Fall

4.1 Unsere kleine Demosite

Unsere Demosite, eine mögliche Website des „Marzipanvereins Lübeck" zeigt in zwei Versionen die bisherige Tabellenbasierte Gestaltung einer Internetseite und die behindertengerechte Gestaltung via XHTML 1.1 und CSS.

Ganz wichtig für Designer ist, dass dessen Website auch nach der Anpassung genauso aussehen sollte, wie mit tabellenbasiertem Design. Nichts einfacher als das:

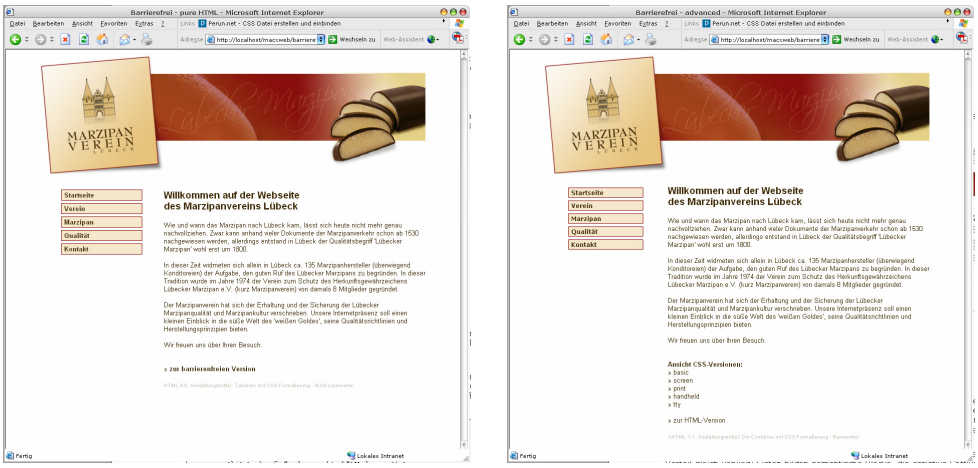

Abb. 26 HTML-Basierte Site (links) und barrierefrei (rechts)

Der Quelltext (in Kurzform) zeigt die technischen Unterschiede.

```
<table cellspacing="0" cellpadding="0" border="0">
<tr>
    <td colspan="2" height="270">[Grafischer Kopf]</td>
</tr>
<tr>
    <td width="248" align="center">[Menü]</td>
    <td width="523">[Inhalt]y/td>
</tr>
</table>
```

Abb. 27 HTML-Version mit Tabellen zur visuellen Gestaltung

```
<div id="head">[Grafischer Kopf]</div>
<div id="content">
    <ul id="menu">[Menü]</ul>
    <div id="incontent">[Inhalt]</div>
</div>
```

Abb. 28 Barrierefreie Gestaltung mit weniger Quelltext

Wie man sieht, können wir den Quelltext bei der Entrümpelung sogar noch verkleinern. Die visuelle Gestaltung übernimmt hier direkt CSS. So kann man unterschiedliche Websites mit einem und dem gleichen Inhalt online stellen, in dem man ganz einfach die Stylesheets austauscht.

Unsere Beispielsite würde in der Druckversion, bei der wir via media-Argument unterschiedliche Stylesheets einbinden, so aussehen:

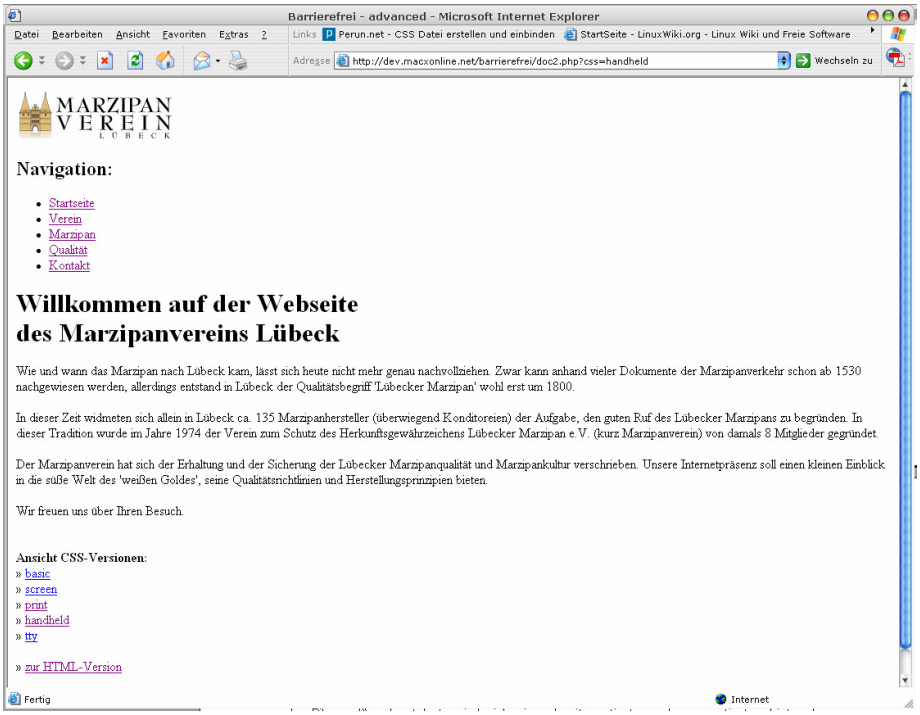

Abb. 29 Druckversion der Beispielsite

4.2 Fallbeispiel 1: Listen

Listen werden oft als die hässlichen Entlein in HTML dargestellt. Jeder, der HTML „von der Pike auf" gelernt hat, wird sich einmal mit sortierten oder unsortierten Listen beschäftigt haben. Freunde grafischer Editoren werden Listen als „Aufzählungselemente" kennen. Beide Arten von Webdesignern werden Listen in der Praxis eher selten verwenden.

Unser Fallbeispiel zeigt die Anwendung unsortierter Listen als Navigationselement. Der Vorteil gleich vorweg: Listen bilden semantische Blöcke, die assistive Software als solche erkennen kann. Zudem bilden sie auch im Quellcode einen leicht zu identifizieren Block.

Die einfachste Liste sieht so aus:

```
<h2>Navigation:</h2>
<ul id="menu">
    <li><a href="start.html">Startseite</a></li>
    <li><a href="verein.html">Verein</a></li>
    <li><a href="marzipan.html">Marzipan</a></li>
    <li><a href="qualitaet.html">Qualität</a></li>
    <li><a href="kontakt.html">Kontakt</a></li>
</ul>
```

Abb. 30 Eine ganz einfache Liste

Besonders schön sieht das nicht aus, aber die Struktur ist klar erkennbar. Mit Hilfe von CSS verschönern wir unsere Liste und verabschieden uns als erstes von den Aufzählungszeichen, indem unsere Liste mit eigenem ID-Attribut ausstatten und im Stylesheet das neue Element mit dem Attribut `list-style: none` versehen.

```
#menu { list-style: none; margin: 0px; padding: 0px; }
#menu a {
 width: 150px; color: #3F2B07; background: #F6E8CC;
 text-decoration: none; font: bold 12px verdana, sans-serif;
 display: block; border: 1px solid #800000; padding: 2px 5px 2px 5px;
margin: 5px 0 0 0;
}
<ul id="menu">..</ul>
```

Abb. 31 Liste ohne Aufzählungszeichen

Nun erstellen wir über die Pseudoklasse hover auch gleich einen MouseOver-Effekt: sobald der Mauszeiger über ein Listenelement (list-item) fährt, ändert sich dessen Hintergrundfarbe:

```
  #menu a:hover { color: #800000; background-color: #E7C37B; }
```

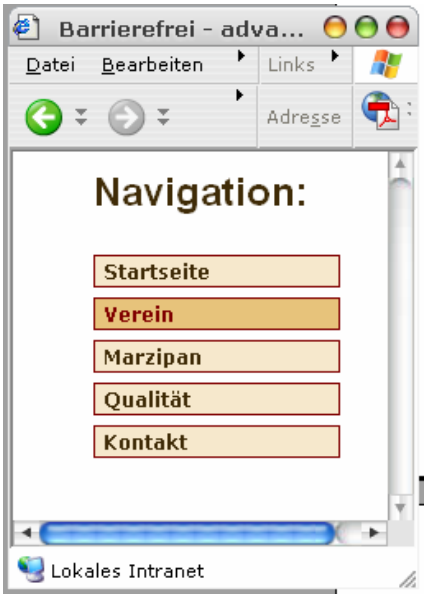

Abb. 32 Liste mit MouseOver-Effekten

Schon haben wir eine ansprechend aussehende Navigation mit MouseOver-Effekten! Und das alle ohne Javascript und Grafik. Selbstverständlich sind wir mit der Navigation nicht auf die vertikale Darstellung beschränkt – ein `display: inline` im -Tag bewirkt eine horizontale Navigation.

Der Phantasie sind keine Grenzen gesetzt. Man beachte, dass der zugrunde liegende Quellcode der Liste immer gleich bleibt – geändert wird lediglich das eingesetzte Stylesheet; gewissermaßen haben wir hier ein Paradebeispiel der Trennung zwischen Layout und Inhalt.

Selbst Grafiken lassen sich mit dieser Methode dynamisch austauschen. Hier muss man jedoch darauf achten, dass sich (nur) mit CSS keine neuen Elemente in den DOM-Baum eines Dokumentes einfügen lassen, wie es mit Javascript möglich wäre. Umgehen lässt sich dies, indem man in den „Normalzustand" des zu verändernden Objektes eine „unsichtbare Grafik" – z.B. ein transparentes Bild – als Background einsetzt.

4.3 Fallbeispiel 2: Formulare

Auf fast jeder Website treffen wir auf Formularelemente – sie sorgen für Interaktivität zwischen Betreiber und Benutzer. Ohne Formulare wäre die gesamte Welt des eCommerce vollkommen undenkbar. Wie sind Formulare aufgebaut? Sie bestehen aus einer mehr oder weniger großen Zahl von unterschiedlichen Eingabeelementen (einzeilige bzw. mehrzellige Textfelder, Radioboxen, Checkboxen, Listen und Dateidialogen), sowie dem öffnenden und schließenden <form>-Tag. Wie die erfassten Formulardaten im Einzelnen verarbeitet werden, soll uns an dieser Stelle nicht weiter interessieren.

Meistens sind die einzelnen Eingabefelder auch in irgendeiner Form beschriftet (meistens per HTML-Tabelle), damit der Besucher weiß, was er wo eintragen soll. Einen logischen Zusammenhang zwischen Beschriftung („Label") und Eingabefeld gibt es meistens aber nicht – Anwender assistiver Software haben dann einen schweren Stand. Genauso wenig sind zusammengehörige Felder selten zu einer Gruppe zusammengefasst.

Dabei lässt sich mit HTML 4.0 genau das recht einfach bewerkstelligen. Betrachten wir zunächst ein „herkömmliches" Formular (zu vergleichen mit dem berühmt-berüchtigten „herkömmlichen Waschmittel"):

```
<form action="script.php" method=“post“>
<strong>Angaben zur Person</strong><br>
Name<br><input type="text" name="name"><br>
Vorname<br><input type="text" name="vorname"><br>
Alter<br><input type="text" name="alter" size="2">
<br><br>
<strong>Lieblingsobst</strong><br>
<input type="checkbox" name="obst1" value="1"> Zitronen<br>
<input type="checkbox" name="obst2" value="2"> Bananen<br>
<input type="checkbox" name="obst3" value="3"> Melonen<br>
<input type="checkbox" name="obst4" value="4"> Erdbeeren
<br><br>
<input type="submit" value="Absenden">
</form>
```

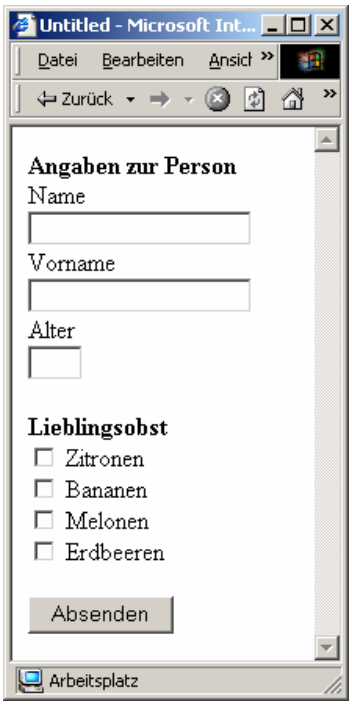

Abb. 33 Herkömmliches Formular

Mit diesem Formular werden die meisten Besucher umgehen können; trotzdem wollen wir es natürlich verbessern… Setzen wir die in HTML 4.0 bereitgestellten Formular-Tags konsequent ein, erhalten wir ein wesentlich aussagekräftigeres Formular:

```
<form action="script.php">
<fieldset style="padding:10px">
  <legend style="color:#0000ff">Angaben zur Person</legend>
    <label for="name">Name</label><br>
    <input type="text" name="name" id="name"><br>
    <label for="vorname">Vorname</label><br>
    <input type="text" name="vorname" id="vorname"><br>
    <label for="alter">Alter</label><br>
    <input type="text" name="alter" id="alter" size="2">
  </fieldset>
<fieldset style="padding:10px">
  <legend style="color:#0000ff">Lieblingsobst</legend>
    <input type="checkbox" name="obst1" id="obst1" value="1">
    <label for="obst1">Zitronen</label><br>
    <input type="checkbox" name="obst2" id="obst2" value="2">
    <label for="obst2">Bananen</label><br>
    <input type="checkbox" name="obst3" id="obst3" value="3">
    <label for="obst3">Melonen</label><br>
    <input type="checkbox" name="obst4" id="obst4" value="4">
    <label for="obst4">Erdbeeren</label>
</fieldset>
<input type="submit" value="Absenden">
</form>
```

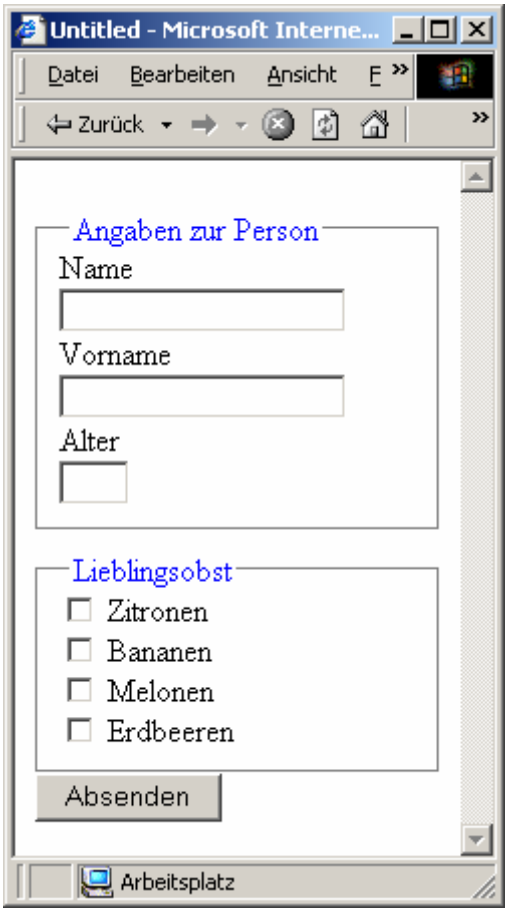

Abb. 34 Verbessertes Formular

Dieses Formular fasst mehrere Eingabefelder via <fieldset> zu logischen Gruppen zusammen. <legend> sorgt für eine Beschriftung und zeichnet auch gleich einen Rahmen um die Gruppe (wer das nicht möchte, stellt den Rahmen per CSS ab). Jedes Eingabefeld ist nun neben dem Attribut „name" auch über „id" identifizierbar. Das ist für den Einsatz von <label> wichtig: mit diesem Tag wird nun endlich eine logische Verknüpfung zwischen Beschriftung und Eingabefeld ermöglicht, ganz gleich, wo die Beschriftung sich tatsächlich befindet. Über die „id" kann der Browser den Zusammengang feststellen. Im Klartext heißt das, wenn der Benutzer auf das Label klickt, erhält das zugeordnete Eingabefeld den Fokus. Das ist vor allem bei „kleinen" Eingabeelementen wie Radio- oder Checkboxen interessant, da sie nun mit dem Label einen wesentlich größeren Hotspot erhalten.

Natürlich kann man dieses Verhalten auch mit Javascript erreichen (und wir haben das auch schon gesehen) – aber wozu der Umstand, wenn man dies alles auch mit Bordmitteln erreichen kann?

Formulare lassen sich noch benutzerfreundlicher gestalten; so wäre es für den Anwender schön, wenn das jeweils ausgewählte Eingabefeld optisch hervorgehoben wird. An dieser Stelle muss der Marktführer der Browser, der Microsoft Internet Explorer, leider passen – offenbar hält man sich in Redmond lieber an eigene Standards. Andere Browser wie Netscape, Mozilla oder die verschiedenen Mac- und Unix-Browser bieten hier die Unterstützung von Pseudo-Klassen:

```
input: focus { background:#e0e0e0; }
input: hover { background:#ff0; }
```

Abb. 35 Eingabelemente mit Pseudoklassen besser hervorheben

Zur „Ehrenrettung" des Internet Explorers sei jedoch vermerkt, dass sich dieses Verhalten immerhin nachbauen lässt. Dann ist der Einsatz von Javascript allerdings unver-

meidbar. Im Microsoft-Browser ermöglicht Javascript, auf bestehende Styles zuzugreifen und Designer können Regeln (so genannte „behaviors") definieren, die dafür sorgen, dass sich beliebige Elemente beim Auftreten eines Events (z.B. MouseOver) auf eine benutzerdefinierte Art und Weise verhalten. Browser wie Netscape oder Opera, die keine Behaviors kennen, ignorieren diese Angaben ganz einfach.

Weitere Informationen über Behaviors und ihren Einsatz gibt es unter http://www.xs4all.nl/~peterned/csshover.html

4.4 Fallbeispiel 3: Tooltipps mit CSS

Tooltipps, die beim Darüberfahren mit der Maus erscheinen, sind eine schöne und vor allem anwenderfreundliche Sache. Meistens werden sie durch den Einsatz von Javascript umgesetzt – sie lassen sich aber auch mit reinem CSS erzeugen.

```
a.tt { position: relative; z-index : 24; color : #000; text-
decoration: none; cursor: help; }

a.tt: hover { z-index: 25; background-color: #EBEFFC; text-decoration
: none; }

a.tt span { display: none }

a.tt: hover span { display: block; position: absolute; top: 2em; left:
2em; width: 17em; border: 1px solid #707070; background-color:
#fffacd; color: #000000; text-align: left; padding: 2px; font-size:
11px; text-decoration: none; }
```

Abb. 36 Tooltipps mit CSS - das Stylesheet

Die Idee bei diesem Ansatz besteht darin, für Tooltipp-Links das Attribut „relative" zu verwenden, damit ein zwischen den <a>-Tags platziertes -Element korrekt zum Tooltipp dargestellt werden kann. Diese Methode funktioniert im Internet Explorer ab Version 5.5, Mozilla und Netscape.

```
<p>PHP ist eine tolle < a href="#" class="tt">Programmiersprache
<span> ...mit der sich leistungsstarke Webapplikationen schnell und
einfach entwickeln lassen</span></a>.</p>
```

Abb. 37 Tooltipps mit CSS - das Ergebnis

Definiert man nun zwei Style-Sets (einen für die Bildschirm- und einen für die Druckerausgabe), dann lassen sich die Tooltipps auch in ansprechender Forma zu Papier bringen. Im IE funktionieren die Attribute :before und :after leider nicht (sie werden ignoriert). Andere Browser dagegen können davon Gebrauch machen und die Tooltipps beispielsweise geklammert ausgeben.

4.5 Spezialnavigation einsetzen

Hierbei handelt es sich um zusätzliche Navigationselemente über ein Dokument, die in herkömmlichen Browsern unsichtbar bleiben, für Anwender von Screen-Readern aber eine ungeheure Erleichterung bieten. Stellen wir uns eine ganz normale Webseite mit einem layouteten Kopfbereich, einer Navigation und vielen weiteren Elementen vor. Ein Screen-Reader würde dies alles von oben nach unten vorlesen. Viel praktischer wäre es doch, wenn man sich all das schenken könnte und gleich zu Beginn der Seite einen Link fände, der einen direkt zum eigentlichen Inhalt führt. Das lässt sich folgendermaßen erreichen:

```
<body>
<a href="#inhalt" title="Direkt zum Inhalt"><img src="blank.gif"
alt=""></a>

…[beliebiger Code]

<a name="inhalt"></a>
..[Inhalt]
</body>
```

Abb. 38 Interne Anker erleichtern die Navigation

So genannte „interne Anker" machen es möglich. In der Anfangszeit des Webs wurden sie übrigens recht häufig verwendet, heute wird meist eine neue Seiten geladen (oder unschön ein Popup geöffnet).

4.6 Style Switcher

Kompliziert wird es, ein Webdesign zu entwickeln, welches Menschen mit Sehschwächen gerecht wird. Während man sich bei Blinden darüber wenig Gedanken machen muss (sie sehen ja nun gar nichts), wollen Menschen mit Sehschwächen auf jeden Fall etwas sehen. Und damit muss das Webdesign klarkommen – beinahe ein Alptraum für Webdesigner. Generell gilt, das sehbehinderte Menschen die folgenden vier Punkte selbst einstellen können müssen:

1. Schrift in Größe und Laufweite
2. Vorder- und Hintergrundfarbe
3. Reihenfolge von Elementen (mehrspaltiges Layout ist hier unpraktisch)
4. Navigation

Bei der Entwicklung einer Webseite kann man grundsätzlich so viele Styles einsetzen, wie man möchte, um beliebig viele alternative Anzeigemöglichkeiten anzubieten. Beispielsweise könnte die Hauptnavigation ganz am Anfang stehen und eine Art digitalen Ariadne-Faden bereitstellen („You are here"). Letztendlich kann sich dann jeder Besucher seinen bevorzugten Stil einstellen.

Nur wenige Browser bieten derzeit die Möglichkeit, den Style zu wechseln und noch weniger Menschen machen Gebrauch davon bzw. besitzen überhaupt die Kenntnis von dieser Möglichkeit. Die logische Konsequenz: der Designer muss dafür sorgen, dass de Besucher irgendetwas in die Hand gedrückt bekommen, womit sie den Style wechseln können (z.B. Buttons oder Links). Das ist aber viel leichter gesagt als getan und ganz andere Probleme tauchen auf:

- Sehr oft sind die Elemente zum Style-Wechsel sehr klein und effektiv versteckt, weil die Designer sie nicht allzu sehr ins Auge springen lassen wollen
- Oftmals kann nur die Schriftgröße in ein paar bescheidenen Schritten verändert werden – die komplette Seite wird selten in den Wechsel mit einbezogen

- Es existiert keine einheitliche Terminologie geschweige den einen Standard für den Ort, an dem die Wechselelemente zu finden sind. Das ist umso bizarrer, als dass solche Elemente ja für Menschen gedacht sind, die ohnehin schon schlecht sehen – und diese Besucher müssen erst intensiv suchen (obwohl sie ja gar nicht wissen, ob derartige Elemente auf der besuchten Seite überhaupt existieren).

Mittlerweile gibt es den Vorschlag, Webseiten mit einer Art Präferenzseite auszustatten („Optionen zur Anzeige"). Die gewählten Einstellungen könnten sogar für spätere Besuche gespeichert werden – hier wären Cookies eine durchaus angemessene Verfahrensweise. Das Problem besteht jedoch darin, dass eine solche Präferenzseite per definitionem barriefrei sein müsste. Ein Patentrezept dafür existiert (noch) nicht, gerüchteweise könnten entsprechende Guidelines in ferner Zukunft jedoch vom W3C aufgestellt werden.

4.7 Sprachliche Auszeichnungen

Ein Screen-Reader, der eine Webseite vorliest, „versteht" natürlich nicht ihren tatsächlichen Inhalt. Noch weniger kann er die verwendet Sprache identifizieren. In der Regel wird ein Screen-Reader auf eine Sprache hin konfiguriert (in unserem Fall sicherlich Deutsch) und „spricht" dann alle fremdsprachlichen Wörter in „seiner" Sprache aus. Das mag u.U. witzig klingen, bringt dem auf den Screen-Reader Angewiesenen aber kaum Erheiterung. Abhilfe schafft die korrekte sprachliche Auszeichnung von Fremdwörtern:

```
<p>Indem er seiner Dame nicht die Hand küsste, begang er einen
folgenschweren <span lang="fr">fauxpas</span>… Die <span
class="en">Twentieth Century Fox</a> ist ein großes Unternehmen…</p>
```

Abb. 39 Auszeichnung der verwendeten Sprache

5 Prüfungen – so ist's richtig

5.1 HTML-Validator

(X)HTML-Dokumente zu validieren ist im Grunde genommen ganz einfach. Man ruft einen Validator im Browser auf (zum Beispiel den W3C-Validator `http://validator.w3.org/`), gibt die URL seiner Website ein und erhält das entsprechende Ergebnis. Eine solche schnelle Validierung setzt natürlich voraus, dass der richtige Doctype (vergl. 3.1.1) im Dokument enthalten ist. Zwar ist der Validator in der Lage, andere Doctypes bei der Prüfung zu nutzen; allerdings ist das dann keine wirklich korrekte Validierung mehr. Gewaltige Unterschiede gibt es auch, wenn man spaßhalber XHTML 1.1-Dokumente zu HTML 4.0 validiert und anders herum.

Wie sehen nun die Meldungen aus?

This Page is Valid XHTML 1.1

Abb. 40 Erfolgreiche Prüfung eines XHTML 1.1-Dokumentes

This Page is **not** Valid HTML 4.01 Transitional

Abb. 41 Prüfung der als barrierefrei titulierten Website stern.de fehlgeschlagen

```
I was not able to extract a character encoding labeling from any of
the valid sources for such information. Without encoding information
it is impossible to validate the document.
```

Abb. 42 Hier ist viel zu tun: Der Validator konnte noch nicht einmal den Doctype erkennen. Ein ganz schlechtes Resutat.

Im Falle eines Fehlers wirft der Validator jeden einzelnen Fehler nach einander aus und beschreibt diesen Kurz. Diese sollte man nacheinander abstellen. Doch Vorsicht: Fehler, die der Validator als erstes findet, könnten schon Ursache für alle weiteren Fehler sein. Die Revalidate-Funktion erlaubt es dem User, die Seite erneut prüfen zu lassen, was die Sache vereinfacht. Im Laufe der Zeit sollte man sich an die möglichen Fehler gewöhnt haben und kann diese dann ohne den ständigen Wechsel zwischen Validator und Editor beheben.

So könnten Fehler aussehen:

```
Below are the results of attempting to parse this document with an
SGML parser.

Line 6, column 41: document type does not allow element "META" here (explain...).
  <meta http-equiv="EXPIRES" content="120" />
                                           ^

Line 16, column 65: document type does not allow element "LINK" here (explain...).
  ...ortcut icon" href="http://www.stern.de/favicon.ico" />
                                                         ^

Line 18, column 22: document type does not allow element "STYLE" here (explain...).
  <style type="text/css">
                        ^

Line 51, column 6: end tag for element "HEAD" which is not open (explain...).
  </head>
       ^

Line 52, column 5: document type does not allow element "BODY" here (explain...).
  <body><div class="invisible"><img src="http://stern.ivwbox.de/cgi-
bin/ivw/CP/101
     ^

Line 637, column 123: end tag for element "DIV" which is not open (explain...).
  ...px 12px;">' + banner_content + '</div>';
                                              ^
```

Abb. 43 Fehlerliste des W3C-Validators

Da die Zeilennummerierung bei dem Quelltext der PHP-Datei anders aussieht als die des resultierenden HTML-Dokumentes, erlaubt es der Validator, den Quelltext anzuzeigen, sodass man sehr schnell Fehler lokalisieren kann.

Auch manche Editoren (wie z.B. Macromedia HomeSite) bieten interne Validatoren. Das ermöglicht die schnelle lokale Validierung der Dokumente ohne Onlineverbindung.

5.2 CSS-Validator

Die Validierung von CSS-Dokumenten (zum Beispiel unter `http://jigsaw.w3.org/css-validator/`) läuft ähnlich ab wie die Validierung von (X)HTML-Dokumenten. Eine erfolgreiche Validierung sollte dann so aussehen:

```
Keine Fehler oder Warnungen gefunden
Gratulation
```


```
Dieses Dokument wurde als CSS validiert!
```

Abb. 44 Erfolgreiche Validierung eines CSS Stylesheets

5.3 Sites auf Barrierefreiheit testen

Während man (X)HTML-Dokumente und CSS-Files in wenigen Sekunden validieren lassen kann, ist dies bei der Prüfung auf Barrierefreiheit schon sehr viel schwieriger. Neben nicht zuverlässigen Onlinevalidatoren gibt es auch Software zum Downloaden, die statische Dokumente lokal prüft – für uns PHPler nicht gerade geeignet.

Das größte Problem bei diesen Validatoren ist jedoch, dass manche Vorgaben einfach nicht überprüft werden können. So muss zum Beispiel immer noch der Programmierer entscheiden, ob er ein Bild genauer beschreiben kann, oder dies nur für die grafische Darstellung gedacht ist keine genaue Beschreibung nötig hat.

Die im Web verfügbaren Validatoren können allenfalls als Kurztest der eigenen Webseite verwendet werden. Viel effektiver ist es, wenn man sich eine Checkliste erstellt und diese jedes Mal durchgeht. Das alleine reicht meistens schon aus.

Ideale Tests müssten eigentlich in Zusammenarbeit mit den Betroffenen geschehen, was sich in der Praxis jedoch fast immer als unmöglich erweist. Immerhin kann man sicher sein, dass die Einhaltung der Standards und die Verwendung valider Code weitaus mehr als nur die bekannte „halbe Miete" darstellt. Sind diese beiden Punkte gegeben, ist die tatsächliche Barrierefreiheit einer Site mit hoher Wahrscheinlichkeit gesichert.

6 Abschluss

6.1 Ausblick

Wie sich die Zukunft möglicherweise entwickeln wird, steht wie immer in den sprichwörtlichen Sternen. Aus diesem Grunde können wir auch wieder absolut keine Prognose abgeben, weil sich eh alles ganz anders (und erfahrungsgemäß viel wilder, als wir träumen) entwickeln wird. Es besteht eine realistische Chance, dass zukünftige Browser die verabschiedeten Standards einhalten werden und die Zahl der Problemfälle im Bereich Darstellung stark abnehmen wird. Noch schöner wäre eine Welt, in der es keine Rolle spielt, welchen Browser man zum Surfen verwende – eine Welt ohne „Optimiert für…"-Buttons.

6.2 Fazit

Dieser Vortrag ist einer der umfangreichsten in der nun schon dreijährigen Geschichte der PHP usergroup Hannover. Aber was will man auch bei einem derart komplexen Thema erwarten – zumal sich gleich zwei Autoren damit auseinander gesetzt haben?

Immerhin: der geneigte Webdesigner sollte nach dem (hoffentlichen) Genuss des vorliegenden Dokumentes die nötigen Impulse und die grundsätzlichen Werkzeuge zur Hand haben, um zukünftig barrierefreie Webseiten realisieren zu können. Mit an Sicherheit grenzender Wahrscheinlichkeit werden Probleme auftreten, die mit „altem" HTML-Code („Gestrüpp") so nicht aufgetaucht wären, aber wir denken, dass es eine lohnende Herausforderung darstellt, sich diesen Problemen zu stellen.

Das Internet hat den Alltag der Menschen längst erobert. In nur wenigen Jahren hat es sich vom elitären Zirkel der technikverliebten Ingenieurstypen über Cyberspace-Phantasien zum gesellschaftlich anerkannten Kommunikationsmedium gewandelt. Seine Extravaganz und teilweise auch seine Faszination haben vielleicht abgenommen – nicht aber seine Bedeutung.

Eine Gesellschaft, die mit digitalen Informationsnetzen lebt und zum Fortbestehen auf sie angewiesen ist, darf es sich nicht erlauben, ganze Gruppen von ihnen auszuschließen. In unserem Text ist in der Argumentation für barrierefreie Seiten pauschal von Behinderten die Rede gewesen – keineswegs vergessen werden dürfen hier ab ältere Menschen. Wenn in naher Zukunft in Deutschland mehr als die Hälfte der Bevölkerung über 60 Jahre alt ist, wird das Argument von betroffenen Minderheiten ad absurdum geführt. Und noch weiter gedacht: elektronische Kommunikationsmedien und digitale Netze werden den letzten Winkel unseres Planten erobern und das Leben der Menschen beeinflussen. Viele davon haben nie eine Schulbildung genossen, bringen kaum technisches Wissen mit – sind aber gleichzeitig darauf angewiesen, mit den Anfordernissen unserer modernen (westlichen) Welt Schritt zu halten.

Barrierefreiheit ist keine modische Erscheinung oder ein Trend. Über kurz oder lang werden wir alle uns damit in der unterschiedlichsten Weise auseinandersetzen, Standards entwickeln, Barrieren niederreißen und zukünftige Entwicklungen verantwortungsbewusst planen.

Letztendlich handelt es sich um eine gesellschaftliche Überlebensfrage.

7 Anhang

7.1 Rückblick: die Entwicklung des Internets

Kein Vortrag ohne Vermittlung der nötigen Grundlagen… welche mit einem kurzen historischen Rückblick das Verständnis der heutigen Situation verbessern soll.

Wir blicken zurück ins Jahr 1982: erstmalig fällt der Begriff „Internet" – die bereits zehn Jahre zuvor gegründete Internet working group (INWG), die sich schon damals mit den Grundlagen von technischen Übertragungsprotokollen auseinandersetzte, stellt das standardisierte TCP (transmission control protocol) sowie IP (internet protocol) vor. Dies war ein bedeutender Schritt nach vorne, denn nun konnten die vielen einzelnen voneinander isolierten Netzwerke mit ihren jeweils eigenen Übertragungsverfahren sich zusammenschließen. TCP/IP wurde zum Standard – und das ist es heute noch.

1989 schließen sich die europäischen service provider zum RIPE (Reseaux IP Europeans) zusammen und schaffen gemeinschaftlich die technische und administrative Basis für ein europäisches Netzwerk. Ein Jahr später dringt das Internet in kommerzielle Sphären ein; die US-amerikanische National Research Initiative (CNRI) biete das neue MCI Mail an. Gleichzeitig hört der Urgroßvater des Netzes, das ARPANET, auf zu existieren.

1992 kommt ein neuer Servicedienste zu den bereits bestehenden Internetkomponenten eMail, ftp, telnet, WAIS und gopher hinzu: das world wide web. Mit ihm einher geht die neue Seitenbeschreibungssprache HTML – vorgestellt durch Tim Bernhard-Lee am Schweizer Kernforschungszentrum CERN.

HTML ist heute noch das tägliche Brot aller Webprogrammierer (allen Unkenrufen von Flash, XML usw. zum Trotz). Was inzwischen alles mit HTML gemacht wird, ist wesentlich umfangreicher als alles, was man sich für diese Sprache einmal vorgestellt hatte. Umso wichtiger sind Standards, die sicherstellen sollen, dass HTML-Dokumente validiert werden können, um eine einigermaßen korrekte Darstellung in beliebiger Software zu gewährleisten. Dass sich hier den letzten Jahren eine ganze Menge getan hat, soll exemplarisch der Rückblick auf den „Browserkrieg" im folgenden Kapitel belegen.

7.2 Rückblick: Der Browserkrieg (1995-2001)

Begonnen hat die Geschichte mit der Software MOSAIC, die der Student Marc Andreessen 1992 am National Center for Supercomputing Applications (NCSA) an der Universität Illinois entwickelt. 1993 war sein Browser extrem erfolgreich und mutierte zum De-facto-Standard. „...an application program so different and so obviously useful that it can create a new industry from scratch..." schrieb die New York Times 1993.

Besonders innovativ an MOSAIC war die gleichzeitige Darstellung von Text und Grafik innerhalb einer Seite sowie die visuelle Unterstützung von Hyperlinks. Ebenfalls integriert war eine intuitiv zu bedienende Oberfläche (GUI). MOSAIC wurde für nahezu jede Hardwareplattform angeboten.

Mitte 1994 gründeten Andreessen und Jim Clark die Mosaic Communications Corp. Im Silicon Valley, die Ende 1994 in Netscape Communications Corp. umbenannt wurde. Das Geschäftsmodell: für Studenten und Bildungseinrichtungen war der Browser frei zugänglich, für alle anderen kostete die Software 39,00 $. Parallel dazu entwickelte Netscape Server-Software (inkl. SSL-Technologie), die im preislichen Segment zwischen 1.500,00 – 5.000,00 $ vor allem an Versandfirmen und Banken verkauft wurde. Die Strategie hieß „Marktanteile vor Gewinn". 1995 betrug der Marktanteil von Netscape 85%.

Anfang 1996 erkennt Microsoft auf einmal das ungeheure Potenzial des Internets, welches man in Redmond jahrelang nicht beachtet bzw. kräftig unterschätzt hatte. Kurzerhand lässt Bill Gates eine Milliarde Dollar in die Entwicklung des Microsoft Internet Explorers (IE) investieren. Seine Strategie sieht anders aus als die von Netscape: den IE kann jedermann kostenlos im Netz herunterladen.

Die ersten Versionen des IE sind kaum erwähnenswert. Selbst die Version 3.0 hinkt dem unter ebenfalls unter dieser Releasenummer erscheinenden Netscape-Browser technisch hinterher. Doch bereits die folgende Version übertrifft den (ebenfalls neuen) Netscape Communicator.

Microsoft geht einen Schritt weiter und „überredet" die PC-Hersteller, Windows95 gemeinsam mit dem IE als Standardausrüstung auf jedem PC zu installieren. In den folgenden Jahren nutzt der Redmonder Riese seine führende Marktstellung weiter aus und koppelt den IE mit Windows98. Und das neue Windows NT sticht das teurere Netscape-Paket im Bereich Serversoftware aus. Gleichzeitig entwickelt Microsoft eigene Standards und Formate (z.B. ActiveX).

Im Dezember 1997 ergeht ein einstweiliges Urteil gegen Microsoft: die Windows-Lizenzvergabe ist nicht mehr an die Installation des IE gebunden. Ein Jahr später wacht Netscape endlich auf und ändert seine Strategie: auch der Communicator kann nun kostenlos aus dem Netz heruntergeladen werden.

Inzwischen ist Microsoft eine Allianz mit AOL eingegangen und erhält dadurch rund 13 Millionen neuer IE-Benutzer auf einen Schlag. Mit diesem Coup zieht die Gates Company mit Netscape gleich. Es kommt zwischen den beiden Unternehmen zu einem Wettbewerb zwischen Standards, der auf den Schultern der Benutzer ausgetragen wird.

Im April 1998 legt Netscape den Quellcode seines Browser offen: das Mozilla-Projekt ist geboren. Ein von Netscape weitestgehend unabhängiges Entwicklerteam überwacht als Koordinator die selbständige Weiterentwicklung des freien Browsers. Das Release von Netscape 6 (mit Mozilla Know-how) floppt gewaltig. Gleichzeitig bringt Mozilla das erste eigene Release heraus.

Mit Beginn des neuen Jahrtausends ist der Marktanteil von Netscape faktisch bedeutungslos geworden. Ob es nun an der schieren Größe des „Gegners" gelegen hat, an der viel zu späten Reaktion auf die Konkurrenz oder die möglicherweise chaotischen Strukturen der open source Szene, sei an dieser Stelle dahingestellt.

7.3 Zusätzliche Informationen zu Behinderungen

Grundlegend kann man Behinderungen in mehrere Klassen unterteilen, wobei sich die folgende Zusammenfassung ergibt (welche keinen Anspruch auf detaillierte Vollständigkeit legt, um den Rahmen des Vortrages nicht gänzlich zu sprengen).

- Beim *Grauen Star* kommt es zu einer Trübung der Linse, wodurch der Eindruck eines verschwommenen Bildes entsteht.
- Eine *Makula-Degeneration* bezeichnet die Zerstörung der Nervenzellen, die für das Scharf-Sehen zuständig sind.
- Ist der Augeninnendruck erhöht, spricht man vom *Grünen Star*. Hierbei wird die Sehrnerv so stark geschädigt, dass es zu Ausfällen im Sichtfeld kommen kann.
- Ein dauerhaft hoher Blutzuckerwert kann *Diabetische Retinopathie* verursachen, welche zu dauerhaften Ausfällen im Sichtfeld kommt.
- *Retinitis Pigmentosa* bezeichnet eine Gruppe von Netzhauterkrankungen, deren Anzeichen von Blendempfindlichkeit, Nachtblindheit bis hin zum so genannten „Tunnelblick" reichen.
- Wenn sich Teile der Netzhaut ablösen, führt dies zu Totalausfällen im Sichtbereich.
- Menschen mit *Albinismus* leiden unter stark herabgesetzten Sehvermögen und hoher Blendempfindlichkeit.

7.3.1 Körperbehinderungen

Unter diese Gruppen fallen unter anderem

- Bewegungsstörungen mit spastischen Lähmungen, verursacht durch eine Schädigung des zentralen Nervensystems
- Einschränkung der Motorik und der Sensibilität aufgrund von Fehlbildungen der Wirbelsäule und des Rückenmarks
- Schädigung des Skelettes und der Muskulatur, Verlust von Gliedmaßen
- Chronische Krankheiten und Fehlfunktionen von Organen (z.B. Asthma, Hautkrankheiten oder innere Krankheiten)

7.3.2 Geistige Behinderungen

In Bezug auf Barrieren bei der Internetnutzung sind in der Gruppe der geistigen Behinderungen vor allem Epilepsie und Dyslexie zu nennen. Erstere ist eine Krankheit des zentralen Nervensystem und äußert sich bei den Betroffenen durch Anfälle mit vorübergehender und Abnormer Aktivierung der Gehirnzellen – letztere bezeichnet eine Leseschwäche.

7.3.3 Sprachbehinderungen

Die bekannteste Sprachbehinderung ist sicherlich der Autismus. Diese spezielle Form der Selbstbezogenheit wird durch eine Beziehungs- und Kommunikationsstörung bedingt. Sie beeinträchtigt erheblich die persönliche Entwicklung und die kognitiven Prozesse.

Eine weitere Ursache für Sprachbehinderungen kann auch Multiple Sklerose sein, wobei diese Krankheit eine Vielzahl weiterer Symptome mit sich bringt.

7.3.4 Hörschäden

Am weitesten verbreitet sind über längere Zeiträume entstehende Schwerhörigkeit, plötzlich verursachte Ertaubung (z.B. durch einen Hörsturz, Unfall etc.) sowie die angeborene Gehörlosigkeit.

7.4 Links

Die hier aufgeführten Links halten wir für empfehlenswerte Ausgangspunkte für weitere Informationen und Recherchen. Sortiert sind sie in alphabetischer Reihenfolge.

- „Einfach für alle" – Aktion Mensch e.V.

  http://www.einfach-fuer-alle.de

  Besonders empfehlenswert: das Access[B]log:

  http://www.einfach-fuer-alle.de/blog/index.php
- A List Apart Webmagazine
  http://www.alistapart.com
- Aktionsbündnis für barrierefreie Informationstechnik
  http://www.abi-projekt.de
- Angebotsscreening zum Thema "Behinderung" - Webrecherche von 50 Internetauftritten
  http://www.fdst.de/pdf_zip/fdst_de_webrecherche.pdf
- A-Prompt, Software zum Testen von barrierefreiem Webdesign
  http://www.wob11.de/publikationen/aprompt/programm.html
- Barrierekompass
  http://www.barrierekompass.de
- BIENE: "Barrierefreies Internet eröffnet neue Einsichten" (Award)
  http://www.biene-award.de/award/index.html

- Cascading Stylesheets Level 1 Specification
  http://www.w3.org/TR/REC-CSS1
- Cascading Stylesheets Level 2 Specification
  http://www.w3.org/TR/REC-CSS2
- Color Vision, Color Deficiency
  http://www.firelily.com/opinions/color.html
- CSS user interfaces
  http://userwww.sfsu.edu/~ibueno/wireframe.html
- CSS-Layouts zum Mitnehmen
  http://www.thenoodleincident.com/tutorials/box_lesson/boxes.html
- devmag.net
  http://www.devmag.net/html/
- Diverse Tipp's & Tricks rund um CSS
  http://www.css-technik.de/
- Doctype-Switching: Wie Browser über die Darstellung von (X)HTML entscheiden
  http://www.heise.de/ix/artikel/2004/03/136/
- Einführung in Aural Stylesheets
  http://www.gpad.ac.ru/%7Edemon/css2/aural.html
- Forschungsinstitut Technologie Behindertenhilfe der Fernuni Hagen
  http://www.fernuni-hagen.de/FTB/info/betrifft.htm
- HTML 4.01 Specification
  http://www.w3.org/TR/html401
- iX CSS Tutorial
  http://www.heise.de/ix/artikel/2003/03/050/
- Stiftung digitale Chancen
  http://www.digitale-chancen.de
- Teilhabe behinderter Menschen – Informationen aus dem Bundesministerium für Gesundheit und Soziale Sicherung
  http://www.bmgs.bund.de/deu/gra/themen/sicherheit/teilhabe/index.cfm
- User Agent Accessibility Guidelines 1.0
  http://www.w3.org/WAI/UA/WD-UAAG10-20010126
- W3C CSS-Validierungsservice
  http://jigsaw.w3.org/css-validator
- W3Cmarkup Validation Service
  http://validator.w3.org
- Web Accessibility Initiative (WAI)
  http://www.w3.org/WAI
- Web Content Accessibility Guidelines 2.0
  http://www.w3.org/TR/WCAG20
- XHTML 1.0
  http://www.w3.org/TR/xhtml1/
- Zugänglichkeitsmythen
  http://www.andreas-hollmann.de/netztips/zugaenglichkeit.html

7.5 Die Autoren

7.5.1 Ulrich Hacke

Ulrich Hacke ist Jahrgang 1972 und arbeitet seit Mitte der achtziger Jahre mit Computern. Nach Abitur und Zivildienst absolvierte er das Studium der Religionspädagogik an der Ev. Fachhochschule Hannover, wo er sich in seiner Diplomarbeit intensiv mit den gesellschaftlichen Auswirkungen der Computertechnologie des Informationszeitalters beschäftigte. Es folgte ein halbes Jahr stellvertretende Projektleitung für "kirche online" in Frankfurt am Main. 1997 begann er mit der Ausbildung zum Informatikassistenten (Softwaretechnologie) am b.i.b. Hannover. Anschliessend arbeitete er haupt- und freiberuflich im IT-Umfeld und gründete 1999 mit anderen das Unternehmen TRILOS IT-Dienstleistungen, dessen Geschäftsführer er heute ist.

7.5.2 David Maciejewski

David Maciejewski ist selbständiger Designer, Webdeveloper und Journalist und kommt ursprünglich aus dem Druck- und Designbereich. Schon als Jugendlicher wollte er im Bereich der Werbung arbeiten. HTML programmiert er seit 1998, PHP erst seit 2000. Zu seinen Kunden zählen Danzas Euronet / DHL / Deutsche Post, Mail Source / Schweizer Post, Fleischhauer Hannover, The Sourcing Group AG Zürich und wie bei Selbständigen üblich, viele kleine Noname-Firmen. Privat engagiert er sich in der Mittelalter-Szene und stellte im Juli 2003 das Onlinemagazin Mittelalter-Spectaculum.de mit Schwerpunkt Geschichte, Mittelalter online. Heute hat sich die Seite als Anlaufpunkt für Mittelalterinteressierte etabliert. Selbst in mittelalterliche Tracht begibt er sich allerdings nicht.