Pearson Education

ADDISON-WESLEY

HTML lernen

Pearson Education

ADDISON-WESLEY

Die Lernen-Reihe

In der Lernen-Reihe des Addison-Wesley Verlages sind die folgenden Titel bereits erschienen bzw. in Vorbereitung:

André Willms
C-Programmierung lernen
432 Seiten, ISBN 3-8273-1405-4

André Willms
C++-Programmierung lernen
408 Seiten, ISBN 3-8273-1342-2

Guido Lang, Andreas Bohne
Delphi 5 lernen
432 Seiten, ISBN 3-8273-1571-9

Judy Bishop
Java lernen
636 Seiten, ISBN 3-8273-1605-9

Michael Schilli
Perl 5 lernen
ca. 400 Seiten, ISBN 3-8273-1650-9

Michael Ebner
SQL lernen
336 Seiten, ISBN 3-8273-1515-8

René Martin
VBA mit Word 2000 lernen
412 Seiten, ISBN 3-8273-1550-6

René Martin
VBA mit Office 2000 lernen
576 Seiten, ISBN 3-8273-1549-2

Patrizia Sabrina Prudenzi
VBA mit Excel 2000 lernen
512 Seiten, ISBN 3-8273-1572-7

Patrizia Sabrina Prudenzi, Dirk Walter
VBA mit Access 2000 lernen
680 Seiten, ISBN 3-8273-1573-5

Dirk Abels
Visual Basic 6 lernen
425 Seiten, ISBN 3-8273-1371-6

Pearson Education

ADDISON-WESLEY

Walter Herglotz

HTML lernen

anfangen, anwenden, verstehen

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

**Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich**

Umwelthinweis:
Dieses Produkt wurde auf chlorfrei gebleichtem Papier gedruckt.
Die Einschrumpffolie – zum Schutz vor Verschmutzung – ist aus umweltverträglichem und recyclingfähigem PE-Material.

10 9 8 7 6 5 4 3 2 1

04 03 02 01

ISBN 3-8273-1717-7

Einbandgestaltung:	Barbara Thoben, Köln
Lektorat:	Christina Gibbs, cgibbs@pearson.de
Korrektorat:	Ulrike Oswald, Forstern
Herstellung:	Ulrike Hempel, uhempel@pearson.de
Farbtafel:	Harald Taglinger. In: Jetzt lerne ich HTML. Markt+Technik, ISBN: 3-8272-5717-4.
Satz:	mediaService, Siegen
Druck und Verarbeitung:	Bercker, Kevelaer

Printed in Germany

I Inhaltsverzeichnis

Pearson Education

ADDISON-WESLEY

V Vorwort

Haben Sie heute schon im Internet gesurft?

Wenn ja, dann sind die Chancen groß, dass Sie auf Seiten aus verschiedenen Ländern gestoßen sind. Vielleicht waren Sie bei einer deutschen Zeitung im Sportteil, bei einem Hersteller für Scanner in Taiwan, um den neuesten Treiber zu suchen oder beim großen Chip-Hersteller Intel in den USA.

So schön und interessant das Surfen ist – es gibt noch etwas Spannenderes: selbst Seiten gestalten und auch im Netz anderen zur Verfügung stellen.

Die Grundausrüstung haben Sie wahrscheinlich schon, um zumindest einfache Seiten selbst erstellen zu können. Es fehlt vielleicht noch die Kenntnis der Sprache, die man braucht, um weltweit lesbare Seiten zu produzieren – HTML. Aber das können wir mit diesem Buch ändern. Nach wenigen Seiten werden wir die erste einfache HTML-Datei geschrieben haben und arbeiten uns dann durch die Möglichkeiten der Sprache.

Warum HTML? Nun, HTML (= Hypertext Markup Language) ist der Standard für Webseiten, den ein Browser, das Leseprogramm für Webseiten versteht. Mit HTML kann man die Seiten produzieren, die Verweise zum Klicken enthalten.

Neben der eigentlichen Sprache HTML brauchen wir noch ein wenig Wissen zum gesamten Internet. Auch diesem Bereich wollen wir uns in diesem Buch widmen.

Wenn sich alle an den Standard halten, dann können alle Menschen, die mit völlig unterschiedlichen Geräten und Programmen auf die Daten im weltumspannenden *WWW* (World Wide Web / weltweites Netz) zugreifen, diese auch richtig lesen. Alle Daten, seien es nun Texte, Bilder oder anderes, werden dazu mit Hilfe des HTML-Standards verpackt.

HTML bringt uns die Möglichkeit, Texte und andere Informationen in einer weltweit einheitlichen Weise zu produzieren und sie allen anderen Benutzern eines vernetzten Computers unabhängig vom Hersteller des Computers oder des verwendeten Betriebssystems zugänglich zu machen. Voraussetzung ist dabei nur, dass wir uns an die allgemeinen Spielregeln der Standards halten.

Das *Surfen*, das Wandern in den Seiten des WWW, kennen viele und sicher haben Sie auch schon viele interessante Informationen, Bilder, Töne oder Filme gefunden. Aber das alles gab es schon vorher. Gehen Sie doch einmal in eine gute Bibliothek. Dort stehen mehr Bücher, Zeichnungen oder Videos, als ein Einzelner von uns jemals aufnehmen kann.

Das Besondere an HTML ist der andere Weg: Sie können selber produzieren, publizieren und andere informieren. Und Ihre Seiten stehen von heute auf morgen sofort den Millionen von Benutzern des WWW in der ganzen Welt zur Verfügung.

Ein paar Probleme gibt es schon noch. Ich weiß nicht, wie viele Menschen in Frankreich, Italien, Spanien oder Australien Deutsch verstehen. Aber sie können trotzdem Ihre Seiten sehen. Und wenn die für den einen oder anderen wichtig und interessant sind, dann wird der Leser schon Wege finden, die Sprachbarrieren zu überwinden. Schließlich lesen viele von uns auch englische Webseiten, wenn sie interessante und für uns wichtige Inhalte anbieten. Oder Sie helfen Ihren Lesern durch mehrsprachige Seiten.

Die Grundlage aller Seiten ist HTML – und das lernen wir zusammen Schritt für Schritt in diesem Buch. Danach brauchen Sie eigentlich nur noch eine Idee, was Sie anderen mitteilen möchten.

Dieses Buch wird Sie also unterstützen beim eigenen Publizieren. Und falls Sie nicht der ganzen Welt Informationen anbieten wollen, dann kann Ihnen das Buch helfen, Verständnis für die Welt des WWW zu entwickeln, mit Webseiten Ordnungshilfen für eine volle Festplatte zu schreiben oder einfach ein wenig Spaß am Experimentieren zu haben.

Dieses Buch lebt von Experimenten, die Sie alle nachvollziehen und tatsächlich ausprobieren können. Alle Beispiele finden Sie auf der Webseite des Autors unter *www.walter-herglotz.de*.

Viel Erfolg, spannende Ideen und weltweite Kontakte daraus wünscht Ihnen

Walter Herglotz

»Es gibt nichts Gutes, außer man tut es.«
Erich Kästner

v.1 Die Icons in diesem Buch

Um Ihnen die Orientierung in diesem Buch zu erleichtern, haben wir den Text in bestimmte Funktionsabschnitte gegliedert und diese durch entsprechende Symbole oder Icons gekennzeichnet. Folgende Icons finden Verwendung:

Beispiele helfen Ihnen, sich schneller und sicher im Feld der HTML-Programmierung zu orientieren. Sie werden darum mit diesem Icon gekennzeichnet.

Manches ist von besonderer Bedeutung und verdient darum auch, besonders hervorgehoben zu werden. Solche **Hinweise** sind sehr nützlich, sie bringen einen geschwinder ans Ziel.

Manches geht ganz leicht. Wenn man nur weiß, wie. **Tipps und Tricks** finden Sie in den Abschnitten, wo dieses Icon steht.

Übungen helfen Ihnen, das Gelernte gleich in die Praxis umzusetzen und Ihre Kenntnisse zu vertiefen.

1 Einleitung

Liebe Leserin, lieber Leser,

nun stehen Sie und ich vor einer wichtigen Frage: fangen wir mit der Historie an, den technischen Hintergründen, surfen wir zuerst im Internet oder machen wir zusammen ein Beispiel?

Im Zweifel für die Praxis lautet das Motto dieses Buches.

Unsere Experimente sollen Ihnen ermöglichen, Aufbau, Idee und Darstellung der Webseiten zu erkennen und nachzuvollziehen. Wenn immer möglich werden wir uns dabei den allgemeinverbindlichen Standards orientieren.

Also fangen wir mit unserem ersten Experiment an. Nur eine Bitte möchte ich an dieser Stelle äußern. Sie sollten wissen, wie man Textdateien erstellt, Dateien kopiert und wie man in einem Programm eine Datei lädt. Auch die Begriffe Verzeichnis, übergeordnetes Verzeichnis und Pfad werden wir brauchen. Natürlich werden alle Begriffe erklärt. Aber vielleicht würde es helfen, wenn Sie sich im Zweifel noch ein Buch über Betriebssysteme besorgen, in dem diese Begriffe ausführlich dargestellt sind. Danke! Und nun geht's los!

Starten Sie Ihren Rechner. Für das erste Experiment brauchen wir nur einen ganz einfachen Editor und einen Browser.

Auch wenn Sie die Beispiele aus dem Internet benutzen können, wäre es doch schöner, wenn Sie die wenigen Zeilen eintippen würden. Mir hat es immer mehr geholfen, den Text abzutippen, Tippfehler zu korrigieren, Fehler zu suchen und erst dann das Ergebnis zu haben. Vielleicht lernen Sie auch so ähnlich wie ich.

Alle Beispiele, die Bilder in den Beispielen und weitere Informationen finden Sie auch im Internet auf den Web-Seiten des Autors. Besuchen Sie dazu den Server mit der Adresse *www.walter-herglotz.de*. Sie finden dort bei den Büchern die Möglichkeit, alle Beispiele auf Ihren eigenen Rechner zu kopieren.

Die Beispiele sind getestet und sollten mit allen gängigen Browsern funktionieren. Sollten Sie derzeit einen älteren Browser benutzen, wäre es schön, wenn Sie zumindest für die Experimente der späteren Kapitel einen heute üblichen Browser benutzen. Bei Browsern der Firmen Netscape und Microsoft haben die Browser dann eine Versionsnummer "4.0" oder besser.

Die Version erhalten Sie im Netscape-Browser angezeigt, wenn Sie im HILFE-Menü den Eintrag ÜBER COMMUNICATOR wählen.

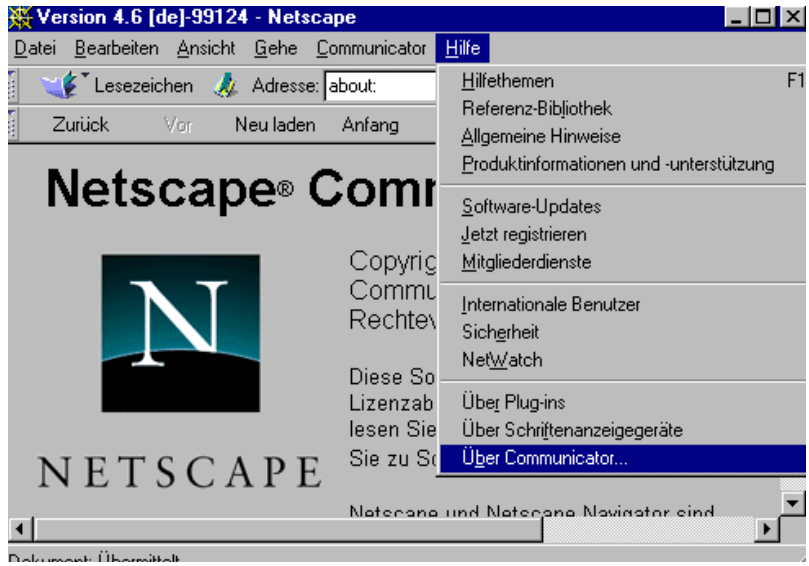

Abbildung 1.1: Browser-Information bei Netscape

Im Internet Explorer verbirgt sich die Browser Information hinter dem Menü ?, der ebenfalls zur Hilfe führt. Hier ist es der Menüeintrag INFO, der die notwendigen Informationen zum verwendeten Browser angibt.

Abbildung 1.2: Browser-Information bei Microsoft

Sollten Sie einen der zahlreichen anderen Browser verwenden, wäre es schön, wenn er weitestgehend den HTML Sprachstandard 4.0 unterstützt.

Welchen Editor Sie verwenden, ist nicht so wichtig. Unter Windows kann es der NOTEPAD.EXE sein, den man über START/PROGRAMME/ZUBEHÖR und dort unter EDITOR findet, oder der WORDPAD, der im gleichen Verzeichnis steht. Sollten Sie die Editoren nicht finden, dann wurden sie nicht mitinstalliert. Das kann man aber auch nachholen. Dazu nutzt man in der *Systemsteuerung* die Anwendung SOFTWARE. Unter *Linux* arbeitet der `vi` oder der `joe` genauso gut.

Diese Editoren sollen reinen Text abspeichern können. Das Gegenstück zu einem reinen Text ist z.B. eine Windows DOC-Datei. Die enthält neben den Buchstaben, die Sie tippen, noch eine ganze Reihe von Formatierungsinformationen. Ihr *Editor* sollte also auf keinen Fall einige Kilobyte auf die Festplatte speichern, wenn Sie nur zehn Buchstaben getippt haben. Wir benötigen nur den eingegebenen HTML-Code in der Datei.

Für den Moment wollen wir uns mit einem simplen Texteditor begnügen. Das reicht solange man einfache Webseiten für Lernzwecke erstellt. Später werden wir uns auch mit Editoren beschäftigen, die auf die Erstellung von Webseiten spezialisiert sind.

Manche Leser werden vielleicht erwarten, dass wir ausschließlich mit einem Editor arbeiten, der eine graphische Benutzerführung hat. Möglicherweise denken Sie dabei an Programme wie *Winword*. Das Problem dieser Programme ist es, dass sie alles verdecken, was zum Verständnis der zugrunde liegenden Sprache benötigt wird. So schön eine graphische Benutzerschnittstelle ist, so schwierig macht sie es zu

verstehen, was eigentlich passiert. Denn es ist ja die große Aufgabe der graphischen Benutzerführung, alles leicht bedienbar zu machen. Diese Leser bitte ich um etwas Geduld und die Bereitschaft, sich auf Dinge einzulassen, die ihnen möglicherweise sogar ein wenig veraltet vorkommen. Der Lohn wird aber ein weit besseres Verständnis sein.

Aber nun zu unserem ersten HTML-Experiment.

Die Lösungen (oder jeweils eine mögliche Lösung) finden Sie jeweils am Ende eines Kapitels unter dem Kurznamen des Experimentes, der immer am Anfang der Aufgabe steht. Natürlich hat die Lösungsdatei noch die Kennung *.HTML.

E-EXP01:

Schreiben Sie eine korrekte HTML-Datei, die den Text „Willkommen auf meiner Webseite!“ anzeigt.

Am besten legen Sie ein eigenes Verzeichnis für Ihre Experimente an, tippen die Lösung mit Hilfe des gewünschten Editors ein und speichern den Text in einer Datei ab. Als Namen können Sie z.B. E-EXP01.HTML verwenden. Jeder andere Name funktioniert auch. Nur auf die Kennung der Datei sollten Sie achten. Wenn es Ihr Betriebssystem erlaubt, dann ist meist die beste Kennung *.HTML. Sollte Ihr Betriebssystem nur 3 Buchstaben nach dem Punkt im Dateinamen zulassen (z.B. früher bei *DOS* oder *Windows 3.1*), dann können Sie auch *.HTM verwenden.

Noch ein Tipp für Windows-Benutzer. Sie können beim Speichern der Datei den Namen in doppelte Anführungszeichen setzen, wenn Sie sicher sein wollen, dass das Betriebssystem nicht von sich aus Dateikennungen (z.B.: .TXT) anfügt.

Nun steht also eine Textdatei auf der Festplatte. Sie enthält unseren gewünschten Text und einige zusätzliche Angaben, deren Sinn wir gleich erkunden werden. Aber wieder zuerst zur Praxis. Schauen wir uns das Ergebnis als Webseite an. Zum Betrachten von Webseiten, egal woher diese Seiten kommen, brauchen wir ein Programm, das Browser genannt wird. Das englische Wort *to browse* bedeutet soviel wie *schmökern*, *durchsehen*, *blättern*. Das entsprechende Programm dient also dazu, im Internet zu schmökern oder eben zu *surfen*.

Davon gibt es ganze Reihe. Ein Browser, der auf verschiedenen Rechnern und Betriebssystemen zur Verfügung steht, kommt von der Firma Netscape. Je nach dem welche Version bei Ihnen installiert ist, heißt das Programm dann *Navigator* oder *Communicator*. Andere

Browser sind der *Internet Explorer* von Microsoft, *Opera* oder der integrierte Betrachter im Dateimanager *KFM* unter Linux. Ein ganz speziellen Browser soll auch noch erwähnt werden: der *Lynx* unter Linux/Unix. Das ist ein Browser, der ausschließlich Texte darstellt.

Starten Sie den Browser, den Sie auf Ihren Rechner als ersten finden. Öffnen Sie nun genau die Datei, die Sie oben mit Hilfe des Editors erstellt und abgespeichert haben. Wie war doch gleich der Name und der Name des Verzeichnisses?

Meist öffnet man eine Datei mit dem Menüpunkt DATEI/ÖFFNEN oder in der englischen Variante mit FILE/OPEN.

Und was steht am Bildschirm? Hoffentlich Ihr freundlicher Willkommensgruß. Wenn Sie den nicht sehen - dann kann ich Sie nur bitten, noch einmal jeden Schritt einzeln nachzuvollziehen. Sind die spitzen Klammern alle vorhanden? Sind die Anweisungen in den spitzen Klammern in der richtigen Reihenfolge? Sind auch die Schrägstriche vor den Anweisungen vorhanden? Zu jeder Anweisung gibt es hier eine zweite Anweisung, mit einem Schrägstrich davor.

Diese Seite kann natürlich auch Umlaute enthalten. Ändern Sie doch noch einmal den Text, so dass er Umlaute bekommt, speichern das Ergebnis und laden die Datei neu im Browser. Meist genügt auch der Knopf (engl. button) NEU LADEN, REFRESH, AKTUALISIEREN oder einer sinngemäß ähnlichen Beschriftung. Evtl. sollten Sie auch die Taste ⇧ (Umschaltung Groß-/Kleinschrift) gleichzeitig gedrückt halten.

Und nun sehen Sie sicherlich den Text mit Umlauten.

So spannend ist das gar nicht, meinen Sie? Klar, wir sind ja erst einmal nur auf dem eigenen Rechner geblieben. Und der ist ja für die eigene Sprache eingerichtet. Aber der gleiche Text könnte auch genauso gut auf einem iMac-Rechner in Texas gelesen werden – und dort sind die Umlaute doch etwas Besonderes. Mit Hilfe der Sprache HTML können wir weltweit lesbare Dateien erstellen.

Im Browser betrachten wir Webseiten. Diese Seiten können aus vielen Einzelinformationen (z.B. Bildern und Texten) zusammengesetzt werden. Eine Datei ist dagegen der Speicherplatz für einen einzelnen Text oder ein einzelnes Bild. Für uns entspricht jetzt am Anfang eine Seite genau einer HTML-Datei. Wir werden aber im Laufe des Buches sehen, wie eine angezeigte Seite aus mehreren Dateien zusammengesetzt werden kann.

Aber damit haben wir unsere erste Praxis hoffentlich erfolgreich geschafft und können ein wenig in der Historie und den Ideen hinter unserem Beispiel blättern.

1.1 Das Buch und die Browser-Kriege

Browser-Hersteller haben zumindest in der Vergangenheit aus sachlichen oder strategischen Gründen Spezialitäten in ihre Browser eingebaut, die nicht in den Standards enthalten waren.

Eine ganze Reihe von Erweiterungen waren so sinnvoll, dass sie später in den Standards aufgenommen wurden.

Umgekehrt gilt auch, dass viele Vorgaben der Standards noch gar nicht in den Browsern berücksichtigt wurden. Es gibt bei der wichtigsten Standardisierungsorganisation WWW-Consortium (*www.w3c.org*) Testbeispiele für korrekte Seiten, die kaum ein Browser richtig anzeigt.

So gerät jeder Autor, der über das Thema HTML schreibt, in eine unangenehme Zwickmühle. Hält man sich exakt an den Standard, gibt es mit unterschiedlichen Browsern unterschiedlichen Ärger. Hält man sich an einen Browser, kann ein anderer diese Spezialitäten gar nicht darstellen.

Und dann gibt es auch noch die schlimmste Diskussionsvariante, die entsteht, wenn man gar nicht mehr die Fakten sondern nur noch Herstellernamen betrachtet.

Lassen Sie uns bitte die Diskussionen weglassen, die nicht der Sache und dem Spaß am Publizieren dienen.

Unterstützen wir mit den Beschreibungen und Experimenten lieber die Grundideen des Webs:

- Offenheit
- Unabhängigkeit von einzelnen Herstellern (Rechner, Betriebssysteme, Browser)
- Unabhängigkeit von lokalen Spezialitäten
- Zukunftssicherheit durch Verwenden der Standards

Wir bleiben nach Möglichkeit also auf der Hauptstraße der Entwicklungen im Web, blicken ein wenig voraus, um kommende Entwicklungen schon zu berücksichtigen und berücksichtigen die Spezialitäten der Firmen nur in Einzelfällen.

Für uns soll eine gute Lösung sein, wenn sie für alle WWW-Benutzer anwendbar und sinnvoll ist.

Im nächsten Kapitel

Das WWW hat in wenigen Jahren eine erstaunliche Entwicklung durchgemacht. Und mit seiner Entwicklung auch sehr dazu beigetragen, dass die zugrunde liegenden Netzwerke enorm gewachsen sind.

Werfen wir daher einen kurzen Blick auf die kurze, aber spannende Historie des Webs.

1.2 Lösungen

E-EXP01:

```
<html>
<head>
<title>
E-EXP01: HTML-Lernen
</title>
</head>
<body>
Willkommen auf meiner Web-Seite!
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

2 Historie

2.1 Geschichte des WWW

Für viele ist der bunte Bildschirm des modernen Browsers mit verschiedenen Bildern, Werbebannern und Texten zur Selbstverständlichkeit geworden. Begonnen hat dies alles etwas einfacher. Viele Ideen und verschiedene Menschen haben über die möglichen Verfahren zur Gliederung und Strukturierung von Informationen nachgedacht.

Angefangen hat alles mit den Katalogen in den Bibliotheken und den Querverweisen und Literaturlisten in den Büchern. Jeder, der sich ein paar Jahre mit einem Thema beschäftigt hatte, kannte die Standardwerke dazu und noch einige weitere.

Das Netz, die Verknüpfung der Texte geschah aber immer im Kopf des Lesenden. Mit den digital gespeicherten Nachrichten kam auch die Idee auf, die Verweise in den eigentlichen Text zu integrieren. Auf den Webseiten von *http://www.w3c.org* finden sich Hinweise zu den Menschen, die Ideen und Entwürfe lange vor dem Zeitalter des PCs entwickelt haben.

Von Ted Nelson stammt die Idee des Hypertextes. Ein Hypertext vermischt den normalen Text mit Steuerelementen, die ihrerseits wieder auf weitere Informationen verweisen und so ein Netz bilden. Ein früherer Name für das WWW war *Mesh*, was eben auch *Netz* bedeutet. Aber WWW, das „World Wide Web“, oder auf Deutsch „das weltweite Netz“, klingt eben doch viel bedeutsamer, besonders wenn man es in voller Länge ausspricht.

Der Name des Buches von Ted Nelson, war *Literary Machines and Dream Machines* (Literarische Maschinen und Traum-Maschinen). Er sah auch Filme voraus, deren Ablauf durch den Benutzer gesteuert werden konnten. Ein paar Video-Spiele sind auf dem besten Weg dahin.

Ein anderer, manchen bekannter Vorläufer war das *HyperCard* System für Mac's. Bill Atkinson, der Entwickler des ersten Pixel-Zeichenprogramms *MacPaint*, schuf 1987 dieses Hypertext-Programm.

Und die beiden Forscher Tim Berners-Lee und Robert Caillau arbeiteten 1989/90 am Forschungszentrum CERN in der Nähe von Genf an Ideen, die schließlich zu den Grundbausteinen des WWW wurden. Ziel war es, eine vereinfachte Version der existierenden Layout-Sprache SGML zu entwickeln und diese Sprache mit einem einfachen Transport-Protokoll zu verbinden. Schließlich sollten die Informationen auf verschiedenen Rechnern bei CERN und außerhalb liegen können.

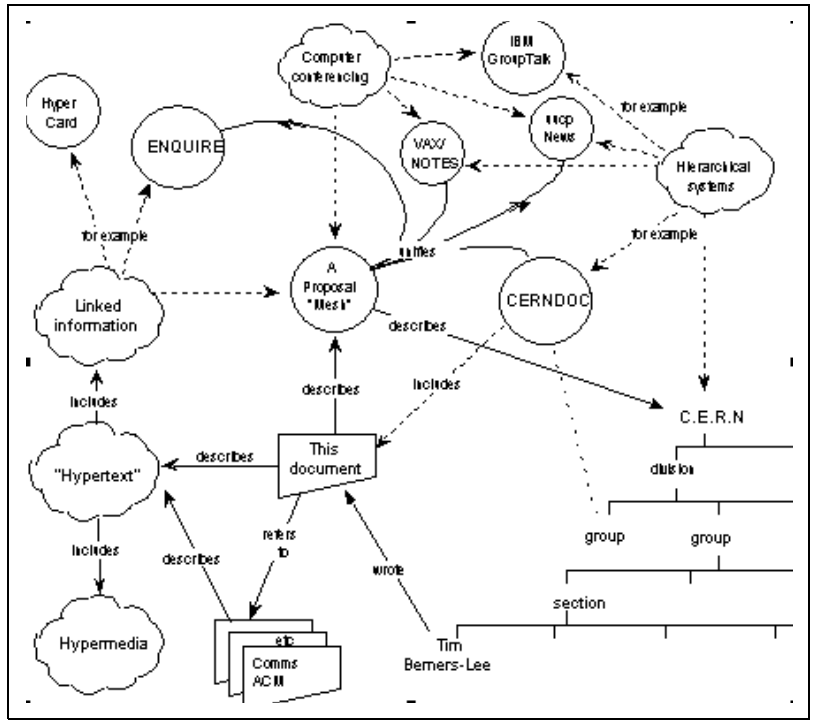

Abbildung 2.1: Skizze der Informationsstruktur am CERN; aus: "Information Management: A Proposal", Tim Berners-Lee, CERN March 1989, May 1990

Wichtig war den Entwicklern, den andauernden Informationsverlust zu begrenzen, der gerade an einer Forschungseinrichtung so leicht entsteht. Die Forscher waren ja meist nur für eine gewisse Zeit zum Forschungszentrum abgeordnet. Nach einigen Jahren oder manchmal schon nach Monaten kehrten Sie an ihre Universität zurück. Und am CERN mussten dadurch immer wieder neue Mitarbeiter eingewie-

sen werden und sich mit den bereits erarbeiteten Ergebnissen beschäftigen. Dieser Prozess sollte durch eine allgemein zugängliche, einheitliche Veröffentlichung zusammen mit den notwendigen Verweisen auf Vorarbeiten, beschleunigt werden.

Die Grundbausteine der Webidee waren:

- einheitliche Veröffentlichung mit Hilfe einer einfachen Publikationssprache.
- einfacher, einheitlicher Zugang über Systemgrenzen hinweg
- Veröffentlichungen sollte jeder Forscher lesen, schreiben und durch Verweise verbinden können
- Die Sprache (deutsch, englisch etc.) sollte keine technische Barriere darstellen.

Die textbasierte Version des WWW-Systems startete am CERN im Jahre 1991. Die Zahl der Nutzer war natürlich noch sehr klein.

In den Jahren entwickelten verschiedene Autoren Browser, die textbasiert für Konsolen/Monitore (heute in der Windows-Welt als DOS-Box bekannt) geschrieben wurden oder als graphische Anwendung für die UNIX-Oberfläche *X-Window*. Beispiele sind *Erwise* und *Viola* aus dem Jahr 1992.

Ganz spannend sind auch die Zahlen, die den Datentransport für das WWW auf dem großen Hauptstrang des Internets in den USA angeben (NSF backbone). Im März 1993 waren es 0,1% des Gesamtverkehrs; im September des gleichen Jahres bereits 1%. Die Anzahl der angeschlossenen Server (Rechner mit freigegebenen Dateien) lag im November 1993 bei 200. In diesem Jahr hat auch die Kommission der Europäischen Union mit ihrem Sekretariat DG XIII zusammen mit CERN und der Fraunhofer Gesellschaft begonnen, das WWW zur Verteilung von technischer Information zu nutzen.

Die Entwicklung des WWW ist also seit ihrem Anfang eine internationale Entwicklung.

Ein besonders wichtiger Treiber in der Entwicklung sei noch genannt: Tim O'Reilly, der mit seinem gleichnamige Verlag „O'Reilly" und als Ausrichter von Konferenzen viel zur die Verbreitung der Webidee beigetragen hat.

In Genf fand dann 1994 die erste WWW-Konferenz statt. Die Veranstaltung war mit 800 Angemeldeten hoffnungslos überbucht – nur die Hälfte konnte teilnehmen.

1994 waren es dann schon 1500 Server, die den WWW-Dienst anboten. Und Hr. Bangemann stellt den Plan der Europäischen Kommission für den Informations-Superhighway vor. Vielleicht fällt Ihnen auf, dass das lange vor den Reden des amerikanischen Vizepräsidenten Al Gore war. Die Last auf den WWW-Server hat sich nun schon in den 3 Jahren von 1991 bis 1994 vertausendfacht. Auch die Organisation W3C wird in Zusammenarbeit zwischen CERN in Europa und MIT in den USA ins Leben gerufen.

Und 1994 ist das Jahr, in dem der Urvater aller heute verwendeten graphischen Browser, *Mosaic,* von seinen Entwicklern in eine eigenständige Firma eingebracht wurde. Sein heutiger Name ist *Netscape*. Der bekannteste Entwickler des Teams ist Marc Andreessen.

Die Geschichte seit dieser Zeit kennen Sie sicher aus eigener Erinnerung. Der Kampf der Browser-Hersteller hat viele Schlagzeilen geliefert. Aber was auch immer ein einzelner Hersteller produziert oder durchzusetzen versucht, der Maßstab bleibt die Arbeit des internationalen Gremiums W3C (World Wide Web Consortium), dem heute Organisationen aus Europa, USA und Japan angehören. (die europäische INRIA, die die Rolle von CERN übernommen hat, MIT für die USA und die Keio-Universität für Japan).

Die Geschichte des Netzes ist hier nur sehr, sehr kurz erwähnt. Vielleicht surfen Sie selbst ein wenig. Eine gute Ausgangsbasis zur Erforschung der Geschichte des Netzes ist die Homepage des internationalen Konsortiums *http://www.w3c.org*, von deren Webseiten auch die Daten dieser kurzen Historie stammen.

Im nächsten Kapitel

Haben Sie noch einen Moment Geduld, um zuerst noch einen Blick auf einige allgemeine Spielregeln und die Erklärung der Buchstaben „HTML“ zu werfen.

3 Auszeichnungsanweisungen

Im ersten Beispiel haben wir bereits eine vollwertige HTML-Datei kennen gelernt. Nehmen wir uns die Zeit, ein paar charakteristische Merkmale der Sprache anzusehen, bevor wir uns dann im nächsten Kapitel wieder mit einer Reihe von Experimenten beschäftigen.

3.1 Grundprinzipien

HTML folgt mehreren Grundprinzipien. Ein Grundprinzip verbirgt sich im Namen der Sprache. HTML steht für: Hypertext Markup Language (Hypertext Auszeichnungssprache). Der Begriff *Hypertext* bezeichnet dabei die Mischung aus Darstellung und Steuerung. Eine HTML-Seite kann also neben dem normalen Text weitere Elemente wie Bilder, Töne etc. Steuerungen enthalten, die auf andere Seiten verweisen. Die Steuerung kennen Sie vom Surfern im Internet oder von einem Hilfesystem. Es sind die anklickbaren Verweise (engl. *links*), die einen Sprung an eine andere Textstelle bewirken können. Die Verweise werden wir in einem späteren Kapitel besprechen.

3.1.1 Auszeichnungen

Die Wirkung der Auszeichnungsanweisungen kann man sich leicht mit Hilfe eines bekannten Stiftes aus dem Schreibwarenladen vorstellen. Kauft man einen *Marker*, einen Stift, mit dem man Textstellen farbig markieren kann, und würde z.B. alle wichtigen Überschriften auf einer Zeitungsseite in rot, alle nächstwichtigen in grün und die nicht ganz so wichtigen in blau markieren, dann hätte man auf der Seite einige logische Auszeichnungen angebracht.

Ähnlich funktioniert auch eine Auszeichnung in HTML. An der Stelle, an der eine logische Auszeichnung beginnen soll, fügt man in den Text die Start-Marke einer Anweisung ein. Meist folgt auf eine

Start-Marke Text, der mit der Auszeichnung näher bestimmt werden soll und eine passende Ende-Marke schließt die ganze Auszeichnung ab.

Eine HTML-Anweisung besteht also meist aus drei Teilen: einer Start-Marke, dem Inhalt und der Ende-Marke. In seltenen Fällen gibt es nur eine Start-Marke. Die Marken werden in spitzen Klammern geschrieben. Die Ende-Marke erhält innerhalb der spitzen Klammern einen „/“ (normaler Schrägstrich) vor dem Namen der Anweisung.

Beispiel:

‹head› .. **‹/head›**

Anweisungen funktionieren in HTML also ähnlich wie die Klammern in einer mathematischen Aufgabe. Auch dort kann man verschachteln. Aber man muss die Klammern sozusagen von innen her auflösen. Die zuletzt geöffnete Klammer muss zuerst wieder geschlossen werden.

Beispiel:

```
<head>
<title>
Der Titel
</title>
<head>
```

In unserem kleinen Beispiel wurde ein Titel in einem Kopfbereich verschachtelt.

Betrachten Sie möglichst jeden der ausgezeichneten (oder anders: markierten) Bereiche wie einen Container, der wertvolle Informationen enthält. Diese Container können ineinander verschachtelt werden.

Damit haben wir schon drei der vier Buchstaben von HTML erklärt. *HT* steht für den *HyperText* und das *M* steht für *markup* (das logische Markieren). Der letzte Buchstabe, das *L*, steht für *language* / *Sprache*. Eine Sprache für Computeranwendungen ist ein Satz von Spielregeln, den man zur Lösung der gestellten Aufgabe benutzen kann. In unserem Fall sind die Spielregeln eben die Auszeichnungs-Anweisungen zusammen mit einigen weiteren Vorgaben zur Schreibweise, zur Verschachtelung etc.

3.1.2 Trennung zwischen Autor und Darstellung

Ein weiteres Grundprinzip neben der Verwendung der Sprache HTML ist die Trennung zwischen Autor und Darstellung. Eigentlich sollte in einer HTML-Datei nichts über die Formatierung stehen. Aber hier geriet HTML zwischen die Fronten derjenigen, die eine immer bessere und genauere Darstellung meist für Werbezwecke wollten, und denjenigen, die das ganze System im Auge haben.

In den Anfängen des WWW, des World Wide Web, hat der Autor nur sehr einfache HTML-Befehle (oder Anweisungen) zur Verfügung gehabt. Damit war die Trennung automatisch gegeben. Nur der Browser hat sich um die Darstellung gekümmert.

Je umfangreicher in den verschiedenen Versionen der Sprache die Möglichkeiten wurden, desto unschärfer wurde diese Trennung. Der Höhepunkt wurde mit *HTML 3.2* erreicht. Danach besann man sich wieder stärker auf das Grundprinzip der Trennung zwischen Autor und Darstellung.

Man nennt übrigens die Art der Schreibweise für HTML auch das Freitext-Format. Sie können gerne Leerzeichen und Zeilenschaltungen einfügen. Im Browser werden sie zumeist entfernt, denn es ist ja der Browser, der formatiert, nicht der Autor.

Gestalten Sie also Ihre ersten Beispiele nach Wunsch. Zeilenschaltungen, Leerzeichen und Tabulatoren verlängern zwar die Datei, machen Sie aber gleichzeitig doch viel leichter lesbar.

3.1.3 Korrekte Kodierung der Anweisungen

In der Vergangenheit gingen Autoren und HTML-Editoren sehr freizügig mit den Regeln um. Das geschah sicher nicht aus bösem Willen oder gar Unkenntnis. Sogar der frühere Standard hatte nichts dagegen, wenn so klare und einfache Anweisungen wie die `<head>`- oder `<html>`- Anweisung fehlten oder ohne schließende Ende-Marke geschrieben wurden.

Noch schlimmer wird die Situation bei anderen Anweisungen wie `<p>`- oder `<li>`, die wir noch detailliert kennen lernen werden. Die Ende-Marken wurden kaum geschrieben und auch heute noch entfernen manche HTML-Editoren heimlich etwa vorhandene.

Wir wollen diese Schlamperei nicht unterstützen. Zum einen ist es irgendwie unsauber und damit auch unschön, zum anderen stehen solche Unsauberkeiten der zukünftigen Handhabung entgegen. Je komplexer und umfangreicher die verschiedenen, ineinander grei-

fenden Standards werden, desto wichtiger werden Editoren, Hilfsmittel, Generatoren, Datenbanken und Autorensysteme und damit eine formal korrekte Sprache. Lassen Sie sich daher nicht irritieren, wenn hier im Buch manchmal mehr Korrektheit gefordert wird, als heutige Umgebungen erzwingen oder voraussetzen.

Mit einem lachenden und einem weinenden Auge betrachte ich die Entwicklung, denn es war ein tolles Gefühl, mit ein paar Befehlen und einem Editor Webseiten zu erstellen, die in der ganzen Welt sichtbar sind. Viel von dieser Stimmung können Sie auch heute nachvollziehen. Und deshalb finden Sie in jedem Kapitel eine Reihe von Experimenten, die Ihnen Stück für Stück die Welt von HTML erschließen.

3.2 Kommentare in HTML

Wollen Sie in die HTML-Dateien eigene Bemerkungen und Zusatzinformationen einfügen, dann bietet HTML auch eine Kommentaranweisung an.

Der Beginn eines Kommentars wird mit `<!--` eingeleitet. Es ist vielleicht etwas ungewöhnlich, den Kommentar mit einem Ausrufezeichen und zwei Trennstrichen einzuleiten. Die Ende-Marke des Kommentars werden dann wieder mit zwei Trennstrichen gebildet. Das liest sich dann `-->`. Der Kommentar wird mit einer offenen spitzen Klammer begonnen und mit einer schließenden spitzen Klammer beendet. Es gibt damit bei Kommentaren keine schließende Marke mit einem vorangesetzten Schrägstrich.

Es hat sich als praktisch erwiesen, Kommentaren eine eigene Zeile zu spendieren.

3.2.1 Versionsangaben

In der ersten Zeile der meisten HTML-Seiten werden Sie einen speziellen Eintrag finden, der die verwendete Sprachversion angibt. Diese Angabe ist eine Vorgabe der "Muttersprache" von HTML: *SGML* (standardized general markup language), einer sehr mächtigen Seitenbeschreibungssprache. *HTML* ist sozusagen der kleine Ableger davon.

Grundlage für die Beispiele dieses Buches ist die HTML Version 4.0, die aber noch Anweisungen enthält, die aus früheren Versionen stammen und deren Verwendung heute nicht mehr empfohlen wird. Da aber Millionen von Webseiten existieren, werden sicher auch zukünftige Browser ältere Webseiten anzeigen können.

Beispiele:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
   "http://www.w3.org/TR/REC-html40/strict.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/REC-html40/loose.dtd">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
   "http://www.w3.org/TR/REC-html40/frameset.dtd">
```

Der Eintrag beschreibt eine sogenannte *document type definition* (Dokumenten Aufbau Definition). Die Anweisung beginnt mit `<!DOCTYPE...>`. Das erste Beispiel gibt an, dass es sich um ein HTML-Dokument der Sprachversion 4.0 handelt. Die Sprachversion wird um das Wort *transitional* ergänzt. Die Sprache der Definition ist "EN" für Englisch.

Es gibt drei Varianten der Definition des Dokumentes: *strict*, *transitional* und *frameset*. Die strikte Definition (*strict*) enthält nur die Anweisungen der Sprachversion 4.0, die dort als zukünftig gültig angesehen werden.

Die Übergangsregelung (*transitional*) enthält neben den eigentlichen Anweisungen der Version 4.0 noch weitere Anweisungen, die aus früheren Versionen der Sprache stammen und deren Verwendung heute nicht mehr empfohlen wird. Diese Angabe wird heute die meist gebräuchliche sein.

Die letzte der drei Varianten ergänzt die Übergangsregelung zusätzlich um Angaben zu den sogenannten *frames* (Rahmen). Diesem Rahmen ist ein eigenes Kapitel gewidmet. Man kann damit den Anzeigebereich des Browsers noch einmal unterteilen und mehrere HTML-Seiten dem Benutzer gleichzeitig präsentieren. Meist werden Rahmen verwendet, um bessere Navigationsmöglichkeiten für den Benutzer zu schaffen.

In den letzten drei Beispielen wird jeweils noch eine Zusatzinformation gegeben. Unter der angegebenen Adresse kann der Browser bei Bedarf die genaue Spezifikation der Definition finden.

Diese Angaben sind für die meisten Browser optional. HTML-Editoren werden aber zumeist diese Zeile einfügen.

Wir werden in unseren einfachen HTML-Beispielen am Anfang des Buches diese Definitionsangaben weglassen. Für dauerhafte und professionelle Auftritte sind sie aber empfohlen.

Im nächsten Kapitel

Nach den wichtigen Vorbemerkungen wollen wir nun einige Dateien schreiben, die bereits Gestaltungselemente von HTML nutzen.

4 Schreiben für die Welt

In unserem Eingangsbeispiel haben wir mit einem einfachen Editor eine Textdatei geschrieben und diese Textdatei in einem Browser betrachtet.

Hoffentlich ist das auch bei Ihnen an Ihrem eigenen Computer gut gelungen.

Aber so selbstverständlich ist das in Wirklichkeit leider nicht. Eine Datei, die Sie auf Ihrem Rechner erstellen, muss ja nicht unbedingt auf einem anderen Rechner in einem anderen Land lesbar sein. Neben den Rechnern, die Sie zu Hause benutzen, gibt es noch Großrechner, Steuerungsrechner oder Kleincomputer, die durchaus auch Seiten aus dem WWW darstellen können. Eine Datei, die im WWW zu sehen sein soll, muss unabhängig von einem bestimmten Betriebssystem, einem bestimmten Editor oder einer speziellen Landessprache sein.

4.1 Zeichensätze

Dazu müssen offene Spielregeln erdacht und eingehalten werden. Eine dieser Spielregeln ist ein gemeinsamer Zeichensatz. Ein Zeichen ist ein allgemeiner Begriff für einen Buchstaben, eine Ziffer, ein Sonderzeichen aber auch für ein Zeichen, das eine Steuerungsaufgabe für einen angeschlossenen Drucker hat. Ein Zeichensatz enthält eine festgelegte Anzahl von Zeichen in einer ebenfalls festgelegten Reihenfolge. Bekannte Zeichensätze sind *ASCII* (american standard code of information interchange / amerikanischer Standard Code für den Informationsaustausch), *ANSI* (american national standards institute / amerikanisches, nationales Standardisierungs-Institut), *ISO-8859-X* (internationale Standard Organisation) oder der immer stärker verbreitete *UniCode* (universeller Zeichensatz).

Der Zeichensatz *ISO-8859-1* ist auch unter dem Namen *Latin 1* bekannt.

Die Bedeutung eines gemeinsamen Zeichensatzes können wir uns an einem Beispiel ansehen.

Kennen Sie eigentlich noch diese hässlichen Adressen, in denen alle Umlaute fehlten und statt dessen mit Ersetzungen gearbeitet wurde? Aus einem Herrn Müller wurde eben dann der Herr Mueller. Und aus dem „ß“ wurden dann einmal zwei „s“ oder ein „sz“. So lange ist das noch gar nicht her.

Heute kann man auch als Amerikaner in Texas sitzen und dort mit einem iMac-Rechner Seiten aus dem World Wide Web (WWW) anschauen, die von einem deutschen Texter auf einem IBM-PC geschrieben wurden – mit richtigen Umlauten.

4.1.1 Frühere Zeichensätze

Wieso kam es denn überhaupt zu den Ersatzdarstellungen für Umlaute?

Des Rätsels Lösung liegt in der Dominanz der amerikanischen Computerhersteller, dem Wunsch der europäischen Hersteller, kompatibel zu sein, und den Problemen mit den verschiedenen Zeichensätzen.

Der bekannte Zeichensatz *ASCII* umfasst 127 verschiedene Zeichen für Buchstaben, Ziffern und Steuerzeichen und hat keinen Raum für Umlaute. Jedes Zeichen benötigt nur 7-Bits statt der üblichen 8-Bits eines ganzen Bytes, was einen erheblichen Gewinn bei der Übertragung von Nachrichten über langsame Telefonleitungen mit sich bringt. Je kürzer ein Zeichen ist, desto schneller funktioniert die Übertragung. Die Übertragung war deshalb notwendig, da die sehr teuren Computer zentral aufgestellt wurden und die Benutzer oft die Daten über weite Strecken schicken mussten.

Es gibt noch weitere Gründe, nur wenige Zeichen zu verwenden. Frühe Schnelldrucker brauchten Druckketten, auf denen jedes Zeichen angebracht war. Je weniger Zeichen vorhanden waren, desto kürzer konnten diese Ketten sein und damit auch schneller drucken.

Heute schmunzeln wir inmitten unserer Tintenstrahl- und Laserdrucker über diese Probleme.

Aber was hat das mit unserem WWW und HTML zu tun? Auch jeder Text, den Sie eintippen, muss mit Hilfe eines Zeichensatzes (*ASCII*, *Latin 1*, *UniCode*...) dargestellt werden. Innen im Computer gibt es ja nur Zahlen. Also muss so ein Zeichensatz festlegen, welche Zahl zu welchem Buchstaben gehört. Und da gibt es deutliche Unterschiede.

IBM-PC Benutzer können hier unser Experiment F-EXP01 mitmachen.

F-EXP01: Zeichensätze

Erstellen Sie mit einem Editor unter Windows HTML-Datei nach dem ersten Muster. Sie soll auch Umlaute enthalten. Speichern Sie diese Datei mit der Kennung unter dem Namen F-EXP01.HTML ab.

Betrachten Sie die Datei in Ihrem Browser (*Explorer*, *Navigator*, *Lynx*, *Opera*, ...).

Öffnen Sie nun eine DOS-Box (Eingabeaufforderung). Bei meinem Rechner mit Windows NT kann dies über START/PROGRAMME/ EINGABEAUFFORDERUNG geschehen.

Laden Sie nun die gerade geschriebene Datei mit dem DOS-Editor *Edit* (`edit f-exp01.html`). Sollten Sie die spartanische Oberfläche der Eingabeaufforderung (auch DOS-Box oder Kommandozeile genannt) nicht gut kennen, dann helfen vielleicht die folgenden Tipps.

Mit der Eingabe des Laufwerksbuchstabens, eines Doppelpunktes und der ↵-Taste können Sie auf das gewünschte Laufwerk wechseln (z.B. auf das Laufwerk A:, das meist ein Diskettenlaufwerk ist, oder C:, das die erste Festplatte ist). Mit `CD Ihr_Verzeichnis` können Sie in ein neues Arbeitsverzeichnis wechseln. Der Befehl lässt sich natürlich wiederholen, wenn Sie Ihre Verzeichnisse tiefer verschachtelt haben. Mit `CD ..` (zwei einzelne Punkte) verlassen Sie es wieder. Auch dies kann mehrfach hintereinander geschehen. Damit bewegen wir uns immer weiter nach oben. Wenn Sie in dem Verzeichnis angekommen sind, in dem die gerade geschriebene Datei liegt, dann geben Sie einfach `edit Ihr_Dateiname` ein.

Suchen Sie doch nun einmal die Umlaute. Sie sind verschwunden und durch eigentümliche Sonderzeichen ersetzt worden, die Sie vielleicht noch nie gesehen haben.

Auf ein und demselben Rechner gibt es also unterschiedliche Darstellungen für Umlaute, da verschiedene Zeichensätze benutzt werden.

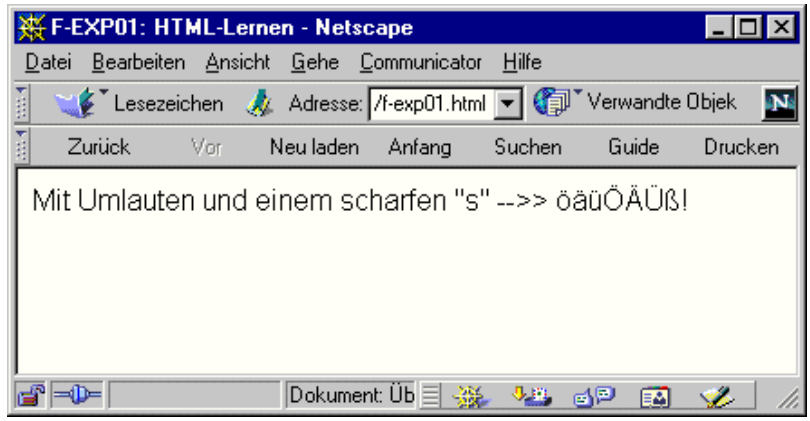

Abbildung 4.1: Im Browser dargestellt

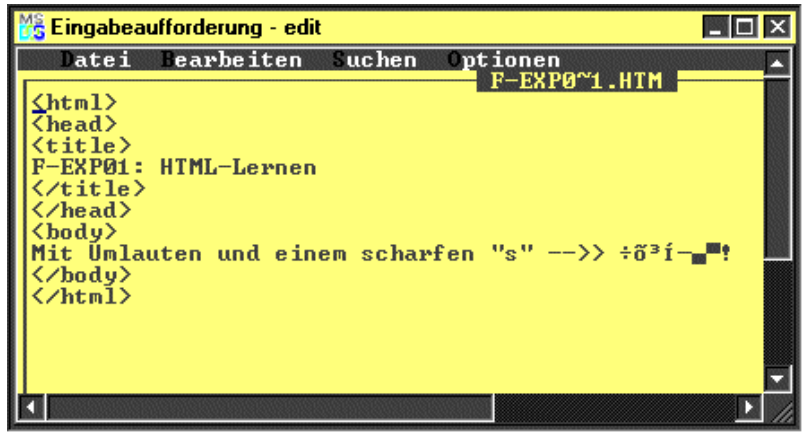

Abbildung 4.2: Unter DOS betrachtet

Für die Dateien (gelegentlich auch Dokumente genannt), die man mit HTML erstellt, wird ein einheitlicher Zeichensatz benutzt, der unabhängig vom Rechner ist. Dieser Zeichensatz ist ein ISO Standard (ISO: Internationale Standard Organisation, Genf). Es ist der Zeichensatz *ISO-8859-1*. Vielleicht kennen Sie ihn unter dem Namen *Latin 1*. Unter Microsoft-Betriebsystemen wird er auch *Western* genannt. Dieser Zeichensatz besteht aus den Zeichen des ASCII-Zeichensatzes und 128 weiteren Zeichen, die auch alle europäischen Umlaute und Sonderzeichen enthalten.

Die Wahl dieses Zeichensatzes lag aus zwei Gründen nahe. Das WWW wurde am CERN in Genf entwickelt. Dort waren europäische Umlaute und Akzente natürlich üblich. Daher lag es nahe den passenden, standardisierten Zeichensatz zu verwenden. Erst später ka-

men andere hinzu. Aber davon später. Wir werden noch gelegentlich auf das Thema des verwendeten Zeichensatzes zurückkommen.

HTML verwendet die folgende Spielregel: in den spitzen Klammern, die die Anweisungen umgeben, sollten nur Zeichen aus der ersten Hälfte des ISO-8859-1 Zeichensatzes verwendet werden. Diese Zeichen verwenden eine numerische Darstellung von 0 bis 127 (dezimal). Und es sind die gleichen Zeichen, die auch ASCII verwendet.

Zwischen den Anweisungen, also im eigentlichen Inhalt, ist zuerst einmal Latin-1 die Standardeinstellung.

Vielleicht merken wir uns einfach als Merkregel: Keine Umlaute in den Anweisungen, mit Umlauten im normalen Text.

Wenden wir uns jetzt erst einmal einem weiteren wichtigen Aspekt der Sprache zu – der Formatierung von Texten.

4.2 Textformatierung

Machen wir zuerst ein Experiment und diskutieren das Ergebnis im Anschluss. Vertrauen Sie bitte momentan den Vorgaben. Die Details zu den spitzen Klammern und den verwendeten Anweisungen (z.B. `head`) folgen natürlich noch.

F-EXP02: Formatierungen

Erstellen Sie mit Hilfe eines Editors eine HTML-Datei. Sie besteht wieder aus einem Kopf `<head>` und einem Hauptteil `<body>`. Als Text im Hauptteil sollten Sie nun ein Gedicht (ab-) tippen. Mein Lieblingsgedicht ist die Glocke von Schiller. Das ist auch lang genug, um noch ein paar weitere Experimente damit zu machen. Aber die Länge ist momentan gar nicht so wichtig. Wir benötigen nur ein Gedicht, das konventionell in Zeilen gegliedert ist.

Speichern Sie die HTML-Datei mit dem Gedicht und laden Sie die Datei in Ihren Browser.

Und nun sollte das schöne Gedicht auf einmal ohne eine der mühsam eingetippten Zeilen als einfacher Fließtext angezeigt werden. Die Zeilen sind verloren gegangen.

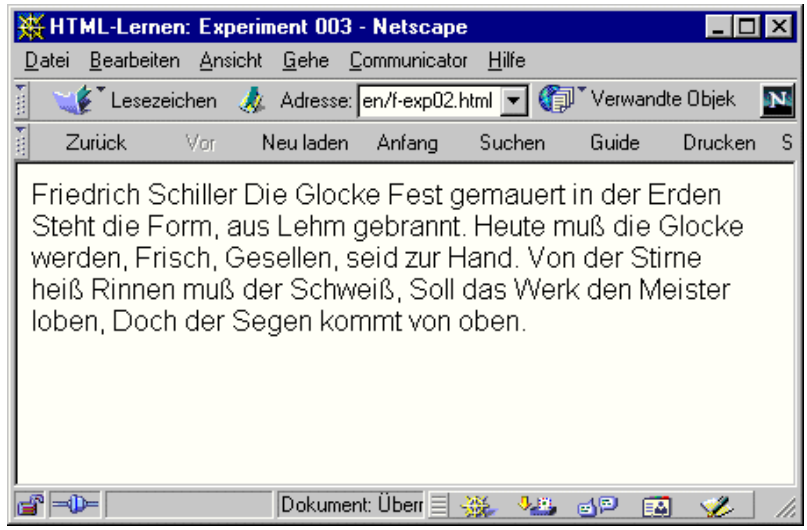

Abbildung 4.3: Gedicht ohne Zeilen

HTML ist eine Sprache, die den Aufbau von Seiten beschreibt. Sie ist aber nicht für die Darstellung verantwortlich. Die Darstellung übernimmt der Browser. Und der muss sich an seine eigene Umgebung anpassen können. Vielleicht haben Sie eine Einstellung mit geringer Auflösung gewählt (z.B. 640 x 480 Bildpunkte). Hier haben nur ein paar Zeilen auf dem Bildschirm Platz. Oder Sie haben eine sehr gute Auflösung gewählt, die viel mehr gleichzeitig anzeigen kann. In jedem Fall wird sich der Browser anpassen und Ihre Datei darstellen.

Noch ein Problem ist mit den üblichen Fenstern verbunden. Mal sind sie klein und schmal, mal füllen Sie den ganzen Bildschirm aus.

Verändern Sie doch einmal die Größe des Browserfensters und sehen sich an, wie verschieden der Text des Gedichtes durch den Browser formatiert wird, wenn das Anzeigefenster immer schmaler wird.

Damit sind wir mitten in einer sehr umfangreichen Diskussion gelandet. Ein wissenschaftlicher Autor, der eine Arbeit verbreiten möchte, wird kaum etwas dagegen haben, wenn der Browser seinen Fließtext so darstellt, wie er eben in das Fenster passt. Eine Firma, die viele Geld für einen schönen Auftritt im Internet bezahlt hat, würde umgekehrt die Seite genau so schön darstellen wollen, wie dies auf gedruckten Seiten der Fall ist.

Auf die Frage der Darstellung von Webseiten haben im Laufe der Zeit verschiedene HTML-Versionen unterschiedliche Antworten gegeben. Wir werden sie alle kennen lernen.

Für den Moment würde ich aber gerne einmal das Gedicht wieder in eine Form bringen, die der üblichen Darstellung entspricht.

Dazu bearbeiten wir unser Gedicht-Beispiel ein weiteres Mal und ergänzen die Zeilenschaltung `<br>` von HTML.

F-EXP03: Zeilenschaltungen für das Gedicht

Öffnen Sie die Datei mit dem Gedicht mit dem Editor. An den Stellen, an denen Sie vom Browser verlangen, dass er eine neue Zeile beginnt, fügen Sie jeweils den Befehl `<br>` ein. *Br* steht für *break / Umbruch*.

Betrachten Sie nun das Gedicht wieder im Browser.

Falls das Fenster breit genug ist, sehen Sie nun das Gedicht wieder in der gewohnten Form.

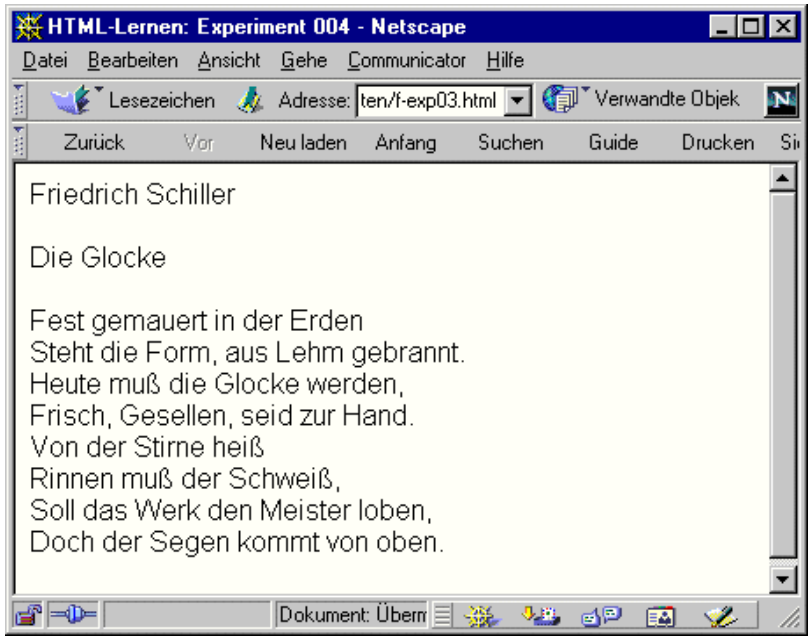

Abbildung 4.4: Gedicht in gewohnter Form

Mit expliziten Befehlen zum Zeilenumbruch haben wir nun unser Gedicht wieder in die bekannte Form gebracht. Aber natürlich wird der Browser eigenständig zusätzliche Zeilenumbrüche hinzufügen, wenn das Fenster zu schmal wird.

Im nächsten Kapitel

Bisher haben wir nur eine sehr einfache Formatierung von Texten mit Hilfe des erzwungenen Zeilenumbruchs geschafft. Und wir haben das Grundprinzip der Trennung zwischen logischer Auszeichnung und tatsächlicher Darstellung gesehen.

Die eingefügte Zeilenschaltung war eine Steuerung des Browsers. Es gibt noch eine Fülle von logischen Steuerungen für die Daten.

4.3 Lösungen

F-EXP01: Zeichensätze

```
<html>
<head>
<title>
F-EXP01: HTML-Lernen
</title>
</head>
<body>
Mit Umlauten und einem scharfen "s" -->> öäüÖÄÜß!
</body>
</html>
```

F-EXP02: Formatierungen

```
<html>
<head>
<title>
HTML-Lernen: Experiment 003
</title>
</head>
<body>
Friedrich Schiller
Die Glocke
Fest gemauert in der Erden
Steht die Form, aus Lehm gebrannt.
Heute muß die Glocke werden,
Frisch, Gesellen, seid zur Hand.
Von der Stirne heiß
Rinnen muß der Schweiß,
Soll das Werk den Meister loben,
Doch der Segen kommt von oben.
</body>
</html>
```

F-EXP03: Zeilenschaltungen für das Gedicht

```
<html>
<head>
<title>
HTML-Lernen: Experiment 004
</title>
</head>
<body>
Friedrich Schiller<br>
<br>
Die Glocke<br>
<br>
Fest gemauert in der Erden<br>
Steht die Form, aus Lehm gebrannt.<br>
Heute muß die Glocke werden,<br>
Frisch, Gesellen, seid zur Hand.<br>
Von der Stirne heiß<br>
Rinnen muß der Schweiß,<br>
Soll das Werk den Meister loben,<br>
Doch der Segen kommt von oben.<br>
<br>
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

5 Logische Textformatierungen

Bisher haben wir ja schon diskutiert, dass HTML eine Sprache ist, bei der streng zwischen den Aufgaben des Autors und den Aufgaben des Darstellungsprogramms unterschieden wird.

Sehen wir uns einige Beispiele an, in denen der Autor die logische Auszeichnung liefert und erst der Browser die aktuelle Darstellung festlegt.

5.1 Überschriften

Eine Überschrift wird durch ein Überschriftenanweisung dargestellt. Wir haben ja schon gesehen, dass die Anweisung dabei aus einer Start-Marke (engl. opening tag), dem Inhalt, der hier eine Überschrift ist, und der Ende-Marke (engl. closing tag) besteht. Die Marken der Anweisungen werden in HTML in spitze Klammern gesetzt. Die Überschrift besteht also nun aus drei Teilen: Start- und Ende-Marke und dem eigentlichen Text.

In unserem Gedicht-Beispiel wollen wir die Überschrift markieren.

L-EXP01: Auszeichnung von Überschriften

Laden Sie das Beispiel mit dem Gedicht in den Editor. Benutzen Sie dabei die `<h1>`-Anweisung. Setzen Sie also vor die Überschrift die Start-Marke `<h1>` und hinter die Überschrift die Ende-Marke `</h1>`.

Speichern Sie das Beispiel und zeigen es im Browser an. Wie sieht nun die Überschriftzeile aus? Welche Eigenschaften hat sie?

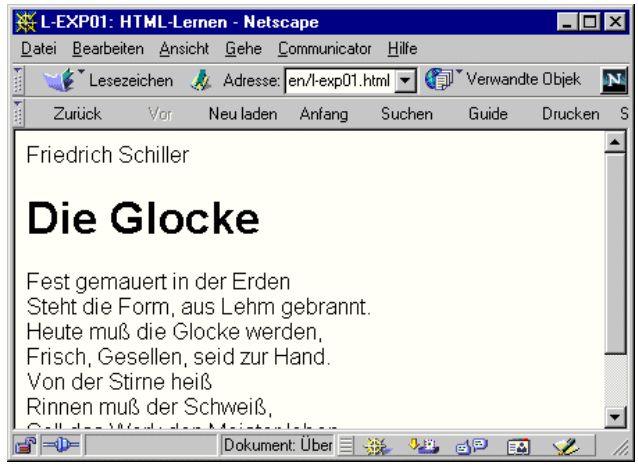

Abbildung 5.1: Große Überschrift

Überschriften werden nach ihrer Wichtigkeit unterschieden. Dabei gibt es eine Abstufung von 1 (sehr wichtig) bis 6 (weniger wichtig). Im Experiment 5 haben wir die wichtigste Überschrift gewählt.

Überschriften haben einige besondere Eigenschaften. Eine Überschrift besteht aus einer eigenen Zeile. Zusätzliche Zeilenschaltungen kann man rund um die Überschrift entfernen. HTML-Elemente, die immer auf einer eigenen Zeile beginnen, heißen auch Blöcke.

Der Browser wird die Schriftart (*Arial*, *Courier*, ...), die Größe (*12*,*14*,*16*, ... Punkt) und die anderen Schriftattribute (*halbfett*, *kursiv*, ...) selbständig festlegen und dabei die verschiedenen Gewichtungen der Überschriften berücksichtigen.

In unserem Beispiel kann es passieren, dass unterhalb und oberhalb der Überschrift viel Platz bleibt. Vielleicht stehen hier noch Leerzeilen, die wir mit der `<br>` erzeugt haben. Die unnötigen Zeilenschaltungen können wir entfernen.

Die Überschriften sollen übrigens nur logisch sinnvoll geschachtelt werden. Nach einer Überschrift der Wichtigkeit 4 darf keine Überschrift der Wichtigkeit 6 folgen. Man darf keine Wichtigkeitsebene auslassen.

L-EXP02: Darstellung von Überschriften

Erstellen Sie eine korrekte HTML-Datei, in der alle 6 Überschriftenarten vorkommen. Zeigen Sie die HTML-Datei in verschiedenen Browsern an.

Welche Überschrift wird mit welchen Mitteln (Schriftart, Größe, Attribut) dargestellt?

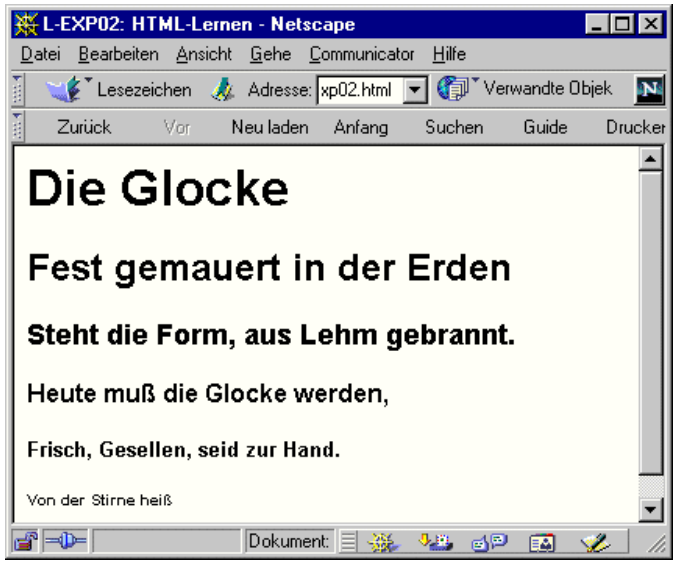

Abbildung 5.2: Überschriften in HTML

Je nach dem Betriebssystem, das Sie verwenden, werden Sie unterschiedliche Browser zur Verfügung haben. Solange Sie Browser für graphische Oberflächen verwenden, wird der Unterschied nicht allzu groß werden. Aber es gibt noch einen ganz speziellen Browser, den ich gerne noch einmal erwähnen würde.

Die ältesten Browser wurden für Textterminals geschrieben. Sie konnten nur Text darstellen. Und natürlich waren die Darstellungsmöglichkeiten dadurch sehr begrenzt. Unter *Linux* finden Sie sicher das Programm *lynx*, das ein Beispiel für so einen Browser ist. Allerdings ist *lynx* sehr schnell, da er Bilder, die in heutigen Webseiten enthalten sind, nicht lädt. Sollten Sie ausschließlich am textlichen Inhalt interessiert sein, wäre es sicher interessant, ihn einmal auszuprobieren.

Möglicherweise ergibt eine Suche im WWW, dass dieser Browser auch auf anderen Plattformen zur Verfügung steht.

Für manche mag das ein Ausflug in die Historie sein. Trotzdem hat der Browser für Dokumentationen und zum Testen durchaus noch seine Berechtigung.

5.2 Auszeichnung von Texten

Man kann Textstellen ähnlich den Überschriften mit logischen Auszeichnungen versehen. Die Auszeichnungen spiegeln die logische Verwendung wieder. Allerdings bleibt die Darstellung wieder dem verwendeten Browser überlassen.

Auszeichnung	Befehl
Zitat (Block)	`blockquote> ... </blockquote>`
Zitat (eingestreut) (HTML 4.0)	`<q> ... </q>`
Verweis auf (aber nicht aktiv)	`<cite> ... </cite>`
Programmlisting	`<code> ... </code>`
E-Mail-Adresse	`<address> ... </address>`
Text hervorheben	`<em> ... </em>`
Text deutlich hervorheben	`<strong> ... </strong>`
die Schrift klein machen	`<small> ... </small>`
die Schrift groß machen	`<big> ... </big>`
Schriftgröße steuern	`<font > ... </font>` Parameter: size =[+|-]1-7, color=#...
Abkürzung markieren (HTML 4.0)	`<abbr> ... </abbr>` Parameter: title
Kürzel markieren (HTML 4.0) z.B. WWW, DNS	`<acronym> ... </acronym>` Parameter: title

Tabelle 5.1: Logische Formatierungen

Die logischen Auszeichnungen können Sie der Tabelle entnehmen. Die meisten dürften einfach in der Anwendung sein. Bei allen Auszeichnungen handelt es ausschließlich um die Steuerung der Darstellung. Andere Dienste, die wir noch kennen lernen werden, wie Steuerungen zum Anklicken, sind damit nicht verbunden.

In der `<font>`-Auszeichnung taucht nun zum ersten Mal der Begriff des Parameters auf. Ein Parameter folgt dem eigentlichen Befehl innerhalb des spitzen Klammerpaares. Diese Parameter stehen nur in der Start-Marke. In den allermeisten Fällen besteht ein Parameter aus der Angabe des Parameternamens, der von einem Zuweisungszeichen („=“) und dem gewünschten Wert in Anführungszeichen gefolgt wird.

In wenigen Fällen genügt die Anwesenheit des Parameters, um eine Steuerung zu bewirken. Hier fungiert der Parameter wie ein Schalter.

Aber machen wir zuerst wieder ein Experiment.

L-EXP03: Arbeiten mit logischen Formatierungen

Erstellen Sie eine korrekte HTML-Datei, die die Auszeichnungen von `<blockquote>` bis `<big>` aus der Tabelle benutzt.

Notieren Sie, welche Möglichkeiten Ihr Browser zur Darstellung der jeweiligen Auszeichnung benutzt. Ein Tipp: `<address>` wird mit 2 „d“ geschrieben. Manchmal vertippt man sich hier.

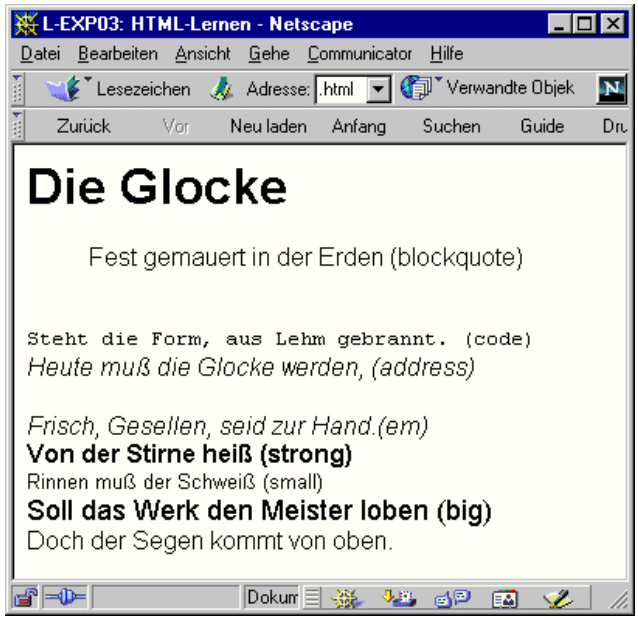

Abbildung 5.3: Logische Auszeichnungen

Die Auszeichnung *blockquote* wird für Zitate verwendet. Meist wird der Absatz mit dem Zitat eingerückt und ein wenig Platz darüber und darunter gelassen. Dieses Zitat wird also von normalen Text abgehoben. Meist stellt man ein derartiges Zitat einem längeren Text voraus.

Die zweite Auszeichnung für Zitate ist die `<q>`-Anweisung. Eigentlich sollten eingestreute Zitate vom Browser automatisch mit Anführungszeichen versehen werden. Viele Browser ignorieren jedoch diesen Teil der Anweisung.

Will man den Quelltext eines Programms, einen Hinweis auf ein Schlüsselwort einer Computersprache oder eine Bildschirmausgabe darstellen, eignet sich der `<code>`-Befehl. Er wird meist mit einer Schrift mit gleich großen Buchstaben dargestellt, um den bekannten Eindruck von einem Terminal oder einem älteren Computerdrucker wiederzugeben.

Möchten Sie auf einer Webseite eine E-Mail-Adresse angeben, dann können Sie sie mit der `<address>`-Anweisung formatieren. Meist verwenden die Browser dann zur Darstellung eine kursive Schrift.

Einzelne Worte oder kurze Abschnitte lassen sich mit `<em>` (für emphasize / betonen) oder `<strong>` (für stark betonen) hervorheben.

Und schließlich lässt sich mit `<small>` und `<big>` die Schriftgröße einstellen. Wie groß oder wie klein allerdings die Schrift wird, das weiß nur der Browser.

5.3 Schriftgrößen und Farben steuern

Mit der `<font>`-Anweisung kann man die Schriftgröße etwas präziser steuern. Die mögliche Schriftgröße im Browser wird auf einer willkürlichen Skala mit 1 bis 7 bewertet. Mit dem Parameter `size` kann man nun entweder einen der Werte vorgeben oder die momentan benutzte Schriftgröße nach oben oder unten verändern. Dazu setzt man einfach ein Vorzeichen vor den Wert. Insgesamt kann man aber die Skala von 1 bis 7 nicht verlassen. Die normale Schriftgröße wird hier durch einen Wert von 3 angegeben.

Eine `<font>`-Anweisung mit Parameter sieht also nun so aus:

```
<font size="+2">Text... </font>
```

L-EXP04: Setzen der Schriftgröße

Schreiben Sie eine HTML-Datei (oder ändern eine vorhandene Datei ab) mit den möglichen Schriftgrößen. Alle Größen sollen relativ angegeben werden.

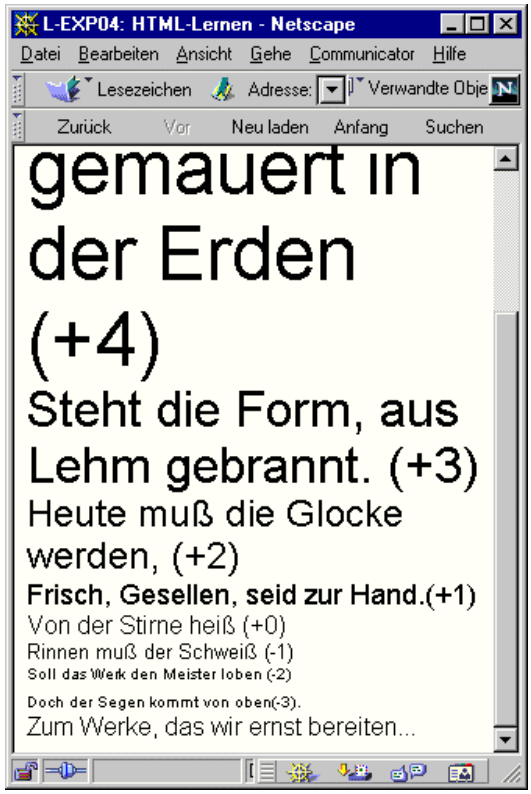

Abbildung 5.4: Relative Schriftgrößen

5.4 Schreibweisen von Anweisungen und Parametern

Alle Anweisungen (*font*, *head*, ...) und alle Parameternamen sollten klein geschrieben werden. Noch werden groß geschriebene Anweisungen und Parameter akzeptiert und in vielen Anleitung auch als richtig angegeben. In der Empfehlung zu XHTML, der Version des Standards vom Januar 2000, wird aber die einheitliche Kleinschrift vorgeschlagen.

Und noch etwas: alle Werte für Parameter sollten immer in Anführungszeichen gesetzt werden. Auch das ist heute normalerweise nur dann notwendig, wenn es die Möglichkeit zu Verwechslungen gibt. Das kann meist dann passieren, wenn der Wert ein Text aus mehreren Worten ist und damit Leerzeichen enthält.

5.5 HTML und Formatierungsanweisungen

Vielleicht fangen Sie sich schon an, ein wenig zu wundern. Wir haben doch diskutiert, dass der Autor den Inhalt und die logische Auszeichnung vorgibt und sich der Browser um die Darstellung kümmert. Was noch hinzukommen kann, ist, dass es nicht immer ein Browser sein muss. Eigentlich sollten die HTML-Seiten auch auf anderen Geräten mit und ohne Display auszugeben sein. Ganz dürfte dieser Wunsch des Standards nicht realisierbar sein. Aber anstelle der Textanzeige auf einem Bildschirm kann sicher ein Lesegerät für Blindenschrift angeschlossen werden.

Auf jeden Fall stehen wir hier am Beginn einer Entwicklung von HTML hin zu einer immer feineren und festeren Vorgabe der Seite. Insbesondere die Werbeindustrie möchte gerne schön und gleichmäßig gestaltete Seiten erzielen.

In den verschiedenen Version von HTML wurden immer mehr Möglichkeiten eingebaut, um genauere Formatierungen für Browser zu erreichen. Mit der Version 4.0 wurden dann aber auch Möglichkeiten geschaffen, diese Formatierungen wieder aus der HTML-Datei zu entfernen und extern zu verwalten. Aber das kommt noch. Für den Moment wollen wir mit unseren Beispielen der Entwicklung von HTML folgen.

Im Standard zu HTML 4.0 werden alle Parameter und Anweisungen von HTML als „deprecated / nicht mehr erwünscht“ bezeichnet, die Formatierungen exakt festlegen. Alle Formatierungen sollen in Zukunft getrennt von der eigentlichen HTML-Datei verwaltet werden. Sie werden dies im Kapitel über die Stilvorlagen (CSS-cascading style sheets) kennen lernen.

Folgen wir trotzdem hier der historischen Entwicklung. Schließlich existieren viele Millionen von Webseiten, die man nicht einfach über Bord werfen kann. Auch zukünftige Browser werden die älteren Webseiten anzeigen müssen.

5.6 HTML und Farben

Die `<font>`-Anweisung kann noch einen weiteren Parameter verwenden. Mit `color` kann man für einen Text eine Farbe vorgeben. Die Angabe der Farbe geschieht entweder mit einer Gruppe aus drei hexadezimalen Werten oder einem Klartextnamen angeben.

Die Klartextnamen sind englische Farbbezeichnung. Die Liste der gültigen Namen finden Sie im Anhang. Dort steht auch ein Beispiel zur Berechnung der hexadezimalen Farbangaben.

5.6.1 Farbvorgaben für die ganze Seite

Im Browser können wir in den Optionen einstellen, welche Farbe für den Hintergrund normalerweise verwendet wird. Und auch, welche Farbe für die Schrift genommen, wird, damit möglichst ein kontrastreiches Bild entsteht. Diese Angaben können wir auch in der HTML-Datei in der `<body>`-Anweisung der Datei mitgeben.

Die gleichen Farbvorgaben, wie in der `<font>`-Anweisung, können benutzt werden, um die gewünschten Farben anzugeben.

Sehen wir uns die einzelnen Parameter an. Die Hintergrundfarbe ist mit `bgcolor` einstellbar und die Farbe des normalen Textes mit `text`.

Einige weitere Farbvorgaben für `<body>` werden wir uns im Kapitel über Verweise betrachten.

L-EXP05: Regenbogentext

Schreiben Sie eine HTML-Datei, die eine Reihe von farbigen Texten wie bei einem Regenbogen ausgibt.

Der Hintergrund der Seite soll grünlich (greenyellow) sein. Die Standardschriftfarbe setzen wir auf schwarz (black).

Einige einfache Farbnamen sind: *red*, *green*, *greenyellow*, *blue*, *brown*, *black*, *white*, *cyan*, *gray*, *pink*. usw.

Abbildung 5.5: Beispiel für Farbdarstellungen

5.7 Schriftarten

Die Farben sollten auf jedem modernen Rechner darstellbar sein. Und damit sind sie in den allermeisten Fällen kein Problem. Bedeutend schwieriger wird es, wenn wir den Parameter der `<font>`-Anweisung nutzen, der uns ermöglicht, eine bestimmte Schrift zu verwenden.

Schriften sind traditionell geistiges Eigentum und damit geschützt. Außer wenigen allgemein bekannten Namen gibt es praktisch keine standardisierte Bezeichnung für Schriften. Und so finden Sie auf vielen Rechnern Schriftnamen, die den Erfindungsreichtum ihrer Namensgeber gut dokumentieren.

Man kann also kaum eine Schrift für die Darstellung eines Textes vorgeben, die allgemein gültig ist.

Der notgedrungen magere Ausweg aus dem Dilemma ist die Möglichkeit, eine Liste der gewünschten Schriften (jeweils durch ein Komma getrennt) vorzugeben. Und wenn doch kein Name exakt gefunden wird, dann schaltet der Browser eben auf seine eigene Darstellung um.

Zu den wenigen allgemeingültigen Schriftbegriffen gehören die Namen *serif*, *sans-serif* und *monospace*.

Im *HTML*-Standard wurden zusätzlich *cursive* und *fantasy* genormt.

Die Bezeichnungen beziehen sich auf die kleinen Häkchen (Verzeihung: es muss Serifen heißen), die insbesondere in den Schriften der Zeitungen gern benutzt werden. Es hat sich gezeigt, dass Schriften mit Serifen leichter zu lesen sind. Das menschliche Auge erkennt viel leichter Schriften, die deutliche graphische Unterschiede anbieten. Unsere Mischung aus Groß- und Kleinschreibung im Deutschen erleichtert beispielsweise sehr das Lesen. Inzwischen nutzen insbesondere amerikanische Programmierer gerne die bessere Lesbarkeit der gemischten Schreibweise und fügen in viele Bezeichnungen am Wortanfang Großbuchstaben ein, auch wenn dies nach der üblichen englischen Grammatik nicht richtig ist.

Der Begriff *monospace* ist das Gegenteil einer proportionalen Schrift. Eine Schrift, in der jeder Buchstabe die gleiche Breite besitzt, ist für Geräte wie mobile Telefone oder einfache Bildschirme ideal. Ein weiterer Ausdruck dafür, der an einer anderen Stelle verwendet wird, ist *teletype* für den Fernschreiber. Diese Geräte benutzten ebenfalls Zeichen, die immer den gleichen Raum (oder Breite) verwendeten.

Die Begriffe *cursive* (dt. kursiv) und *fantasy* (dt. Erfindung) lassen sich nicht direkt mit einem Schriftbild in Verbindung bringen. Die Auswahl bleibt einmal mehr dem Browser überlassen.

Der Parameter zur Angabe der Schriftarten ist *face*.

L-EXP06: Schriftarten auswählen

Schreiben Sie eine HTML-Datei, die einigen Teiltexten verschiedene Schriftarten zuweist. Nutzen Sie die Möglichkeit, lokale Schriftarten Ihres Rechners vorzugeben. Schließen Sie aber immer die Liste der jeweils gewünschten Schriftarten mit einer der fünf genannten Standardschriftarten ab.

Die Liste der vorhandenen Schriftarten zeigt Ihnen entweder ein Dienstprogramm des Betriebssystems oder die entsprechende Auswahlliste eines komfortablen Editors.

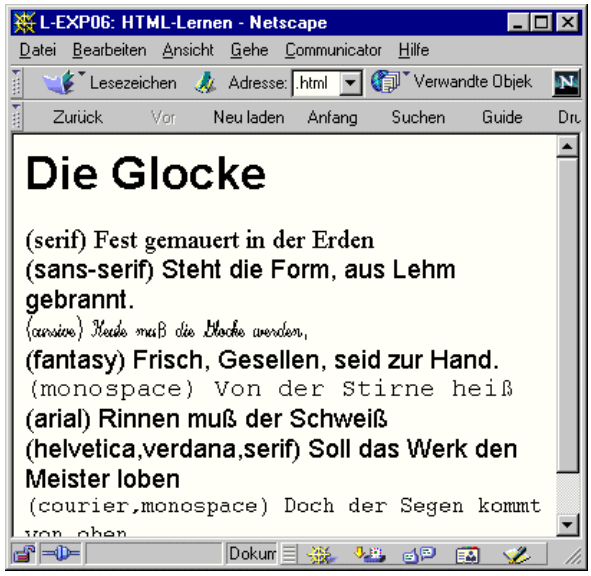

Abbildung 5.6: Verschiedene Schriftarten

5.8 HTML-Strukturen

Einen Begriff werden wir noch gelegentlich benötigen, den wir an Hand der Überschriften gut vorstellen können. Es ist der Block. Ein Block hat die Eigenschaft, dass er immer am Anfang der Zeile mit der Darstellung beginnt. Eine Überschrift benötigt keine explizite Zeilenschaltung mit `<br>`, um immer eine eigene, neue Zeile zu verwenden.

L-EXP07: Blöcke in HTML

Schreiben Sie eine HTML-Datei, die eine einfache, wenn auch nicht besonders logische Anordnung nutzt. Es sollen mehrere Überschriften nacheinander geschrieben werden ohne, dass Text dazwischen steht.

Jede Überschrift wird trotzdem eine eigene Zeile benutzen.

Meist stellen Browser Blöcke auch mit einem gewissen Abstand oberhalb und unterhalb dar. Im letzten Experiment kann es daher durchaus passieren, dass die Überschriften weit auseinander stehen.

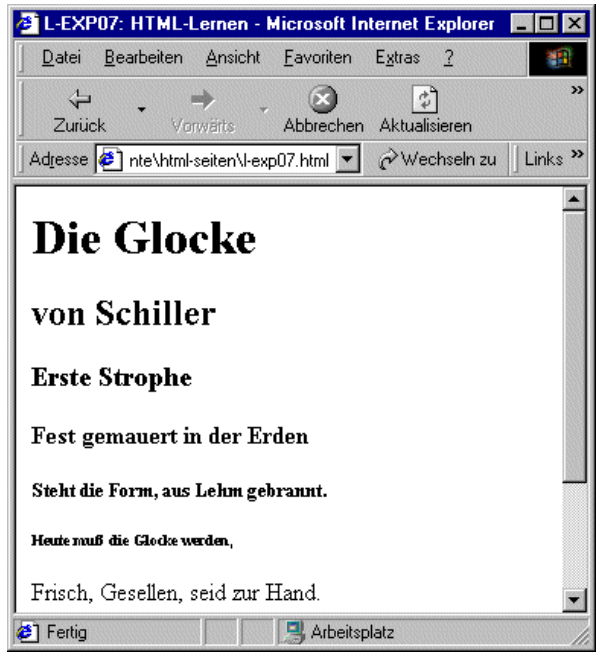

Abbildung 5.7: Überschriften sind Blöcke

Im nächsten Kapitel

Die logischen Auszeichnungen bieten eine Fülle von Möglichkeiten. Aber einige wenige explizite Anweisungen brauchen wir doch noch.

5.9 Lösungen

L-EXP01: Auszeichnung von Überschriften

```
<html>
<head>
<title>
L-EXP01: HTML-Lernen
</title>
</head>
<body>
Friedrich Schiller<br>
<br>
<h1>Die Glocke</h1>
Fest gemauert in der Erden<br>
Steht die Form, aus Lehm gebrannt.<br>
Heute muß die Glocke werden,<br>
Frisch, Gesellen, seid zur Hand.<br>
Von der Stirne heiß<br>
Rinnen muß der Schweiß,<br>
Soll das Werk den Meister loben,<br>
Doch der Segen kommt von oben.<br>
<br>
</body>
</html>
```

L-EXP02: Darstellung von Überschriften

```
<html>
<head>
<title>
L-EXP02: HTML-Lernen
</title>
</head>
<body>
<h1>Die Glocke</h1>
<h2>Fest gemauert in der Erden</h2>
<h3>Steht die Form, aus Lehm gebrannt.</h3>
<h4>Heute muß die Glocke werden,</h4>
<h5>Frisch, Gesellen, seid zur Hand.</h5>
<h6>Von der Stirne heiß</h6>
</body>
</html>
```

L-EXP03: Arbeiten mit logischen Formatierungen

```
<html>
<head>
<title>
L-EXP03: HTML-Lernen
</title>
</head>
<body>
<blockquote>"Streng dich an.
 Versuche soviel Ausbildung wie möglich zu bekommen,
 und dann,
 um Himmels Willen,
 tu etwas." (blockquote) <br>
- Lee Iacocca </blockquote>
<h1>Die Glocke</h1>
Fest gemauert in der
Erden<br>
<code>Steht die Form, aus Lehm gebrannt. (code)</code><br>
<address>Heute muß die Glocke werden, (address)</address><br>
<em>Frisch, Gesellen, seid zur Hand.(em)</em><br>
<strong>Von der Stirne heiß (strong)</strong><br>
<small>Rinnen muß der Schweiß (small)</small><br>
<big>Soll das Werk den Meister loben (big)</big><br>
Doch der <q>Segen</q> kommt von oben.<br>
</body>
</html>
```

L-EXP04: Setzen der Schriftgröße

```
<html>
<head>
<title>
L-EXP04: HTML-Lernen
</title>
</head>
<body>
<h1>Die Glocke</h1>
<font size="+4">Fest gemauert in der
Erden (+4)</font><br>
<font size="+3">Steht die Form, aus Lehm gebrannt. (+3)</font><br>
<font size="+2">Heute muß die Glocke werden, (+2)</font><br>
<font size="+1">Frisch, Gesellen, seid zur Hand.(+1)</font><br>
<font size=+0">Von der Stirne heiß (+0)</font><br>
<font size="-1">Rinnen muß der Schweiß (-1)</font><br>
<font size="-2">Soll das Werk den Meister loben (-2)</font><br>
<font size="-3">Doch der Segen kommt von oben(-3)</font>.<br>
```

```
Zum Werke, das wir ernst bereiten...<br>
</body>
</html>
```

L-EXP05: Regenbogentext

```
<html>
<head>
<title>
L-EXP05: HTML-Lernen
</title>
</head>
<body bgcolor="greenyellow" text="black">
<p>Texte in verschiedenen Farben (nicht ganz der Regenbogen)
</p>
<h1>Die Glocke</h1>
<font Fest gemauert in der
Erden<br>
<font  color="red">Steht die Form, aus Lehm gebrannt.</font><br>
<font  color="yellow"> Heute muß die Glocke werden, </font><br>
<font  color="green"> Frisch, Gesellen, seid zur Hand.</font><br>
<font  color="blue">Von der Stirne heiß </font><br>
<font <font color="white"> Rinnen muß der Schweiß </font><br>
<font color="maron""> Soll das Werk den Meister loben </font><br>
Doch der Segen kommt von oben.<br>
Zum Werke, das wir ernst bereiten...<br>
</body>
</html>
```

L-EXP06: Schriftarten auswählen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>L-EXP06: HTML-Lernen</title>
</head>
<body>
<h1>
Die Glocke</h1>
<font face="serif"><font size=+1>(serif) Fest gemauert in der Erden</
font></font>
<br><font face="sans-serif"><font size=+1>(sans-serif) Steht die Form,
aus Lehm gebrannt.</font></font>
<br><font face="cursive"><font size=+1>(cursive) Heute mu&szlig; die
Glocke
```

```
werden,</font></font>
<br><font face="fantasy"><font size=+1>(fantasy) Frisch, Gesellen, seid
zur Hand.</font></font>
<br><font face="monospace"><font size=+1>(monospace) Von der Stirne
hei&szlig;</font></font>
<br><font face="arial"><font size=+1>(arial) Rinnen mu&szlig; der
Schwei&szlig;</font></font>
<br><font face="helvetica,verdana,serif"><font size=+1>(helvetica,ver-
dana,serif)
Soll das Werk den Meister loben</font></font>
<br><font size=+1><font face="courier,monospace">(courier,monospace) Doch
der Segen kommt von oben</font>.</font>
<br><font size=+1>Zum Werke, das wir ernst bereiten...</font>
</body>
</html>
```

L-EXP07: Blöcke in HTML

```
<html>
<head>
<title>
L-EXP07: HTML-Lernen
</title>
</head>
<body>
<h1>Die Glocke</h1>
<h2>von Schiller</h2>
<h3>Erste Strophe</h3>
<h4> Fest gemauert in der Erden</h4>
<h5>Steht die Form, aus Lehm gebrannt.</h5>
<h6>Heute muß die Glocke werden, </h6>
Frisch, Gesellen, seid zur Hand.<br>
Von der Stirne heiß <br>
Rinnen muß der Schweiß <br>
Soll das Werk den Meister loben <br>
Doch der Segen kommt von oben.<br>
Zum Werke, das wir ernst bereiten...<br>
</body>
</html>
```

6 Explizite Formatierungen

Einige Formatierungen, die direkt die Darstellung steuern, waren schon von Beginn an dabei. Manchmal muss der Autor Eigenschaften genau vorgeben können. Ein Beispiel dafür sind Buchstaben, die man höher oder tiefer stellen möchte.

Gerade bei der Darstellung von Computerthemen will man bisweilen unterscheiden können, ob dies nun der erklärende Text oder eine Ein- oder Ausgabe ist. Diese Daten, die der Computer generiert, werden oft in einem Zeichensatz dargestellt, dessen Zeichen gleiche Breiten haben. Im Gegensatz dazu wird dieser Text proportional gesetzt.

Für solche und ähnliche Anwendungsfällen enthält das klassische HTML einige Anweisungen, um das Erscheinungsbild gezielt zu steuern.

Zur Übersicht hilft die folgende Tabelle:

Anweisung	Bedeutung
... (bold)	Halb fette Anzeige
... (italics)	Kursive Anzeige
<tt> ... </tt> (teletype)	Schrift mit fester Größe
<u> ... </u> (underline)	Unterstreichen
<s> ... </s> (strikethrough)	Durchstreichen
<strike> .. </strike> (strikethrough)	Durchstreichen (identisch zu <s>)
^{...} (superscript)	Hoch stellen
_{...} (subscript)	Tief stellen
<blink> ... </blink> (blink)	Blinken
<pre> ... </pre> (preformatted)	Vorformatiert; respektiert einfache Formatierungen

Tabelle 6.1: Explizite Formatierungen

Die Liste ist nicht allzu lang geworden. Bis auf die `<blink>`-Anweisung sind sie in vielen Fällen nützliche Steuerungen, die die Darstellung von einfachen chemischen Formeln (z.B. H_2O) oder auch von mathematischen Ausdrücken (z.B. $4^2 = 16$) erleichtern.

Für diese einfachen Formatierungen wollte man möglichst kurze Anweisungen benutzen. Daher hat man zusätzlich zu `<strike>` auch `<s>` eingeführt.

Aber sehen wir uns einmal ein praktische Anwendung im nächsten Experiment an.

Im Experiment sollen mehrere Anweisungen auf die gleiche Textstelle wirken. Dazu kann man nacheinander alle Attribute vorgeben, die der Text haben soll. Nach dem Text müssen die Anweisungen immer mit der Ende-Marke geschlossen werden. Dabei ist die Reihenfolge wichtig. Wie in der Mathematik müssen die zuletzt angegebenen Anweisungen zuerst wieder geschlossen werden.

Außerdem darf natürlich keine Ende-Marke fehlen.

P-EXP01: Gemischte explizite Formatierungen

Schreiben Sie eine HTML-Datei, in der die expliziten Formatierungen einzeln und gemischt benutzt werden. Stellen Sie die beiden Beispiele zu chemischen Formeln und mathematischen Ausdrücken in HTML dar.

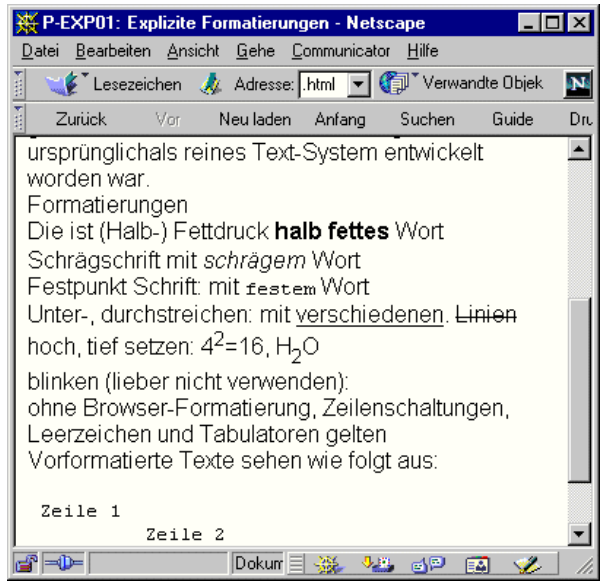

Abbildung 6.1: Explizite Formatierungen

Eine unglückliche Auszeichnung ist die <blink>-Anweisung. Ich weiß nicht, wie es Ihnen mit einer blinkenden Schrift ergeht. Eine Seite mit blinkenden Anzeigen macht mich ziemlich schnell nervös und ich verlasse diese Seite dann. Das gilt bei mir nicht nur für blinkende Texte sondern auch für die oft hektischen Werbeeinblendungen. Vielleicht sollten Sie besser diese Art der Auszeichnung vermeiden.

Eine besondere Rolle spielt die <pre>-Anweisung. Innerhalb dieser Anweisung wird die Formatierung durch den Browser abgeschaltet. Sowohl die normalen Zeilenschaltungen wie auch Leerzeichen bleiben erhalten. Allerdings benutzt der Browser meist eine feste Schriftart mit gleichen Breiten für alle Zeichen, um den Eindruck zu erwecken, es würde ein Computerausdruck angezeigt.

P-EXP02: Anzeigen vorformatierter Texte

Schreiben Sie eine HTML-Datei, in der eine Strophe eines Gedichtes vorkommt. Die Strophe soll innerhalb einer <pre>-Anweisung stehen. Experimentieren Sie mit Zeilenschaltung, Leerzeichen und Tabulatoren.

Sehen Sie die Unterschiede in der Darstellung? Wie werden Zeilenschaltungen, Leerzeichen und Tabulatoren nun behandelt?

Abbildung 6.2: Vorformatierter Text

Mit der Liste der bisher bekannten Formatierungen können wir bereits eine ganze Reihe von Informationsseiten schreiben und sie publizieren. Im wissenschaftlichen Bereich würden wir dabei vermutlich auch Leser finden. Noch aber sind die Möglichkeiten zur Gestaltung nicht ausreichend, um Texte für alle zu erstellen. Dazu brauchen wir noch weitere Gestaltungselemente.

Im nächsten Kapitel

Viele Menschen sind durch die Qualität der Schrifterzeugnisse verwöhnt. Ein gelernter Graphiker oder Buchdrucker ist wahrscheinlich oft entsetzt, wenn er die simplen Seiten von Gelegenheitspublizisten betrachtet. Um wirklich schöne Seiten zu erstellen, braucht es viel Stilgefühl und einige technische Kenntnisse.

Zu den technischen Kenntnissen kann Ihnen das Buch verhelfen – mit der Gestaltung muss ich Sie als Leser auf die vielen guten Beispiele im WWW verweisen. Hoffentlich können Sie dort herausfinden, welchen Stil Sie bevorzugen und dann mit den hier beschriebenen Mitteln auch in die Realität umsetzen.

Bis jetzt haben wir uns mit einzelnen Textstellen befasst. Nun wollen wir uns den größeren Abschnitten zuwenden und Texte in Absätze und Kapitel gliedern.

6.1 Lösungen

P-EXP01: Gemischte explizite Formatierungen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content= "text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>P-EXP01: Explizite Formatierungen </title>
</head>
<body>
<h1>
Explizite Formatierungen in HTML</h1>
Mit HTML beschreibt der Autor W&uuml;nsche zur Darstellung von Textpassa-
gen.Die
Liste der expliziten M&ouml;glichkeiten ist nicht sehr weit
gefa&szlig;t.Man
```

```
merkt hier noch ein wenig, dass HTML urspr&uuml;nglichals reines Text-
System
entwickelt worden war.
<br>Formatierungen
<br>Die ist (Halb-) Fettdruck <b>halb fettes</b> Wort
<br>Schr&auml;gschrift mit <i>schr&auml;gem</i> Wort
<br>Festpunkt Schrift: mit <tt>festem</tt> Wort
<br>Unter-, durchstreichen: mit <u>verschiedenen</u>. <strike>Linien</
strike>
<br>hoch, tief setzen: 4<sup>2</sup>=16, H<sub>2</sub>O
<br>blinken (lieber nicht verwenden): <blink>Blinken</blink>
<br>ohne Browser-Formatierung, Zeilenschaltungen, Leerzeichen und Tabula-
toren gelten
<br>Vorformatierte Texte sehen wie folgt aus:
<pre> Zeile 1
          Zeile 2</pre>
Und noch gemischt: <b><i><strike>gemischte Formatierung</strike></i></b>
und normal weiter.
</body>
</html>
```

P-EXP02: Anzeigen vorformatierter Texte

```
<html>
<head>
<title>
P-EXP02: HTML-Lernen
</title>
</head>
<body>
<h1>Die Glocke</h1>
<pre>
Fest gemauert in der Erden
Steht die Form, aus Lehm gebrannt.
Heute muß die Glocke werden,
Frisch, Gesellen, seid zur Hand.
Von der Stirne heiß
Rinnen muß der Schweiß
Soll das Werk den Meister loben
Doch der Segen kommt von oben
Zum Werke, das wir ernst bereiten..
</pre>
Nach der Strophe ...
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

7 Gestaltung von Absätzen, Kapiteln und Seiten

Neben den bisherigen Formatierungen, die auf einen kurzen Text wirken, erlaubt HTML auch die Strukturierung des gesamten Textes in logischen Einheiten wie z.B. Absätze. Einen Absatz fasst man als einen Darstellungsblock auf. Blöcke beginnen immer am Anfang einer eigenen Zeile. Wie Überschriften sind auch Absätze Blöcke in HTML.

Unsere nächste Anweisung beschreibt so einen Absatz-Block. Es ist die `<p>`-Anweisung.

Einen weiteren Block kennen Sie bereits vom allerersten Beispiel an. Der äußerste Block war `<body>`. Da wir innerhalb des `<body>`-Blocks wiederum andere Blöcke wie die gerade diskutierten Absätze angeben dürfen, sagt man auch, dass Blöcke verschachtelt werden dürfen. Der Absatz-Block ist dabei der kleinste Block, der keinen weiteren Blöcke enthalten kann.

Blöcke werden optisch meist dadurch dargestellt, dass der Browser ein wenig Abstand zwischen Blöcken lässt.

Blöcken werden wir später auch weitere Eigenschaften zuweisen. Dazu können Ausrichtung, Farben, Schriften etc. gehören.

In den Anfängen von HTML bestand die `<p>`-Anweisung immer nur aus der Anfangs-Marke. Erst mit HTML 4.0 wurde die Ende-Marke eingeführt. Fehlt sie, wird der Browser sie sozusagen selbstständig annehmen, wenn der nächste Blockbeginn geschrieben wird.

Allerdings wird in Zukunft die Ende-Marke der `<p>`-Anweisung verbindlich erwartet.

Unsere Diskussion über Blöcke erlaubt es nun, dass wir einen Parameter betrachten, der in einer `<p>`-Anweisung die Ausrichtung bestimmt. Ein Absatz ist ja in einem größeren Block (oder Container) eingebettet. Relativ zum umgebenden `<body>`-Block kann man festle-

gen, dass der Text eines Absatzes links- oder rechtbündig oder in der Mitte angeordnet wird.

Der Parameter ist *align* und kann die Werte *center*, *left* oder *right* annehmen. Es kann damit nur genau einer aus der Liste der Werte benutzt werden.

A-EXP01: Absatzformatierung

Schreiben Sie ein HTML-Programm, dass drei Absätze mit Texten enthält. Die Absätze sollen unterschiedlich ausgerichtet werden, um die drei Möglichkeiten zu zeigen.

Sie können dazu gerne drei Strophen des Gedichtes verwenden und einzelne Strophen als Absätze auszeichnen.

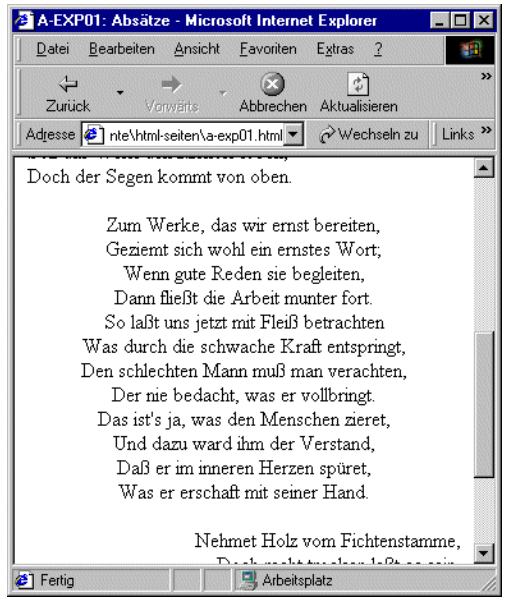

Abbildung 7.1: Absätze und Ausrichtung

Innerhalb eines `<p>`-Blockes gilt die Formatierung des Browsers. Er wird überflüssige Leerzeichen entfernen. Aufeinander folgende Leerzeichen wird er intern als eines anzeigen. Entfernt werden außerdem Zeilenschaltungen und Tabulatoren. Der so erhaltene Text wird dann gemäß der vorhandenen Größe des Browser-Fensters dargestellt.

In einem zweiten Schritt können beliebige Block-Anweisungen in einem größeren Rahmen zusammengefasst werden. Die passende Anweisung heißt `<div>` .. `</div>` für engl. division / Abschnitt.

Solche größeren Abschnitte werden besonders dann wichtig, wenn wir uns in einem späteren Kapitel um weitere Eigenschaften kümmern werden, die für die Formatierung wichtig sind.

Für den Moment wollen wir besser wieder ein Experiment machen, um die grundlegenden Möglichkeiten der verschachtelten Blöcke zu erkunden.

A-EXP02: Bilden von Abschnitten

Nehmen Sie das vorhergegangene Beispiel und entfernen Sie aus den `<p>`-Anweisungen die *align*-Parameter. Nur ein Absatz soll die Ausrichtung nach rechts behalten.

Setzen Sie vor die Überschrift und hinter den dritten Absatz, die Start- und Ende-Marke der `<div>`-Anweisung.

Fügen Sie nun nur in der `<div>`-Anweisung den Parameter zur Mitte-Ausrichtung wieder ein. Unsere Blöcke sollen nun alle in der Mitte stehen. Da auch die Überschrift ein Block ist, wird auch sie mit ausgerichtet.

Nur ein Absatz hat noch eine eigene Ausrichtung, Welche der sich widersprechenden Ausrichtungsparameter gewinnt nun?

Abbildung 7.2: Absatz mit eigener Ausrichtung

Die Firma Netscape hatte in einer frühen Browser-Version auf Wunsch vieler Anwender die `<center>` .. `</center>`-Anweisung eingeführt, die alle nachfolgenden Elemente wie Absätze und Bilder in die Mitte rückt.

Diese Anweisung wurde später auch in den HTML-Standard übernommen. Allerdings fand sich immer der Hinweis, dass die Anweisung nicht allgemein genug ist, um den Ideen von HTML gerecht zu werden. Verwenden Sie daher nach Möglichkeit lieber die `<div>`-Anweisung.

7.1 Graphische Gliederung eines Textes

Die Unterteilung eines Textes geschah bisher nur durch textliche Stilmittel, wie ein wenig Abstand zwischen Absätzen. Zur graphischen und damit leicht erkennbaren Gliederung gibt es in HTML noch ein einfaches Trennelement. Mit der `<hr>`-Anweisung kann man einen graphischen Trennstrich einfügen. Ebenso wie bei `<br>` gibt es keine Ende-Marke. `<hr>` steht dabei für „horizontal rule“ / horizontale Trennlinie.

Die `<hr>`-Anweisung erlaubt folgende Parameter: *align*, *noshade*, *size* und *width*.

Mit *align* kann man wieder die Ausrichtung steuern. Allerdings ergibt dies nur in Verbindung mit dem *width*-Parameter Sinn, der eine Breite kleiner 100% angibt.

Mit *size* kann man die Größe oder genauer gesagt, die Dicke angeben. Und schließlich wird *noshade* (kein Schatten) den 3-D-Effekt des Browsers abschalten. Die Linie sollte dann flach erscheinen.

Wichtig für Größenangaben ist die Möglichkeit, relative Angaben zu machen. Unsere Blöcke wirken ja wie Container. Und die können eine eigene Breite besitzen, auch wenn wir dies noch nicht verwendet haben. Die Größe der Linie kann in Prozent der Größe des umgebenden Containers angegeben werden. Wie sinnvoll dies ist, sieht man, wenn das Browser-Fenster und damit der umgebende Container (Absatz oder *body*-Bereich) schmäler wird. Den Wert des Parameters gibt man dann z.B. mit „60%“ an.

Eine weitere Möglichkeit sind absolute Größenangaben in *Pixel*. Die haben dann aber immer die Schwierigkeit sich der jeweiligen Darstellung anzupassen. Eine 10 Pixel dicke Linie wirkt auf einem Monitor mit geringer Auflösung viel kräftiger, als auf einem Monitor mit hoher Auflösung.

A-EXP03: Graphische Untergliederung

Fügen Sie in die vorhergegangene Datei graphische Untergliederungen ein.

Unter der Überschrift soll eine einfache graphische Linie ohne Parameter stehen.

Zwischen den Absätzen fügen Sie eine Linie ein, die halb so breit wie die Seite ist und in der Mitte steht.

Und den Schluss bildet eine flache Linie, die 6 Pixel dick ist.

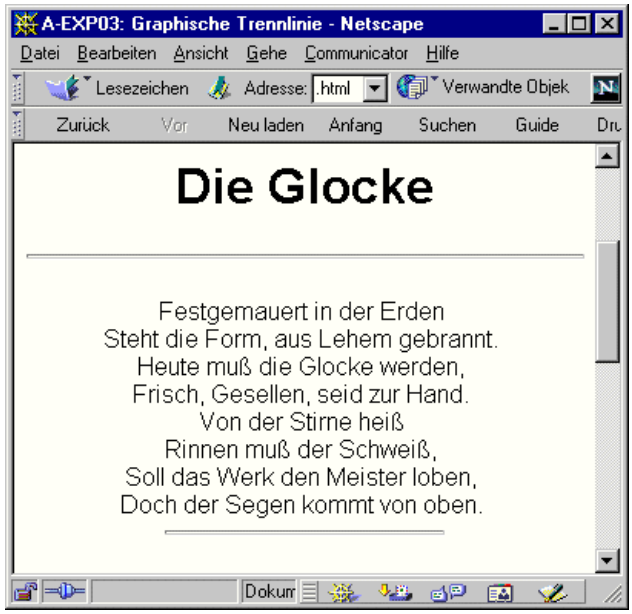

Abbildung 7.3: Erste graphische Elemente

Im nächsten Kapitel

Unsere Seiten sind nun schon deutlich gegliedert und wirken etwas freundlicher als Seiten mit fortlaufendem Text. Wir bleiben noch ein wenig bei der Formatierung von Texten, bevor wir uns dann bald den Bildern zuwenden.

Aber im nächsten Kapitel kümmern wir uns noch einmal um eine spezielle Form der Textgliederung, die Listen.

7.2 Lösungen

A-EXP01: Absatzformatierung

```
<html>
<head>
<title>A-EXP01: Absätze</title>
</head>
<body >
<h1>Absätze in HTML</h1>
<p>
Blöcke geben unserem Text die notwendige Struktur. Als Blöcke kennen wir
bisher Überschriften und Absätze. </p>
<h1>Die Glocke</h1>
<p align="left">
Festgemauert in der Erden<br>
Steht die Form, aus Lehem gebrannt.<br>
Heute muß die Glocke werden,<br>
Frisch, Gesellen, seid zur Hand.<br>
Von der Stirne heiß<br>
Rinnen muß der Schweiß,<br>
Soll das Werk den Meister loben,<br>
Doch der Segen kommt von oben.</p>
<p align="center">
Zum Werke, das wir ernst bereiten,<br>
Geziemt sich wohl ein ernstes Wort;<br>
Wenn gute Reden sie begleiten,<br>
Dann fließt die Arbeit munter fort.<br>
So laßt uns jetzt mit Fleiß betrachten<br>
Was durch die schwache Kraft entspringt,<br>
Den schlechten Mann muß man verachten,<br>
Der nie bedacht, was er vollbringt.<br>
Das ist's ja, was den Menschen zieret, <br>
Und dazu ward ihm der Verstand,<br>
Daß er im inneren Herzen spüret,<br>
Was er erschaft mit seiner Hand. </p>
<p align="right">
Nehmet Holz vom Fichtenstamme,<br>
Doch recht trocken laßt es sein,<br>
Daß die eingepreßte Flamme<br>
Schlage zu dem Schwalch hinein.<br>
Kocht des Kupfers Brei,<br>
Schnell das Zinn herbei,<br>
Daß die zähe Glockenspeise<br>
Fließe nach der rechten Weise.</p>
</body>
</html>
```

A-EXP02: Bilden von Abschnitten

```
<html>
<head>
<title>A-EXP02: Absätze</title>
</head>
<body >
<h1>Absätze in HTML</h1>
<p>
Blöcke geben unserem Text die notwendige Struktur. Als Blöcke kennen wir
bisher Überschriften und Absätze. </p>
<div align="center">
<h1>Die Glocke</h1>
<p>
Festgemauert in der Erden<br>
Steht die Form, aus Lehem gebrannt.<br>
Heute muß die Glocke werden,<br>
Frisch, Gesellen, seid zur Hand.<br>
Von der Stirne heiß<br>
Rinnen muß der Schweiß,<br>
Soll das Werk den Meister loben,<br>
Doch der Segen kommt von oben.</p>
<p align="left">
Zum Werke, das wir ernst bereiten,<br>
Geziemt sich wohl ein ernstes Wort;<br>
Wenn gute Reden sie begleiten,<br>
Dann fließt die Arbeit munter fort.<br>
So laßt uns jetzt mit Fleiß betrachten<br>
Was durch die schwache Kraft entspringt,<br>
Den schlechten Mann muß man verachten,<br>
Der nie bedacht, was er vollbringt.<br>
Das ist's ja, was den Menschen zieret, <br>
Und dazu ward ihm der Verstand,<br>
Daß er im inneren Herzen spüret,<br>
Was er erschaft mit seiner Hand. </p>
<p>
Nehmet Holz vom Fichtenstamme,<br>
Doch recht trocken laßt es sein,<br>
Daß die eingepreßte Flamme<br>
Schlage zu dem Schwalch hinein.<br>
Kocht des Kupfers Brei,<br>
Schnell das Zinn herbei,<br>
Daß die zähe Glockenspeise<br>
Fließe nach der rechten Weise.</p>
</div>
</body>
</html>
```

A-EXP03: Graphische Untergliederung

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content= "text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>A-EXP03: Graphische Trennlinie</title>
</head>
<body>
<h1>
Graphische Trennlinien in HTML</h1>
Bl&ouml;cke geben unserem Text die notwendige Struktur. Als Bl&ouml;cke
kennen wir bisher &Uuml;berschriften und Abs&auml;tze.
<center>
<h1>
Die Glocke</h1></center>
<div align="center">
<hr>
<p>Festgemauert in der Erden
<br>Steht die Form, aus Lehem gebrannt.
<br>Heute mu&szlig; die Glocke werden,
<br>Frisch, Gesellen, seid zur Hand.
<br>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;,
<br>Soll das Werk den Meister loben,
<br>Doch der Segen kommt von oben.
<br>
<hr width="50%" align="center">
<p>Zum Werke, das wir ernst bereiten,
<br>Geziemt sich wohl ein ernstes Wort;
<br>Wenn gute Reden sie begleiten,
<br>Dann flie&szlig;t die Arbeit munter fort.
<br>So la&szlig;t uns jetzt mit Flei&szlig; betrachten
<br>Was durch die schwache Kraft entspringt,
<br>Den schlechten Mann mu&szlig; man verachten,
<br>Der nie bedacht, was er vollbringt.
<br>Das ist's ja, was den Menschen zieret,
<br>Und dazu ward ihm der Verstand,
<br>Da&szlig; er im inneren Herzen sp&uuml;ret,
<br>Was er erschaft mit seiner Hand.
<br>
<hr width="50%" align="center">
<p>Nehmet Holz vom Fichtenstamme,
<br>Doch recht trocken la&szlig;t es sein,
<br>Da&szlig; die eingepre&szlig;te Flamme
```

```
<br>Schlage zu dem Schwalch hinein.
<br>Kocht des Kupfers Brei,
<br>Schnell das Zinn herbei,
<br>Da&szlig; die z&auml;he Glockenspeise
<br>Flie&szlig;e nach der rechten Weise.</div>
<hr noshade size="6">
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

8 Textlisten

Nach den Formatierungen einzelner Worte, Textstellen und Absätze kommen wir nun zu weiteren Möglichkeiten der textlichen Gestaltung – den Textlisten. Manchmal sind es dabei richtige Aufzählungen („die 10 besten Wasserball-Sportler sind...“) oder man gibt nur eine allgemeine Liste der Zutaten eines Koch-Rezeptes.

Und am Schluss eines Buches oder einer Gruppe von Webseiten hilft oft ein Glossar, Begriffe zu finden und kurz nachzuschlagen, was denn ein neuer Begriff bedeutet. Das ist meist schneller, als den Begriff über den Index und die passende Seite im Buch zu finden.

Textlisten lassen sich mit den `<ul>`- und `<ol>`-Anweisungen anlegen. Die englischen Begriffe dazu sind „unordered list“ / ungeordnete Liste und „ordered list“ / geordnete Liste.

Eine Liste benötigt Einträge. Innerhalb der Anweisungen für Listen tauchen daher die Einträge mit der `<li>` .. `</li>`-Anweisung auf. In den ersten Versionen war die Ende-Marke der `<li>`-Anweisung nicht Pflicht. Heute sollte sie benutzt werden und in Zukunft muss sie benutzt werden. Die Formulierung mit „war“, „sollte“ und „muss“ spiegelt einfach die aktuelle Situation wieder. Standards und real existierende Browser decken sich nicht immer und ältere Seiten sollten auch nach Möglichkeit heute noch korrekt dargestellt werden, auch wenn sie in einer veralteten HTML-Version geschrieben wurden.

Sehen wir uns die ungeordnete Liste in einem Experiment an.

T-EXP01: Listen in HTML

Schreiben Sie eine HTML-Datei mit etwas Text und einer Liste.

Die Liste soll ein paar Möglichkeiten darstellen. Das können die Zutaten eines Rezeptes, die Liste der Reiseutensilien, die Aufgabenliste für den Tag etc. sein.

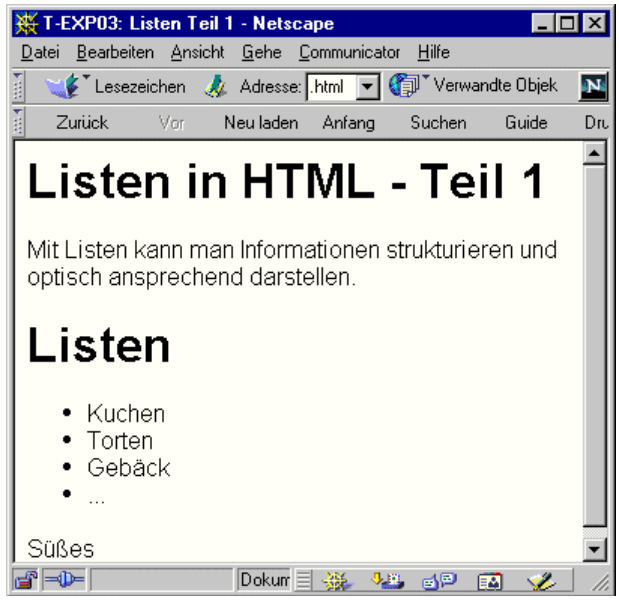

Abbildung 8.1: Einfache, ungeordnete Liste

Die ungeordnete Liste stellt jeden Eintrag mit einem vorangestellten Symbol dar. Es ist für alle Einträge das gleiche.

In der zweiten Variante wird die Liste selbst aktiv. Der Browser wird die Einträge der Liste mitzählen und die laufende Zahl dem jeweiligen Eintrag voranstellen.

T-EXP02: Listen in HTML

Schreiben Sie eine HTML-Datei mit etwas Text und einer Liste.

Die Liste soll eine Reihenfolge darstellen: Ihre 10 geschätztesten Sportler, die Rangfolge der französischen Städten nach Einwohnerzahl, die Verteilung der Bevölkerung nach Einkommensgruppen oder etwas Ähnliches.

Listen lassen sich auch verschachteln. An Stelle einer `<li>`-Anweisung kann wiederum eine Listen-Anweisung stehen.

Abbildung 8.2: Einfache, geordnete Liste

T-EXP03: Verschachtelte Listen

Ändern Sie das vorausgegangene Beispiel ab.

Fügen Sie dabei innerhalb einer `<li>`-Anweisung wiederum eine vollständige Liste ein. In der geordneten (durchnummerierten) Aufzählung sollten Sie eine unnummerierte Liste einfügen und umgekehrt in die unnummerierte Liste eine nummerierte Aufzählung.

Natürlich gibt es noch Parameter für die Listen-Anweisungen, mit denen man das Aussehen steuern kann. Für Aufzählungen kann man den *start*-Parameter angeben, der den numerischen Anfang anstelle der bisher automatisch verwendeten „1“ angibt. Der Wert ist eine positive ganze Zahl.

Die Form der Aufzählung lässt sich mit *type* (Typ) einstellen. Der Wert kann dann bei *a*, *A*, *I*, *i* oder *1* sein. Wenn wir keinen *type* angeben, wird die *1* gewählt. Die Bedeutung können wir am schnellsten wieder mit einem kleinen Experiment herausfinden.

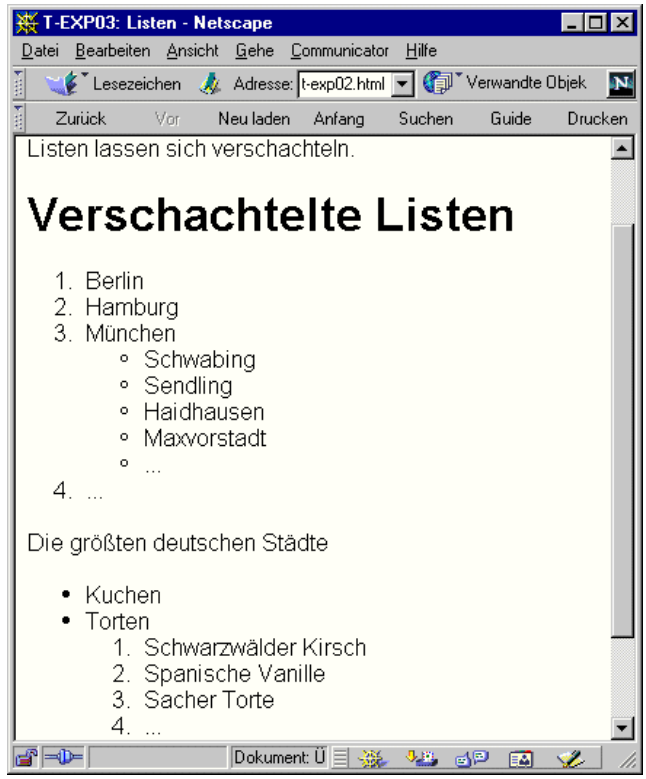

Abbildung 8.3: Verschachtelte Listen

T-EXP04: Typangaben bei Listen

Fügen Sie dazu die erste Liste mit der nummerierten Aufzählung fünfmal in den Quelltext ein. Am schnellsten geht das, wenn Sie den Text markieren, kopieren und mehrfach wieder einfügen.

Verwenden Sie unterschiedliche Startwerte und probieren Sie alle fünf Typangaben einmal aus.

Finden Sie die Darstellungen der Aufzählung nicht interessant? Die normale numerische Darstellung ist ja einfach. Bei der Aufzählung mit Buchstaben taucht sicher die Frage auf, welche Buchstaben genommen werden. Kommt bei Ihnen ein Umlaut vor? Und wenn an welcher Stelle im Alphabet?

Besonders eindrucksvoll sind die kleinen oder großen römischen Ziffern. Vielleicht verwenden Sie einmal ein sehr hohen Startwert, um die römische Zahlen so richtig genießen zu können.

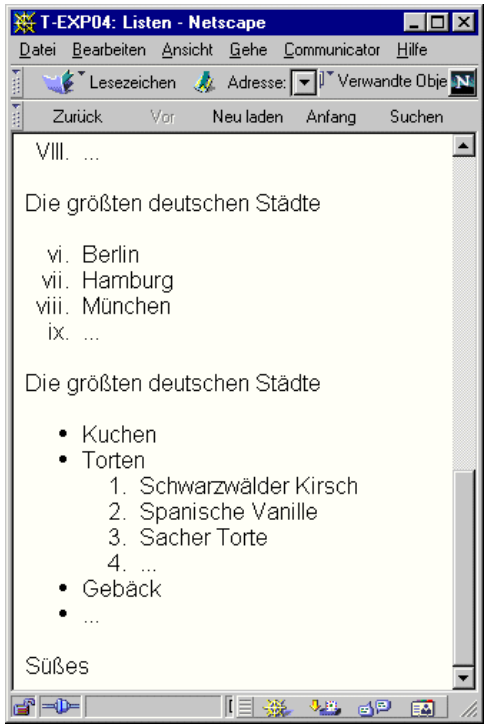

Abbildung 8.4: Varianten der Nummerierung

Leider schaffen die heutigen Browser nicht immer eine korrekte Darstellung großer Zahlen. Manchmal reserviert der Browser nur ein wenig Platz für die Aufzählungswerte und startet die Ausgabe des zugehörigen Textes an einer festen Stelle. Damit kann es zu Überdeckungen von Aufzählungszeichen und dem Text kommen. Vermeiden Sie also besser große Zahlenwerte.

Jetzt sollten wir noch die Steuerungsmöglichkeiten für unsortierte Aufzählungen besprechen. Hier können wir auch den *type*-Parameter verwenden, allerdings mit anderen Werten.

Für unsortierte Aufzählungen können wir für *type* als Parameterwerte *circle*, *disk* oder *square* verwenden. Übersetzt ist das ein Kreis, eine Scheibe oder ein Rechteck.

T-EXP05: Unsortierte Aufzählungen

Ändern Sie eines der vorhandenen Listen-Beispiele ab. Es sollten nun drei unsortierte Listen mit unterschiedlichen (Typ)-Angaben im HTML-Quelltext vorkommen.

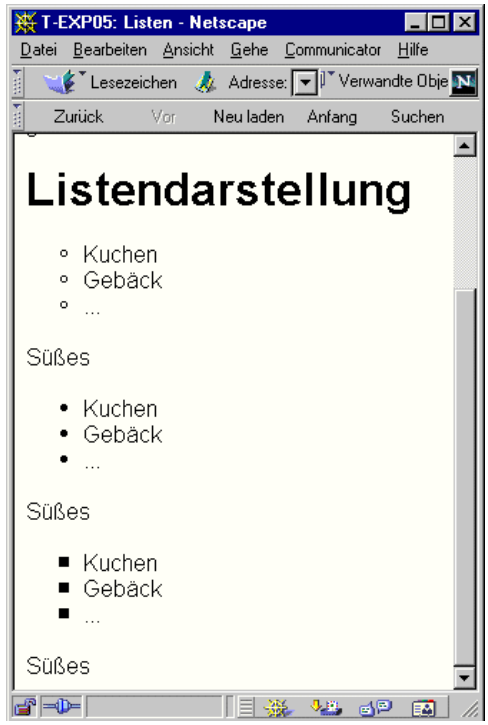

Abbildung 8.5: Varianten der Listenmarkierung

Und schließlich schauen wir uns noch einen Sonderfall an. Was passiert, wenn man gleichartige Listen ineinander verschachtelt? Hier besteht dann ein Aufzählungspunkt erneut aus einer Liste.

T-EXP06: Verschachtelte, gleichartige Listen

Ändern Sie das vorausgegangene Beispiel noch einmal ab. Es soll nun jeweils eine nummerierte Liste in einer ebenfalls nummerierten Liste und genauso eine unnummerierte Liste in einer unnummerierten Liste vorkommen. Denken Sie daran, dass die verschachtelte Liste innerhalb einer `<li>`-Anweisung der umgebenden Liste stehen muss.

Geben Sie dabei keinen *type*-Parameter an.

Wie verschachtelte Listen dargestellt werden, bleibt dem Hersteller des Browsers überlassen. Oft bleibt der Browser bei Aufzählung bei der gleichen Art. Bei unsortierten Listen wird meist das Symbol geändert.

Abbildung 8.6: Gleichartige, verschachtelte Listen

Und noch etwas: es ist auch üblich, dass verschachtelte Listen durch einen Einzug dargestellt werden, so als stünde jeweils ein Tabulatorzeichen vor jeder Zeile der verschachtelten Liste. Natürlich ist das kein wirkliches Tabulatorzeichen, denn das würde ja automatisch entfernt.

In der Diskussion bleibt uns noch das Glossar, das etwas komplexer aufgebaut ist, wie die bisherigen Listen. Ein Glossar besteht ähnlich wie die bisherigen Listen aus einer Listenhülle und den darin enthaltenen Element. Anders ist, dass nun ein Listenelement wiederum aus zwei Teilen besteht: dem Begriff und der Erklärung.

Ein Glossar besteht damit aus den drei HTML-Anweisungen `<dl> .. </dl>` als Hülle, `<dt> .. </dt>` für den Begriff (engl. term) und `<dd> .. </dd>` für die Erklärung (engl. definition). Natürlich wechseln sich in der Liste Begriffe und Erklärungen ab. Ein einzelner Eintrag besteht hier aus zwei Anweisungen.

T-EXP07: Glossar-Liste

Ergänzen Sie ein bestehendes Listen-Beispiel um ein kurzes Glossar.

Abbildung 8.7: Glossar-Liste

Mit den Listen haben wir die grundlegenden Auszeichnungen für Texte und Abschnitte geschafft.

Im nächsten Kapitel

Langsam haben wir die vielen Möglichkeiten der Gestaltung mit Texten ausgeschöpft. Es bleibt noch der Bereich der speziellen Zeichen übrig. Wir brauchen noch eine Möglichkeit, Zeichen darzustellen, die eigentlich eine besondere Funktion haben.

Wenn der Browser bei einer spitzen Klammer immer an einen Befehl denkt und damit die spitzen Klammern überliest, wie können wir dann im Text eine spitze Klammer darstellen?

Das ist nur eines der Probleme, die wir noch nicht erkundet haben.

Aber es wird das letzte Kapitel über Texte sein. Danach gehen wir zu den Bildern über – versprochen.

8.1 Lösungen

T-EXP01: Listen in HTML

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP01: Listen Teil 1</title>
</head>
<body>
<h1>
Listen in HTML - Teil 1</h1>
Mit Listen kann man Informationen strukturieren und optisch ansprechend
darstellen.
<h1>
Listen</h1>
<ul>
<li>
Kuchen</li>
<li>
Torten</li>
<li>
Beb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;e Sachen
</body>
</html>
```

T-EXP02: Listen in HTML

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP02: Listen Teil 2</title>
</head>
<body>
<h1>
Listen in HTML - Teil 1</h1>
Mit Listen kann man Informationen strukturieren und optisch ansprechend
darstellen.
```

```
<h1>
Listen</h1>
<ol>
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
</body>
</html>
```

T-EXP03: Verschachtelte Listen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP03: Listen</title>
</head>
<body>
<h1>
Listen in HTML - Teil 3</h1>
Listen lassen sich verschachteln.
<h1>
Verschachtelte Listen</h1>
<ol>
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<ul>
<li>
Schwabing</li>
<li>
Sendling</li>
<li>
Haidhausen</li>
<li>
```

```
Maxvorstadt</li>
<li>
...</li>
</ul>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
<ul>
<li>
Kuchen</li>
<li>
Torten</li>
<ol>
<li>
Schwarzw&auml;lder Kirsch</li>
<li>
Spanische Vanille</li>
<li>
Sacher Torte</li>
<li>
...</li>
</ol>
<li>
Geb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;es
</body>
</html>
```

T-EXP04: Typangaben bei Listen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content= "text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP04: Listen</title>
</head>
<body>
<h1>
Listen in HTML - Teil 4</h1>
Listen lassen sich gestalten.
<h1>
```

```
Listendarstellung</h1>
<ol start="2" type="1">
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
<ol start="3" type="A">
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
<ol start="4" type="a">
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
<ol start="5" type="I">
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
<ol start="6" type="i">
```

```
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<li>
...</li>
</ol>
Die gr&ouml;&szlig;ten deutschen St&auml;dte
<ul>
<li>
Kuchen</li>
<li>
Torten</li>
<ol>
<li>
Schwarzw&auml;lder Kirsch</li>
<li>
Spanische Vanille</li>
<li>
Sacher Torte</li>
<li>
...</li>
</ol>
<li>
Geb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;es
</body>
</html>
```

T-EXP05: Unsortierte Aufzählungen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP05: Listen</title>
</head>
<body>
<h1>
Listen in HTML - Teil 5</h1>
```

```
Auch unsortierte Listen lassen sich gestalten.
<h1>
Listendarstellung</h1>
<ul type="circle">
<li>
Kuchen</li>
<li>
Geb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;es
<ul type="disk">
<li>
Kuchen</li>
<li>
Geb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;es
<ul type="square">
<li>
Kuchen</li>
<li>
Geb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;es
</body>
</html>
```

T-EXP06: Verschachtelte, gleichartige Listen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP06: Listen</title>
</head>
<body>
<h1>
Listen in HTML - Teil 6</h1>
Verschachtelte Listen gleichen Typs.
```

```
<h1>
Listendarstellung</h1>
<ol>
<li>
Berlin</li>
<li>
Hamburg</li>
<li>
M&uuml;nchen</li>
<ol>
<li>
Schwabing</li>
<li>
Sendling</li>
<li>
Haidhausen</li>
<li>
Maxvorstadt</li>
<li>
...</li>
</ol>
<li>
...</li>
</ol>
<ul >
<li>
Kuchen</li>
<ul >
<li>
Mensch</li>
<li>
Tier</li>
<li>
Stein</li>
</ul>
<li>
Geb&auml;ck</li>
<li>
...</li>
</ul>
S&uuml;&szlig;es
</body>
</html>
```

T-EXP07: Glossar-Liste

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>T-EXP07: Listen</title>
</head>
<body>
<h1>
Listen in HTML - Teil 7</h1>
Verschachtelte Listen gleichen Typs.
<h2>
Das Glossar</h2>
<dl>
<dt>
Anweisung</dt>
<dd>
Befehl der Sprache HTML. Besteht meist aus einer Start- und einer Endemarke
(engl. tag) und dem dazwischen liegenden Inhalt</dd>
<dt>
Hypertext</dt>
<dd>
Mischung aus Text und Steuerelementen insbesondere f&uuml;r Verweise, mit
denen man zu einer anderen Stelle springen kann. Besonders bei Hilfesyste-
men
und im WWW verwendet</dd>
<dt>
HTML</dt>
<dd>
Hypertext markup language / Hypertext Auszeichnungssprache</dd>
<dt>
Sprachelement</dt>
<dd>
siehe auch Anweisung</dd>
</dl>
</body>
</html>
```

9 Sonderzeichen und spezielle Zeichensätze in HTML

Erinnern Sie sich noch an unsere Diskussion über Zeichensätze und Umlaute?

Wir wollen in diesem Kapitel noch einmal näher auf Zeichensätze und spezielle Zeichen eingehen. In der ursprünglichen Spezifikation hatte man zwei Bereiche unterschieden: innerhalb der Anweisungs-Marken und außerhalb davon. Für die zwei Bereiche wurden Zeichensätze bestimmt.

9.1 Zeichensätze für HTML

Anweisungen bestehen nur aus Zeichen des Basisteils vom verwendeten Zeichensatz *Latin 1* mit numerischen Werten zwischen 0 und 127. Und dieser Teil entspricht dem amerikanischen Zeichensatz *ASCII*. Alle Anweisungen werden heute klein geschrieben und verwenden nur Buchstaben aus dem ASCII-Zeichensatz. Und alle Texte zwischen den Marken, der eigentliche Inhalt, verwendete immer den Zeichensatz *Latin 1* mit den Zeichen für Umlaute und Akzente.

Mit dieser ursprünglichen Regelung gibt es aber immer noch ein paar Probleme. Das grundlegende Problem ist die Festlegung auf einen lokalen Zeichensatz. Auch wenn das westliche Europa, Amerika und Australien gut damit leben können, bleiben doch weite Regionen der Welt mit ihrer Sprache ausgesperrt.

9.1.1 Nicht-westliche Schriften

Viele Millionen Menschen lesen und schreiben ganz andere Schriftzeichen: arabische, chinesische oder japanische. Und das tun sie manchmal auch in einer anderen Leserichtung als unserer „von links nach rechts“.

Für alle Menschen im WWW soll es aber möglich sein, Seiten in der eigenen Sprache zu erstellen und zu lesen. Der Zeichensatz muss also entweder absolut universell oder das jeweilige Schriftsystem einstellbar sein.

Die Einstellung des Zeichensatzes hat mehrere Komponenten. Zuerst muss das Betriebssystem den Zielzeichensatz bereitstellen. Lebt man also in Europa und möchte Seiten aus Taiwan oder Japan lesen, muss man sich den Zeichensatz besorgen und installieren. Es gibt dazu kostenlose Angebote im WWW.

9.1.2 Einstellen des Zeichensatzes

Danach kann der Browser u.U. die Seiten anzeigen. Aber irgendwer muss dem Browser mitteilen, dass er nun eine Seite mit einem ganz anderen Zeichensatz anzeigen soll.

Nachdem der Browser die anzuzeigenden Seiten über das Internet von einem Server holt, also von einem Computer, der einen Teil seiner Festplatten mit Hilfe eines WWW-Server-Programmes der ganzen Internet-Gemeinde zur Verfügung stellt, könnte dies durch den Webserver geschehen.

Wir sollten an dieser Stelle aus praktischen Gründen noch den Begriff *Server* näher definieren. Er kann ja mehrdeutig sein. Einen ganzen Computer mit Hardware und Software zusammen, der Dienste im Netz anbietet, wollen wir einfach als *Server* bezeichnen.

Wird ein spezieller Dienst beschrieben, dann wird das durch eine nähere Bezeichnung des Server-Programms geschehen. Als Beispiel werden wir von einem *HTTP*-Server sprechen, wenn wir gezielt das Programm meinen, das mit unserem Browser kommuniziert und Dateien zur Verfügung stellt. HTTP steht dabei für *das Hypertext Transfer Protocol*. Oder wir sprechen statt von einem HTTP-Server von einem Webserver.

Nehmen wir einmal den Standardfall an. Eine für das westliche Europa normale HTML-Datei wird auf einem Server gespeichert. Wenn nun Sie an Ihrem Rechner, der als Dienstbenutzer oder engl. als Client bezeichnet wird, mit dem Browser diese Datei anfordern, dann regelt ein Satz von Spielregeln, den man als Protokoll bezeichnet, wie diese Datei gefunden und übertragen wird. Der Server kann dabei in einem Vorspann dem *Clienten* mitteilen, welche Art von Datei zu erwarten ist. Außerdem kann auch der gewünschte Zeichensatz ermittelt und dem Browser mitgeteilt werden. Der Browser nutzt den passenden Zeichensatz des Betriebssystems und zeigt die Datei an.

Sollte der gewünschte Zeichensatz nicht zur Verfügung stehen, dann wird die Datei zwar auch übertragen, aber Sie werden nur kleine Kästchen am Bildschirm sehen, da eben keine passende Darstellung in Form eines Zeichensatzes des Betriebssystems zur Verfügung steht.

9.1.3 Auf dem Weg zum UniCode

In Zukunft wird unser Leben mit Zeichensätzen einfacher werden. Seit der Version HTML 4.0 fordert der Standard, dass auch der globale Zeichensatz UniCode unterstützt wird. Hier sind die einzelnen Zeichen im Speicher des Computers doppelt so groß wie die *Latin 1*-Zeichen, nämlich 16 Bit. Darin kam man immerhin mehr als 65000 verschiedene Zeichen darstellen. Und wenn dies nicht reicht, könnte man die Breite noch einmal auf 32 Bit verdoppeln.

In diesem Zeichensatz sind alle Zeichen der Welt darstellbar und außerdem können Informationen über die Lese- und Schreibrichtung mitgegeben werden. Es sollte also in Zukunft kein Problem mehr sein, in einen deutschen Text ein arabisches Zitat einzufügen.

9.2 Spezielle Zeichen in HTML

Auch der beste Zeichensatz kann aber ein kleines Problem nicht lösen. Wie wir ja inzwischen gründlich wissen, bestehen die Anweisungen von HTML aus Befehlswörtern, die in spitzen Klammern geschrieben werden. Der Browser wird also beim Lesen einer sich öffnenden spitzen Klammer „<“ erwarten, dass ein Befehl folgt.

Was machen wir aber, wenn wir eine spitze Klammer im Text darstellen wollen? Das geht mit den bisher diskutierten Mitteln nicht. Die Lösung heißt *Fluchtsymbol*. Dazu sucht man sich ein beliebiges, anderes Zeichen und legt fest, dass dieses Zeichen benutzt wird, um eine Ersatzdarstellung für alle die Zeichen einzuleiten, die sich aus irgendeinem Grund nicht unmittelbar darstellen lassen.

Das Wort *Fluchtsymbol* erinnert an ein spezielles Zeichen, das die Steuerung von Bildschirmen einleitete. Sie finden es auch heute noch auf Ihrer Tastatur mit dem Kürzel *ESC* (engl. escape / flüchten).

Wir benötigen mindestens vier Ersatzdarstellungen für die folgenden Zeichen: die beiden spitzen Klammern, das Anführungszeichen, das den Wert der Parameter einleitet und abschließt, und natürlich müssen wir das Fluchtsymbol anders als normal darstellen. Als Fluchtsymbol wird das kaufmännische *Und*-Zeichen (auch Ampersand genannt) „&“ benutzt.

9.2.1 Ersatzdarstellungen

Die Ersatzdarstellung eines nicht darstellbaren Zeichens besteht aus drei Teilen: dem Fluchtsymbol „&“, einem Namen oder einer Zahl und dem abschließenden „;“ (Strichpunkt).

Gewünschtes Zeichen	Ersatzdarstellung
„ (gerades Anführungszeichen)	" (quotation mark)
< (spitze Klammer auf)	< (less than / kleiner als)
> (spitze Klammer zu)	> (greater than / größer als)
& (kaufmännisches Und)	& (ampersand)

Tabelle 9.1: Grundlegende Sonderzeichen in HTML

Mit diesen Sonderzeichen lassen sich nun alle Texte im westlichen Zeichensatz für HTML darstellen.

U-EXP01: Ersatzdarstellung für spitze Klammern

Schreiben Sie eine HTML-Datei (HTML-Seite), die den Umgang mit den vier Sonderzeichen erklärt. Dabei können Sie im Erklärungstext die Ersatzdarstellungen benutzen, die dann der Browser beim Anzeigen wieder als normale Buchstaben darstellt.

Nicht gelöst haben wir bisher die Darstellung von gelegentlich vorkommenden griechischen Sonderzeichen, z.B. einem alpha („α“) oder Beta („β“) oder einem anderen Sonderzeichen wie („©“ oder „®“).

Diese Zeichen stehen zumindest nicht auf meiner Tastatur. Daher wird die Ersatzdarstellung auf alle Zeichen ausgedehnt, die auf einer Tastatur nicht zu finden sind. Auf deutschen Tastaturen gibt es Umlaute, auf amerikanischen nicht. Daher werden amerikanische Programme oft unsere Umlaute mit einer Ersatzdarstellung benutzen.

Unser „Ä“ wird dann zu einem *Ä* (großes A mit Umlaut). Das kleine „ä“ beginnt dann in der Ersatzdarstellung mit einem kleinen „a“. Dem gleichen Muster folgen die Ersatzdarstellungen für „Ö“ und „Ü“. Bleibt noch das „ß“ (scharfes s), das als *ß* beschrieben wird.

Die Ersatzdarstellungen verwenden übrigens nur die ersten 127 Zeichen im ASCII/Latin 1 Code, so dass sie immer mit den Tasten einer Tastatur zu schreiben sind.

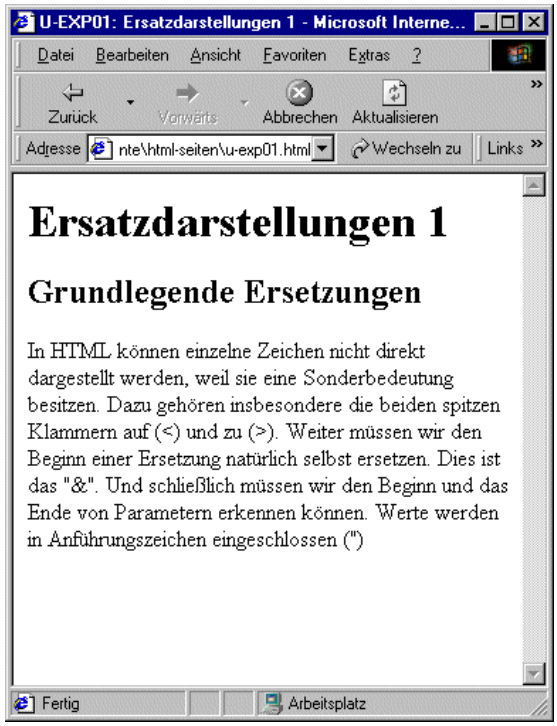

Abbildung 9.1: Grundlegende Ersetzungen

Ein spezielles Zeichen sollten wir noch erwähnen. Es gibt ein spezielles Leerzeichen, das immer dann benutzt wird, wenn ein leerer Text benötigt wird. Das Zeichen ist * * (non breaking space / Leerzeichen ohne Umbruch). Sein Name beschreibt die hauptsächliche Verwendung. Will man mehrere Wörter zu einem Begriff zusammenfassen, der nicht am Zeilenende durch einen Zeilenumbruch getrennt wird, fügt man zwischen den Wörtern das spezielle Leerzeichen ein. Als Beispiel mag die Produktlinie von Microsoft dienen (*Windows 95*).

Die ganze Liste der Ersatzdarstellungen ist ziemlich lang. Sie finden Sie in den Dokumenten zu den HTML-Standards auf dem Server *www.w3c.org*.

U-EXP02: Umlaute in Ersatzdarstellung

Schreiben Sie eine HTML-Datei, die den Umgang mit den Umlauten beschreibt. Verwenden Sie Ersatzdarstellungen.

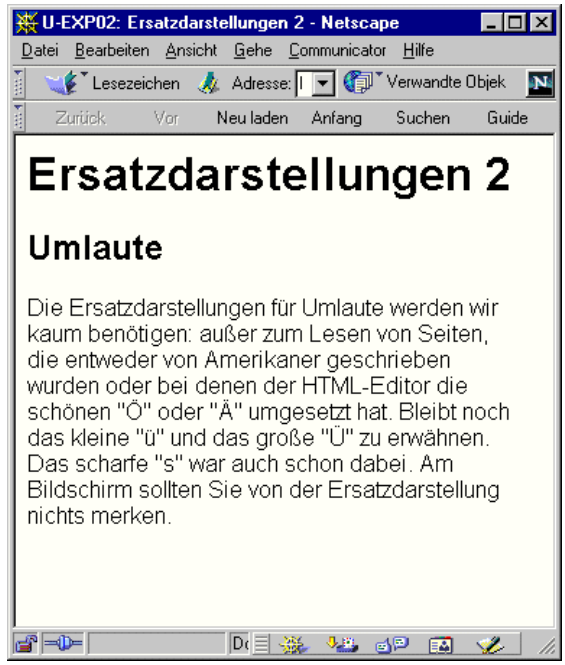

Abbildung 9.2: Umlaute in Ersatzdarstellung

9.2.2 Ersatzdarstellungen mit numerischen Werten

Eine andere Art, ein Zeichen darzustellen, besteht darin, einfach seinen numerischen Wert anzugeben. Jedes Zeichen wird intern als Nummer dargestellt. Kennt man diese Nummer, kann man im Text für die Ersatzdarstellung ein Nummerzeichen („#“) benutzen und die Zahl in dezimaler Darstellung anfügen.

Das Symbol „©“ kann auch durch die Ersatzdarstellung „©“ angegeben werden. Die Zahlen können für *ASCII*-Zeichen bis 127 groß werden, für *Latin 1*-Zeichen bis 255 und für *UniCode*-Zeichen bis ca. 65000.

Aber zur Darstellung muss natürlich immer der Rechner mit Betriebssystem und Treiber zur Darstellung in der Lage sein.

U-EXP03: Ersatzdarstellungen mit Nummern

Ergänzen Sie eines der Beispiele zur Ersatzdarstellung und fügen Sie einen Copyright-Vermerk ein. Das Symbol für das Copyright soll mit einer dezimalen Zeichennummer angegeben werden.

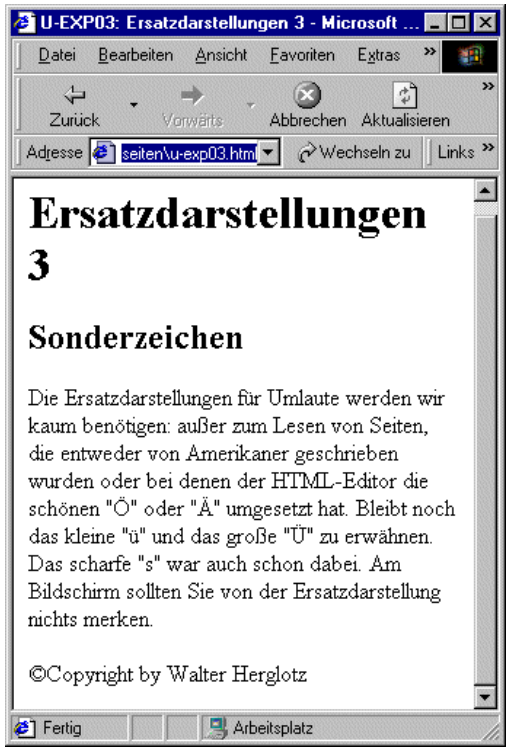

Abbildung 9.3: Numerische Ersatzdarstellung

9.3 UniCode und Meta-Anweisung

Die bisherigen Experimente sollten eigentlich problemlos funktioniert haben, da sie nur grundlegende Funktionen von HTML verwendet haben. Mit den nächsten Experimenten werden wir uns in das neue Feld des universellen Zeichensatzes *UniCode* wagen.

Einmal angenommen, Sie wollen ein Zeichen im Text benutzen, das eben nicht mehr Teil des üblichen westlichen *Latin 1*-Zeichensatzes ist. Das kann nur geschehen, wenn wir den Zeichensatz für die ganze Datei ändern.

Damit kann man dann z.B. russische oder chinesische Texte auf entsprechend eingerichteten Rechnern betrachten.

Bisher haben wir in der Diskussion über Zeichensätze gesehen, dass der Server dem Browser Bescheid geben kann, welcher Zeichensatz zu verwenden ist. In unserem Fall verwenden wir keinen Server, der unseren Browser informieren könnte. Die Information über den gewünschten Zeichensatz müsste also in unserem Beispiel eingefügt werden.

HTML kennt den Begriff der Meta-Anweisungen im Kopf einer HTML-Datei. In diesen Anweisungen können Aussagen über den folgenden Inhalt in der <body>-Anweisung gemacht werden.

Um einen Zeichensatz für den Inhalt vorzugeben, sollten Sie die folgende Anweisung in die <head>-Anweisung jeder Datei mit aufnehmen.

```
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
```

Die Zeile bedeutet, dass eine Aussage über den folgenden Inhalt gemacht wird (*meta*). Die Aussage entspricht einem Befehl, der auch im Rahmen der Kommunikation zwischen Server und dem Browser (dem Client) stattfindet („http-equiv“) und der schließlich den Inhalt nach Art und Zeichensatz näher beschreibt.

Der Kommunikationsbefehl legt die Art des Inhaltes fest („content-type“). Unsere Datei ist eine Textdatei im HTML-Format. Und dann wird zusätzlich festgelegt, dass der Zeichensatz *ISO-8859-1* zu verwenden ist. Hinter dieser Nummer verbirgt sich der immer wieder erwähnte Standard *Latin 1*.

Damit haben wir dem Browser gesagt, welchen Zeichensatz wir wünschen. In der Reihe der Zeichensätze gibt es verschiedene Endnummern, die z.B. auf kyrillische Zeichen verweisen.

U-EXP04: Bestimmung des Zeichensatzes

Fügen Sie die gerade beschriebene Zeile in eines der letzten Experimente ein. Wenn Sie nun die gespeicherte Datei in den Browser laden, sollte sich erst einmal nichts geändert haben. Schließlich haben wir nur beschrieben, dass wir exakt den normalen Zeichensatz für HTML benutzen wollen.

Diese Zeile sollte in Zukunft in keiner Ihrer Dateien fehlen.

Nun wenden wir uns im nächsten Schritt einmal einem Zeichen zu, das sicher nicht in unserem *Latin 1*-Zeichensatz vorhanden ist. Im HTML Standard 4.0 steht das folgende Beispiel, das wir nachprüfen werden.

Es sollen in einem Text die beiden Zeichen &#229 und &#1048 dargestellt werden. Das erste Zeichen sollte keine Probleme machen, da der Code unterhalb von 255 liegt. Aber das zweite würde mehr als das eine Byte benötigen, das der normale Zeichensatz verwendet. Hier müssten mindestens zwei Bytes pro Zeichen benutzt werden.

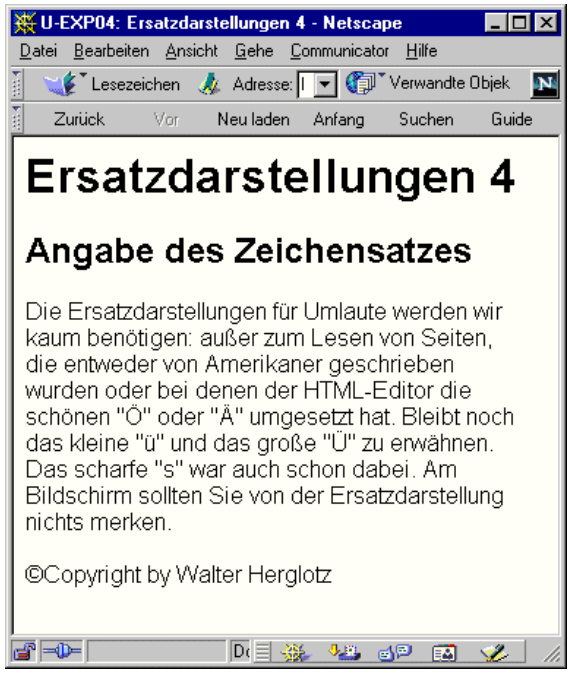

Abbildung 9.4: Vorgegebener Zeichensatz

U-EXP05: Umgang mit Sonderzeichen

Fügen Sie die beiden angegebenen Ersatzdarstellungen in das letzte Beispiel ein. Im Kopf (`<head>`) sollte die besprochene `<meta>`-Anweisung stehen, die den Zeichensatz vorgibt.

Betrachten Sie das Ergebnis wieder im Browser. Das erste Zeichen sollte dargestellt werden (ein a mit einem Kringel oben drauf), das zweite nur als ein Fragezeichen. Anscheinend ist es gar nicht so einfach, Zeichen oberhalb eines numerischen Wertes von 255 darzustellen.

Wenn wir das zweite Zeichen auch am Bildschirm sehen wollen, dann müssen wir den Zeichensatz der Datei umschalten können. Immer vorausgesetzt, das Betriebssystem spielt mit, können wir den *UniCode*-Zeichensatz anfordern.

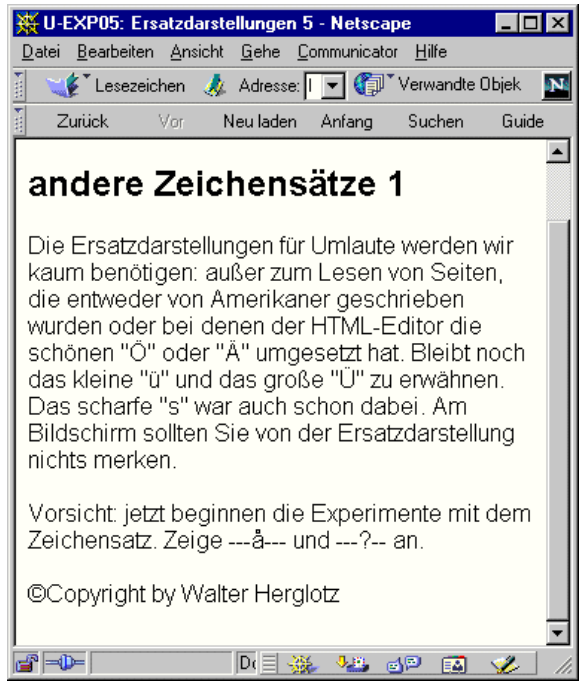

Abbildung 9.5: Sonderzeichen in 8- und 16-Bit

9.3.1 UniCode in UTF-8 Darstellung

Der universelle Zeichensatz kennt für seine vielen Möglichkeiten unterschiedliche Verfahren, die Texte darzustellen. Wir wollen die Darstellung UTF-8 anfordern. UTF steht für „UniCode Transformation Format / UniCode Darstellungs-Format“ mit 8 Bit Zeichen.

Dieses Format verwendet 1 Byte/Zeichen für alle Zeichen, deren Code im Bereich von 0 bis 127 liegt. Darüber werden zwei Zeichen benutzt.

U-EXP06: Zeichensätze einstellen

Ersetzen Sie im vorgegangenen Beispiel in der `<meta>`-Anweisung den angeforderten Zeichensatz. Statt *ISO-8859-1* soll nun der *utf-8* angefordert werden.

Betrachten Sie nach dem Speichern der Datei das Ergebnis im Browser. Bei den am meisten verwendeten Browsern (Netscape und Microsoft) sollte es eine Version besser als 4.0 sein. Vergessen Sie nicht, die Datei im Browser neu zu laden.

Da die normalen Editoren noch nicht korrekt mit UniCode umgehen oder das gewünschte *UniCode*-Format UTF-8 nicht darstellen können (zumindest meine nicht), geben Sie besser Umlaute in einer Ersatzdarstellung an.

Auf meinem Rechner (einer Maschine mit Windows NT) und auch auf meiner Linux Maschine (SuSE 6.4) wird nun als zweites Zeichen ein kyrillisches großes „I“ angezeigt. Es schaut für mich einem großen „H“ ähnlich, wobei der Mittelstrich schräg von links nach rechts oben läuft.

Leider klappt nun die sonst normale Unterstützung für die 8-Bit-Zeichen der deutschen Umlaute nicht mehr; die Ersatzdarstellungen funktionieren noch. Der Grund liegt darin, dass die UTF-8 Darstellung pro Zeichen mehr als ein Byte verwendet – mit Ausnahme der ASCII-Zeichen.

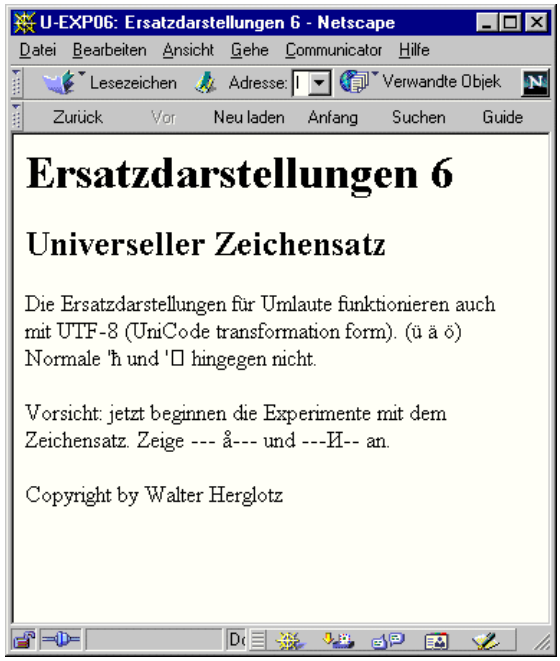

Abbildung 9.6: UniCode für Browser

Unsere Experimente funktionieren also auf den Rechnern solange problemlos auch mit einem UTF-8-Darstellung, wie wir uns mit Ersatzdarstellungen begnügen. Natürlich könnten wir auch einen UniCode-Editor benutzen, der in der Lage ist, das UTF-8-Format zum Speichern zu benutzen. Aber die sind noch nicht so geläufig.

Man kann sicher nur hoffen, dass der UniCode baldmöglichst allgemein akzeptiert und auf allen Rechnern zum normalen Zeichensatz wird.

Für den Moment sollten wir in unseren HTML-Dateien beim bewährten Zeichensatz *ISO-8859-1* bleiben.

Im nächsten Kapitel

Aber nun wenden wir uns endlich den Möglichkeiten zu, Bilder in unsere Webseite zu integrieren. Mit den Bildern werden wir auch mehr von Webseiten als von Dateien sprechen. Webseiten werden in den nächsten Kapitel oft aus mehreren Dateien bestehen. Den Rahmen bildet die HTML-Datei, die Verweise auf Bilddateien enthalten kann. Der Browser wird dann alle Dateien, die zusammen die Webseite bilden, laden und gemeinsam darstellen.

9.4 Lösungen

U-EXP01: Ersatzdarstellung für spitze Klammern

```
<html>
<head>
<title>U-EXP01: Ersatzdarstellungen 1</title>
</head>
<body >
<h1>Ersatzdarstellungen 1</h1>
<h2>Grundlegende Ersetzungen</h2>
<p>
In HTML können einzelne Zeichen nicht direkt dargestellt werden, weil sie
eine Sonderbedeutung besitzen. Dazu gehören insbesondere die beiden spitzen
Klammern auf (&lt;) und zu (&gt;). Weiter müssen wir den Beginn einer
Ersetzung natürlich selbst ersetzen. Dies ist das "&amp;". Und schließlich
müssen wir den Beginn und das Ende von Parametern erkennen können. Werte
werden in Anführungszeichen eingeschlossen (&quot;)
</p>
</body>
</html>
```

U-EXP02: Umlaute in Ersatzdarstellung

```
<html>
<head>
<title>
U-EXP02: Ersatzdarstellungen 2</title>
</head>
```

```
<body >
<h1>Ersatzdarstellungen 2</h1>
<h2>Umlaute</h2>
<p>Die Ersatzdarstellungen f&uuml;r Umlaute werden wir kaum ben&ouml;tigen:
außer zum Lesen von Seiten, die entweder von Amerikaner geschrieben wurden
oder bei denen der HTML-Editor die sch&ouml;nen "&Ouml;" oder "&Auml;"
umgesetzt hat.
Bleibt noch das kleine "&uuml;" und das gro&szlig;e "&Uuml;" zu
erw&auml;hnen. Das scharfe "s" war auch schon dabei.
Am Bildschirm sollten Sie von der Ersatzdarstellung nichts merken.
</p>
</body>
</html>
```

U-EXP03: Ersatzdarstellungen mit Nummern

```
<html>
<head>
<title>
U-EXP03: Ersatzdarstellungen 3</title>
</head>
<body >
<h1>Ersatzdarstellungen 3</h1>
<h2>Sonderzeichen</h2>
<p>Die Ersatzdarstellungen f&uuml;r Umlaute werden wir kaum ben&ouml;tigen:
außer zum Lesen von Seiten, die entweder von Amerikaner geschrieben wurden
oder bei denen der HTML-Editor die sch&ouml;nen "&Ouml;" oder "&Auml;"
umgesetzt hat.
Bleibt noch das kleine "&uuml;" und das gro&szlig;e "&Uuml;" zu
erw&auml;hnen. Das scharfe "s" war auch schon dabei.
Am Bildschirm sollten Sie von der Ersatzdarstellung nichts merken.
</p>
<p>
&#169;Copyright by Walter Herglotz</p>
</body>
</html>
```

U-EXP04: Bestimmung des Zeichensatzes

```
<html>
<head>
<title>
U-EXP04: Ersatzdarstellungen 4</title>
</head>
<body >
<h1>Ersatzdarstellungen 4</h1>
<h2>Angabe des Zeichensatzes</h2>
```

```
<p>Die Ersatzdarstellungen f&uuml;r Umlaute werden wir kaum ben&ouml;tigen:
außer zum Lesen von Seiten, die entweder von Amerikaner geschrieben wurden
oder bei denen der HTML-Editor die sch&ouml;nen "&Ouml;" oder "&Auml;"
umgesetzt hat.
Bleibt noch das kleine "&uuml;" und das gro&szlig;e "&Uuml;" zu
erw&auml;hnen. Das scharfe "s" war auch schon dabei.
Am Bildschirm sollten Sie von der Ersatzdarstellung nichts merken.
</p>
<p>
&#169;Copyright by Walter Herglotz</p>
</body>
</html>
```

U-EXP05: Umgang mit Sonderzeichen

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>
U-EXP05: Ersatzdarstellungen 5</title>
</head>
<body >
<h1>Ersatzdarstellungen 5</h1>
<h2>andere Zeichensätze 1</h2>
<p>Die Ersatzdarstellungen f&uuml;r Umlaute werden wir kaum ben&ouml;tigen:
außer zum Lesen von Seiten, die entweder von Amerikaner geschrieben wurden
oder bei denen der HTML-Editor die sch&ouml;nen "&Ouml;" oder "&Auml;"
umgesetzt hat.
Bleibt noch das kleine "&uuml;" und das gro&szlig;e "&Uuml;" zu
erw&auml;hnen. Das scharfe "s" war auch schon dabei.
Am Bildschirm sollten Sie von der Ersatzdarstellung nichts merken.
</p>
<p>
Vorsicht: jetzt beginnen die Experimente mit dem Zeichensatz. Zeige ---
&#229--- und ---&#1048-- an.
</p>
<p>
&#169;Copyright by Walter Herglotz</p>
</body>
</html>
```

U-EXP06: Zeichensätze einstellen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
```

```
   <title>U-EXP06: Ersatzdarstellungen 6</title>
</head>
<body>
<h1>
Ersatzdarstellungen 6</h1>
<h2>
Universeller Zeichensatz</h2>
Die Ersatzdarstellungen fÃ¼r Umlaute funktionieren auch mit UTF-8 (UniCode
transformation form). (&uuml; &auml; &ouml;)  Normale 'Ä' und 'Ö' hin-
gegen nicht.
<p>Vorsicht: jetzt beginnen die Experimente mit dem Zeichensatz. Zeige
--- &#229;--- und ---&#1048;-- an.
<p>Copyright by Walter Herglotz
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

10 Bilder

10.1 HTML-Seiten mit Graphiken

Die bisher besprochenen Textseiten eignen sich zusammen mit den texttypischen Formatierungen ganz ausgezeichnet für umfangreiche wissenschaftliche Arbeiten. Aber es gilt natürlich auch, dass einige Bilder jede Veröffentlichung bereichern können, denn ein Bild sagt bekanntlich mehr als tausend Worte. Mit den Bildern begann für HTML-Seiten der Einstieg in eine multimediale Präsentation.

Wir wollen für den Moment noch bei unseren einfachen Text-Editoren bleiben. Bald aber steigen wir auf HTML-Editoren um.

10.2 Dateiformate für Graphiken

10.2.1 Das GIF-Format

Die derzeit verwendeten Graphiken sind Pixelgraphiken. Vektorielle Dateiformate werden derzeit nur über Zusatzprogramme unterstützt. Allerdings wird bei W3C an einem vektoriellen Dateiformat gearbeitet.

Als Graphikformate werden heute standardmäßig *JPEG* (joint photographers expert group / gemeinsame Expertengruppe der Photographen) und *GIF* (graphic interchange format / graphisches Austauschformat) benutzt. *GIF* wurde hauptsächlich von CompuServe verwendet. *GIF* wird aber leider durch Patentrechte geschützt. Und die Rechte werden auch eingefordert. Damit ist die Zukunft von *GIF* etwas ungewiss. Auf Grund dieser Situation wurde eine patentfreie, verbesserte Version entwickelt, die *PNG* (portable network graphics / portable Netzwerk Graphik) genannt wird.

Das derzeit vermutlich häufigste Format im WWW ist vermutlich GIF89a – GIF in der Version von 1989.

Als Daumenregel lässt sich sagen: GIF-Dateien werden für technische Bilder verwendet. Sollten Sie also in einem pixelorientierten Zeichenprogramm ein Logo erstellen oder ein abstraktes Bild einscannen, eignet sich oft GIF. Der Nachteil von GIF ist die Beschränkung auf 256 Farben. Der Vorteil von GIF ist seine weite Verbreitung und seine Sondereigenschaften. Mit GIF ist eine Animation möglich. In einer Datei können mehrere Bilder und Bildwechselinformationen hinterlegt werden. Browser ab *Netscape 2.0* können animierte GIF-Dateien darstellen. Leider fallen uns animierte Bilder heute häufig am stärksten in den Werbeeinblendung auf.

Die Animationen in GIF-Dateien funktionieren wie das gute, alte „Daumenkino“. Das ist ein kleiner Block voller Zeichnungen, die den Eindruck einer Animation vermitteln, wenn man mit dem Daumen am Rand die einzelnen Bilder schnell aufklappen lässt.

GIF kann "interlaced" d.h. verschachtelt erstellt werden. Zuerst wird dann beim Laden das Bild nur grob und mit fortschreitender Dauer immer feiner dargestellt. Dies ist für Betrachter besonders bei langsamen Verbindungen sehr angenehm. Und schließlich kann eine Farbe als *transparent* definiert werden. Diese Farbe wird dann beim Darstellen eben nicht mit dargestellt, so dass der jeweilige Hintergrund durchscheint.

Wenn Sie also ein Bild auf einer Seite sehen, das aussieht, als wäre es Teil der Seite, dann war es vermutlich ein GIF-Bild.

10.2.2 JPEG - joint photographer expert group

Dieses Dateiformat wird besonders für natürliche Bilder verwendet. Die Vorteile dieses Dateiformats kommen aus der Definition durch Photographen. Es erlaubt volle Farbauflösung mit 16,7 Millionen Farben (24 Bit) und einer einstellbaren Kompressionsstufe. Damit lässt sich die Qualität der Darstellung einstellen.

Die verwendete Kompression ist allerdings verlustbehaftet. Damit geht ein geringer Anteil der Qualität verloren. Für alle Naturaufnahmen ist JPEG am besten geeignet.

10.2.3 PNG - portable network graphic

Um die Probleme mit dem geschützten Format GIF zu umgehen, wurde eine verbesserte Version von GIF mit 24-Bit Farbtiefe definiert. Seine Vorteile sind einfach zu beschreiben: das Format ist lizenzfrei. Leider gibt es noch Nachteile: die Dateien sind größer als bei GIF und es wird noch nicht von allen Browsern unterstützt.

10.3 HTML und Bilder

Bilder setzen natürlich eine graphische Benutzeroberfläche voraus. Dann aber machen Sie viele Webseiten deutlich attraktiver. Kein Wunder, dass mit dem Auftauchen des ersten graphischen Browsers *Mosaic* vom Gründer der Fa. Netscape ein sprunghafter Anstieg der WWW-Benutzerzahlen einherging.

Der Browser verbreitete sich in ungeahnter Geschwindigkeit über das Netz in unzählige Hochschulen und Labors. Mit dem Auftauchen von MOSAIC war der Grundstein für unsere heutigen Webseiten gelegt.

10.4 Bilder einfügen

Zum Einfügen von Bildern dient die `<img>`-Anweisung. Beachten Sie, dass die `<img>`-Anweisung keine Ende-Marke besitzt. Sie verhält sich damit ähnlich wie `<br>`. Bilder lassen sich somit in den Textfluss einfügen. Ein Bild könnte sich wie ein einzelnes Zeichen verhalten. Es gab genügend Webseiten, die nicht darstellbare Zeichen wie z.B. mathematische Symbole, durch kleine Graphiken ersetzt haben.

Der wichtigste Parameter der `<img>`-Anweisung ist die Angabe der Bildquelle mit *src*. Im einfachsten Fall ist dies einfach ein Dateiname. Dabei wollen wir für den Moment annehmen, dass die Bilddatei im gleichen Verzeichnis wie die zugehörige HTML-Datei liegt.

Die allgemeine Form wird statt des einfachen Dateinamens einen so genannten URL (uniform resource locator / einheitliche Angabe einer Quelle) verwenden.

Anweisung / Parameter	Funktion
<img src= „datei.gif“> (source / Quelle)	Angabe der Quelle mit URL (hier: ganzer Dateiname)
< .. align= „left	right“ ..>	Bild links oder rechts, Text fließt um Bild
< .. align= „bottom	middle	top“	Bildlinie in der gleichen Zeile, Text fließt nicht
< .. alt= „erklärender Text“ ..>	Wichtig: Beschreibung des Bildes
< .. border= „Pixelanzahl“ ..>	Rahmen um das Bild mit Dicke
< ..width= „Breite“ .. height= „Höhe“ ..>	Platzreservierung für das Bild, evtl. Verzerrung
< ..vspace=„oben/unten Platz“ hspace=„rechts/links“.>	Freier Raum um das Bild (Pixel)

Tabelle 10.1: Bildsteuerung

Es ist guter Stil, mit dem Parameter *ALT* eine Erklärung des Bildes hinzuzufügen. Nicht jeder hat einen graphischen Browser (das gibt es immer noch) und nicht jeder schaltet bei langsamen Verbindungen die Anzeige der Bilder im Browser ein. Manchmal will man vielleicht den vielen Werbeeinblendungen entgehen, wenn man etwas Spezielles sucht. Außerdem kann man den Text im Browser nutzen, um ihn gelb unterlegt anzuzeigen, wenn die Maus darüber steht.

Beim ersten Aufbau einer neu angewählten Seite wird vom Browser zuerst der Text geladen, danach die Bilder. Er kann nur dann genügend Platz für die Bilder im Text freihalten, wenn er die Größe kennt. Deshalb ist es immer sinnvoll, die Breite und Höhe des Bildes in Pixeln in der `<img>`-Anweisung mit anzugegeben. Jetzt kann der Browser beim Darstellen des Textes bereits den notwendigen Platz reservieren. Ohne diese Angabe muss die ganze Seite nach dem Laden der Bilder neu formatiert werden. Das führt bisweilen zu einem unangenehmen Flackern der Darstellung.

Es ist möglich, falsche Werte für Bildgrößen anzugeben. Der Browser wird dann versuchen, die Größe des Bildes der reservierten Fläche anzupassen. Wenn er dies kann, dann erhalten wir verzerrte Bilder. Auch die mögliche Angabe in Prozent des umgebenden Containers, also meist der Seite, ist damit problematisch, da in den meisten Fällen damit ebenfalls eine Verzerrung verbunden ist.

Ein Fall für sich sind bei der `<img>`-Anweisung die Angaben zur Ausrichtung. Hier werden in einem Parameter gleich zwei Dinge gesteuert. Benutzt man die Werte *left* oder *right*, dann wird das Bild links oder rechts positioniert, und der Text fließt um das Bild herum.

Die anderen Angabe positionieren das Bild relativ zur Schreiblinie der momentanen Zeile. Der Text fließt nicht um das Bild. Hier wird das Bild wie ein Buchstabe im Lauf der Zeile betrachtet. Man könnte für unbekannte Zeichen kleine Bilder erstellen und sie anstelle des Zeichens in den Text einfügen.

B-EXP01: Bilder in HTML

Schreiben (oder verändern) Sie eine HTML-Datei. Sie soll nun ein Bild enthalten. Der Text soll rechts um das Bild fließen. Beachten Sie momentan, dass die Bilddatei im gleichen Verzeichnis wie die HTML-Datei liegen sollte.

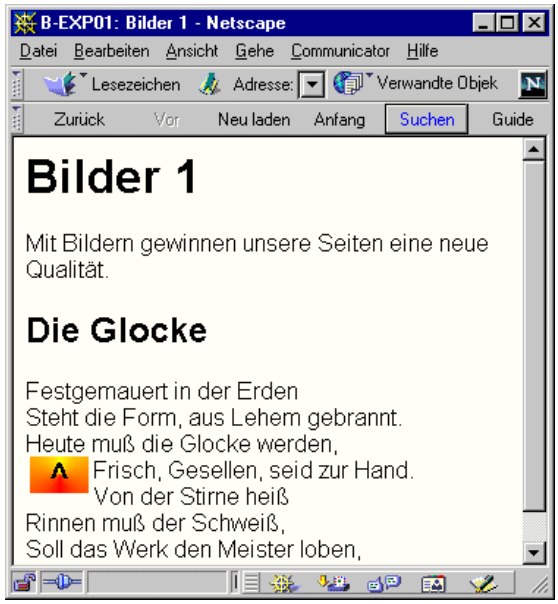

Abbildung 10.1: Einfügen von Bildern

10.5 Bildquellen

Wo kommen eigentlich die Bilder her? Bilder gehören zu den geschützten geistigen Leistungen. Es ist also rechtlich erst einmal nicht zulässig, irgendein Bild aus dem WWW zu holen und in den eigenen Seiten zu verwenden.

Technisch ist es einfach, ein beliebiges Bild aus einer angezeigten Seite aus dem WWW lokal zu speichern. Meist genügt ein Klick mit der rechten Maustaste auf das Bild und danach die Auswahl des Punktes GRAPHIK/BILD SPEICHERN UNTER ....

Bilder können Sie mit einem beliebigen Pixel-orientierten Zeichenprogramm erstellen. Beispiele sind Corel Paint, Paint Shop, GIMP u.a. Mit den Programmen, die im Lieferumfang von Windows NT enthalten sind, hatte ich keine Freude. Paint kann keine GIF-Dateien bearbeiten und stürzt bei mir ab, wenn man es versucht, und das sogenannte *imaging*-Programm kann ebenfalls mit GIF nichts anfangen.

Eine weitere Quelle sind Bildschirmphotos (engl. *screenshots*). Unter Windows genügt die Taste Druck, um den ganzen Bildschirm in die Zwischenablage zu kopieren oder Alt + Druck, um das aktive Fenster zu kopieren. Unter KDE (Linux) gibt es z.B. das Programm *KSnapshot*.

Diese Bildschirmkopien werden danach entweder in ein Graphikprogramm eingefügt und als Datei gespeichert oder gleich als Datei gesichert (KSnapShot).

Meine bevorzugte Quelle ist eine Digital-Kamera.

Mit Graphikprogrammen kann man auch Veränderungen vornehmen und die Größe der Bilder bestimmen. Die meisten Pixel-Bilder verlieren jedoch erheblich an Aussehen, wenn die Größe verändert wird. Meist verwendet man das Ausschneiden wichtiger Teile des Bildes, um so die Größe anzupassen.

B-EXP02: Größenangaben

Ergänzen Sie das bisherige Bildbeispiel um weitere Bilder. Ergänzen Sie die passenden Größenangaben.

Die Größe des Bildes in Pixel sollte bekannt sein. Wenn man es nicht weiß, muss man eines der Graphikprogramme bemühen oder einen HTML-Editor verwenden.

Unter jedem Betriebssystem gibt es kostenlose oder zumindest preiswerte Editoren. Mit dem Netscape Communicator wird der Composer mitgeliefert und mit dem Internet Explorer kommt Frontpage Express.

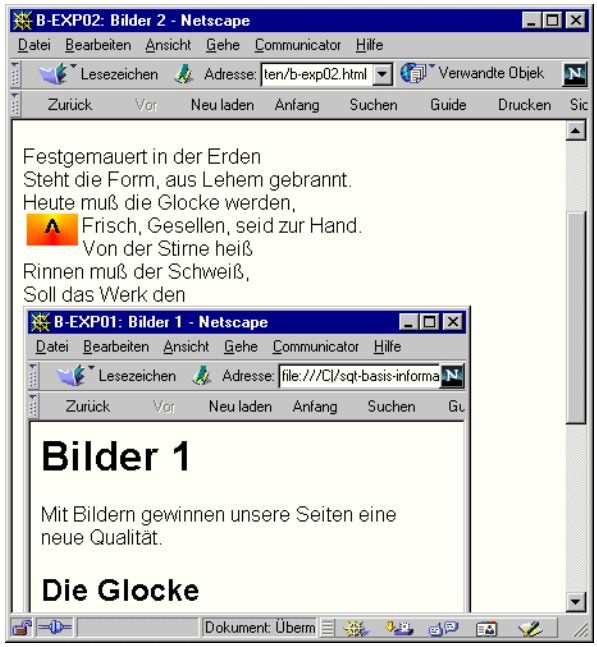

Abbildung 10.2: Bilder in einer Webseite

10.6 Spezielle Funktionen von GIF-Bildern

10.6.1 Transparenz

GIF-Bilder kennen noch weitere Spezial-Funktionen. Eine davon ist die Transparenz. Man kann in einem GIF-Bild eine Farbe auswählen und diese Farbe als transparent erklären. Im Graphik-Editor wird man kaum den Unterschied erkennen. Lädt man aber eine GIF-Datei mit einer transparenten Farbe, dann werden Bildpunkte mit dieser Farbe nicht dargestellt. Statt dessen verwendet das Anzeigeprogramm, also unser Browser, die eigene Hintergrundfarbe.

Es entsteht der Eindruck, als könnte man durch Teile des Bildes hindurch sehen. Das simple Ausgangsbild soll wie im nächsten Bild gezeigt nach dem Laden im Browser so aussehen. Das Bild wurde also als Bildschirmkopie gespeichert und danach der wichtige Teil ausgeschnitten.

Im zweiten Bild wurde eine Farbe, die Hintergrundfarbe als transparent erklärt. Im Browserfenster sieht dann das gleiche Bild ganz anders aus, weil der Hintergrund des Browsers durchscheint. Die leicht unterschiedlichen Größen sind nur durch die Bearbeitung entstanden.

Die Transparenz kann man sehr gut einsetzen, um elegant wirkende Webseiten zu entwickeln.

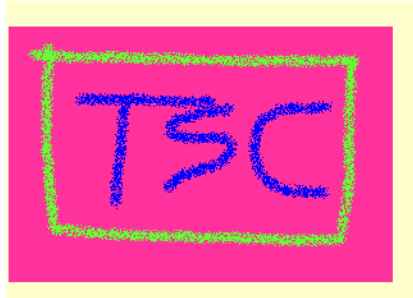

Abbildung 10.3: Undurchsichtiger Hintergrund

Abbildung 10.4: Transparenter Hintergrund

10.6.2 Animationen

Eine der spannendsten Eigenschaften des GIF-Formats ist seine Fähigkeit, mehrere Bilder zusammen mit Bildwechsel-Informationen in einer Datei zu speichern.

Versteht nun ein Anzeige-Programm dieses Format, kommen wir mit sehr einfachen Mitteln zu kleinen Animationen oder Daumen-Kino-Filmen.

Verschiedene Editoren erlauben es, diese Animationen zu erstellen. In einem Buch ist es natürlich unmöglich, bewegte Bilder zu zeigen. Aber auf den Webseiten der meisten großen Informationsanbieter (CNN, Stern, Focus etc.) wimmelt es von animierten Werbeeinblendungen.

Viele neuere Versionen der Graphikeditoren unterstützen die Erstellung von animierten Graphiken. Auch Microsoft bot eine Zeit lang einen Editor an (GIFSETUP.EXE).

B-EXP03: Animationen

Erstellen Sie eine Animation in einer GIF-Datei oder kopieren Sie eine Animation (oft Werbung) aus dem WWW. Schreiben Sie danach eine HTML-Seite, die diese Animation benutzt.

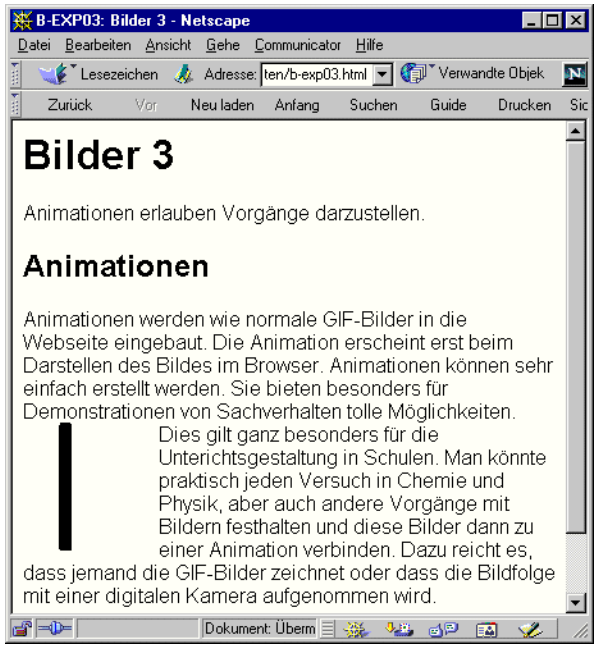

Abbildung 10.5: Einfache Animation

10.7 Arbeiten mit Editoren

Benutzen wir an dieser Stelle einmal einen HTML-Editor. Welchen Editor Sie benutzen möchten, ist meist eine Frage der Anwendung. Für gelegentliche Autoren bieten die Editoren von Netscape (Composer) und Microsoft (Frontpage Express) zumindest eine Grundfunktionalität. Einfache bis mittelschwere Seiten lassen sich damit erstellen.

Professionelle Nutzer werden eher zu Editoren aus dem Multimedia Bereich von Adobe oder Macromedia greifen. Informieren Sie sich am besten auf den Webseiten der Hersteller. Die Produkte ändern sich so häufig und auch die Aufnahme bei den Anwendern, dass gute Empfehlungen in einem Buch kaum sinnvoll sind.

Sie sollten nur auf eines achten: sehen Sie sich den von einem Editor erzeugten Code genau an. Er sollte keinerlei System-spezifischen Eigen-

schaften verwenden. Sonst können Sie bei Änderungen, Übertragung auf andere Webserver etc. sehr unangenehme Erfahrungen machen.

B-EXP04: Ändern eines Experiments mit einem Editor

Laden Sie das Ergebnis des Experiments B-EXP02 (Datei mit Bild) in Ihren Editor. Meist kann man die Datei in den Browser laden und dann unter dem Menüpunkt „Datei“ auf „Bearbeiten“ klicken.

Eine andere Möglichkeit ist es, das Editor-Programm direkt zu starten und die Datei damit zu laden.

Setzen Sie eine Eigenschaft des Bildes: es soll nun einen vier Pixel großen Rand bekommen.

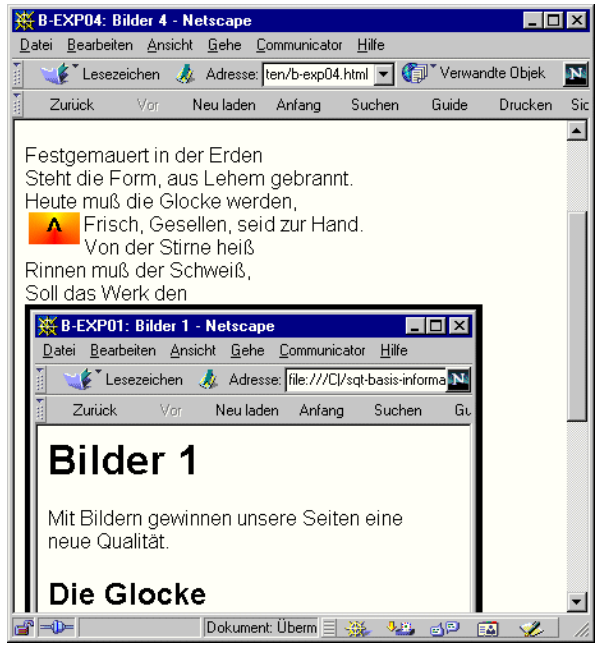

Abbildung 10.6: Rand mit Editor eingefügt

Im Composer (4.7) klickt man auf das Bild mit der rechten Maustaste und wählt die Eigenschaften des Bildes aus (Graphikeigenschaften).

Die Einstellung für die Rahmendicke ist im markierten Bereich.

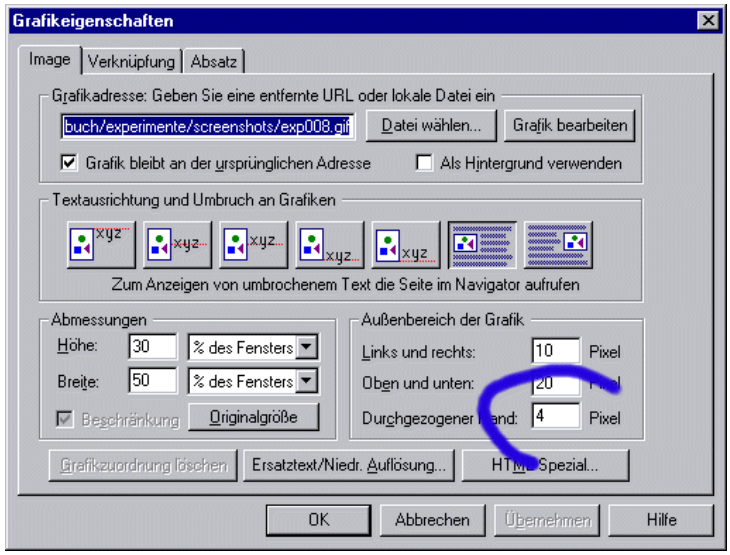

Abbildung 10.7: Rahmen mit dem Composer setzen

Nicht viel anders geschieht dies im Frontpage Express. Auch hier klickt man mit der rechten Maustaste auf das Bild und wählt die Bild-Eigenschaften aus.

Nun muss man nur noch den richtigen Reiter finden und kann dann ebenfalls die Rahmendicke setzen.

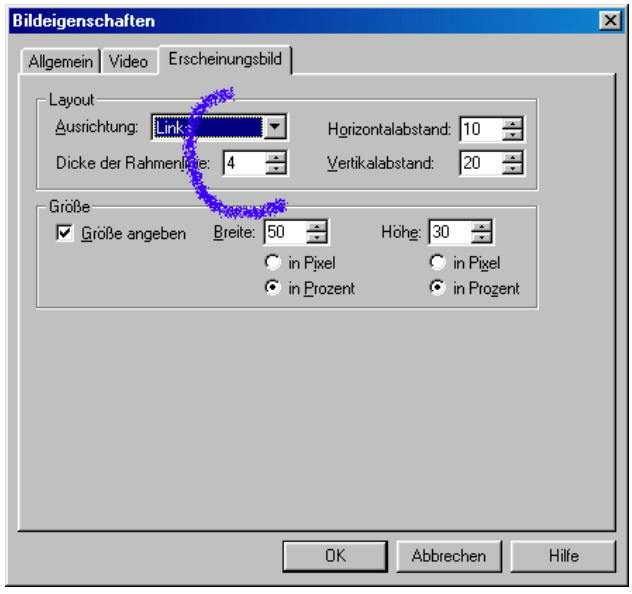

Abbildung 10.8: Rahmen mit Frontpage Express setzen

Verwenden Sie nach Geschmack und Laune für die weiteren Experimente wahlweise den Texteditor oder einen HTML-Editor Ihrer Wahl. Suchen Sie auch einmal im WWW danach. Es gibt noch viele weitere.

Im nächsten Kapitel

Bisher haben wir uns mit der Erstellung von einzelnen Dateien (oder Seiten) mit HTML beschäftigt. Nun wollen wir im nächsten Kapitel einmal mit einem weiteren Kernstück des WWW beschäftigen: mit den Sprüngen (oder Verweisen).

Das Netz im Namen des WWW (world wide web / weltweites Netz) wird schließlich nur die unzähligen Verweise gebildet, die in den verschiedenen Webseiten stehen.

10.8 Lösungen

B-EXP01: Bilder in HTML

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>B-EXP01: Bilder 1</title>
</head>
<body>
<h1>
Bilder 1</h1>
Mit Bildern gewinnen unsere Seiten eine neue Qualit&auml;t.
<h2>
Die Glocke</h2>
Festgemauert in der Erden
<br>Steht die Form, aus Lehem gebrannt.
<br>Heute mu&szlig; die Glocke werden,
<br><img SRC="auf.gif"  align=LEFT> Frisch, Gesellen, seid zur Hand.
<br>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;,
<br>Soll das Werk den Meister loben,
<br>Doch der Segen kommt von oben.
</body>
</html>
```

B-EXP02: Größenangaben

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>B-EXP02: Bilder 2</title>
</head>
<body>
<h1>
Bilder 2</h1>
Mit Bildern gewinnen unsere Seiten eine neue Qualit&auml;t.
<h2>
Die Glocke</h2>
Festgemauert in der Erden
<br>Steht die Form, aus Lehem gebrannt.
<br>Heute mu&szlig; die Glocke werden,
<br><img SRC="auf.gif"  align=LEFT> Frisch, Gesellen, seid zur Hand.
<br>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;,
<br>Soll das Werk den <img SRC="../exp-bilder/b-exp01-1.gif"
height="497" width="350">Meister
loben,
<br>Doch der Segen kommt von oben.
</body>
</html>
```

B-EXP03: Animationen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>B-EXP03: Bilder 3</title>
</head>
<body>
<h1>
Bilder 3</h1>
Animationen erlauben Vorg&auml;nge darzustellen.
<h2>
Animationen</h2>
Animationen werden wie normale GIF-Bilder in die Webseite eingebaut. Die
Animation erscheint erst beim Darstellen des Bildes im Browser. Animationen
k&ouml;nnen sehr einfach erstellt werden. Sie bieten besonders f&uuml;r
```

```
Demonstrationen von Sachverhalten tolle M&ouml;glichkeiten.<img
SRC="strich.gif"  align=LEFT>
Dies gilt ganz besonders f&uuml;r die Unterichtsgestaltung in Schulen.
Man k&ouml;nnte praktisch jeden Versuch in Chemie und Physik, aber auch
andere Vorg&auml;nge mit Bildern festhalten und diese Bilder dann zu einer
Animation verbinden. Dazu reicht es, dass jemand die GIF-Bilder zeichnet
oder dass die Bildfolge mit einer digitalen Kamera aufgenommen wird.
<p>Im Beispiel haben wir (Hallo Markus!) einfach ein paar Striche gemalt
und die Strich-Bilder zusammengesetzt.
</body>
</html>
```

B-EXP04: Ändern eines Experiments mit einem Editor

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>B-EXP04: Bilder 4</title>
</head>
<body>
<h1>
Bilder 4</h1>
Mit Bildern gewinnen unsere Seiten eine neue Qualit&auml;t.
<h2>
Die Glocke</h2>
Festgemauert in der Erden
<br>Steht die Form, aus Lehem gebrannt.
<br>Heute mu&szlig; die Glocke werden,
<br><img SRC="auf.gif"  align=LEFT> Frisch, Gesellen, seid zur Hand.
<br>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;,
<br>Soll das Werk den <img SRC="b-exp01-1.gif" BORDER=4 height=497
width=350>Meister
loben,
<br>Doch der Segen kommt von oben.
</body>
</html>
```

11 Verweise in HTML

Was ist eigentlich der Unterschied zwischen dem Internet und dem WWW? Hört man Menschen zu, die einen Netzzugang haben und dort surfen, dann merkt man schnell, dass die beiden Begriff meist gleichartig verwendet werden. In Wirklichkeit sind dies verschiedene Dinge.

Wenn wir nun anfangen, Verweise in unsere Webseiten einzubauen, dann brauchen wir zumindest ein paar Kenntnisse, wie die Namen für Verweise aufgebaut werden, wie wir andere Rechner erreichen und welche Möglichkeiten für Namen es gibt. Wir wollen uns daher in den nächsten beiden Abschnitten des Kapitels die wichtigsten Begriffe zum Thema Internet und zum WWW ansehen und kurz erläutern.

11.1 Das Internet

Schauen wir uns die Begriffe einmal so weit an, wie es für einen PC-Benutzer sinnvoll ist. Die Begriffe, die auch in den Netzwerk-Einstellungen des eigenen Rechners abgefragt werden könnten, wollen wir kurz beleuchten. Vermutlich wird uns das dann auch helfen, die Adressen des WWW leichter zu verstehen.

Das Internet könnte man genauer Inter-Net schreiben, um die Aufgabe deutlich zu machen. Das Internet verbindet Computernetze. Ein einzelnes Netz kann das Netz einer Firma oder einer Hochschule sein. Diese Netze können technisch sehr unterschiedlich aufgebaut sein. Es gibt noch einen heute häufig vorkommenden Sonderfall. Hier besteht eines der Netze nur aus einem Rechner – Ihrem PC daheim.

Will man nun von einem Netz in ein anderes Daten übertragen, dann benötigt man die Möglichkeit, Daten an einen Adresse zu schicken oder von dort abzuholen. Die Adresse muss dabei zwei Fragen beantworten:

- welches Netz will ich erreichen?
- welchen Rechner im Zielnetz will ich erreichen?

Das weltweit bekannteste Adressierungsverfahren liefert uns das so genannte TCP/IP-Protokoll. IP steht dabei für *Internet-Protocol* und TCP für *Transmission Control Program*. Diese Protokolle liegen wie Schichten aufeinander.

Mit der IP-Schicht kann man Rechner datentechnisch verbinden. Mit der TCP-Schicht kann man sicher Daten übertragen. Schließlich können auch einmal Daten verloren gehen, wenn auf dem Weg zwischen zwei Endpunkten die Vermittlungsrechner überlastet sind.

Auf diesem Netz werden nun viele Dienste angeboten. Dazu gehören sehr bekannte Dienste wie die E-Mail-Übertragung, das Transportieren von Webseiten oder die Übertragung von einzelnen Dateien. Weniger bekannte Dienste sind z.B. Zeitdienste, die extrem präzise Zeitangaben von zentralen Zeitstationen (wie der Physikalisch-Technischen Bundesanstalt) über das Netz verteilen. Die Zeitübertragung wird in vielen Firmen und Forschungseinrichtungen benutzt, um beim Start der Rechner deren interne Uhr genau zu setzen. Jeder Dienst setzt immer ein dafür entwickeltes Protokoll voraus.

Ein beliebiger Rechner, also z.B. Ihr PC daheim, muss im Internet eine genaue Identifikation erhalten. Dazu wird eine IP-Nummer verwendet. Diese Nummer ist die weltweit eindeutige Adresse der Rechnerverbindung. Präziser ausgedrückt, benötigt die Schnittstelle des Rechner, die ins Internet führt, eine solche IP-Adresse. Besitzt ein Rechner mehrere Netzwerk-Anschlüsse, dann benötigt jeder dieser Anschlüsse eine eigene Adresse.

Solange man nur eine Verbindung hat, kann man den Unterschied zwischen Rechner und Rechnerverbindung vernachlässigen.

Die Adresse besteht zurzeit aus vier Bytes, die meist als vier dezimale Zahlen mit Punkten getrennt geschrieben werden (z.B. 192.168.200.97).

Daheim werden Sie vermutlich die Verbindung mit einem so genannten Provider herstellen. Dieser Provider gibt Ihnen eine Telefonnummer, die Sie anrufen können, um darüber eine Verbindung ins Internet zu erhalten. Während diese Verbindung aufgebaut wird, un-

terhalten sich der Rechner des Providers und Ihr eigener Rechner. Dabei wird Ihnen meist eine solche IP-Adresse und weitere Einstellungen für die Dauer einer Verbindung geliehen. Der Dienst ist unter dem Kürzel *DHCP* (dynamic host configuration protocol / Protokoll für die dynamische Konfiguration des Rechners) bekannt.

In größeren Firmen wird oft jedem Rechner seine IP-Adresse fest zugeteilt.

Innerhalb der Adresse verstecken sich jedoch die schon erwähnten Teiladressen für das Netz und den Rechner. Mit Hilfe einer Maske (engl. *subnet mask*) gibt man zusätzlich an, welcher Teil der jeweiligen IP-Adresse für die Netzadresse verwendet wird und wo die Rechneradresse steht. Die Maske ist wieder eine Gruppe aus vier dezimalen Zahlen (z.B. 255.255.255.0). Würden Sie die Nummern der Maske in Form von einzelnen Bits darstellen und die Maske über Ihre IP-Adresse legen, dann würde eine *1* bedeuten, dass dieser Teil der IP-Adresse die Nummer des Netzwerks angibt, eine *0*, dass der zugehörige Teil der IP-Adresse zur Identifikation des Rechners im Netz benutzt wird. Die Maske kommt wieder über den Konfigurationsdienst oder wird lokal fest eingestellt.

Wenn nun jeder seine Adresse hat und die Bestandteile herausfinden kann, dann kann man beginnen, Daten auszutauschen.

Natürlich sind solche IP-Nummern umständlich zu merken und schwer zu benutzen. Deshalb gibt es spezielle Verwaltungsrechner im Netz, die Kataloge besitzen. Diese können für uns die Nummern herausfinden, wenn wir einen Namen kennen und natürlich auch umgekehrt. Der hier angebotene Dienst nennt sich *DNS* (domain name service / Namensdienst in einem Verwaltungsbezirk). Im Rahmen der Verhandlungen beim Anrufen eines Providers wird auch die Nummer dieses Rechners mit zu Ihnen übermittelt.

Jetzt bleibt nur noch eine Adresse zu besprechen, die wir bei der Benutzung des Internets wissen müssen. Wie der Name Inter-Net schon sagt, ist es nicht der Regelfall gewesen, dass einzelne Rechner über Telefonleitungen mit dem Internet verbunden werden, sondern dass ganze Organisationen ihre Netze verbunden haben. Ist nun ein Rechner z.B. Teil des Netzes einer Universität, dann wird dieser Rechner meist mit seinen Nachbar-Rechnern kommunizieren. Zur Verbindung mit anderen Netzen wird ein spezieller Rechner ausgesucht, der alle Verbindungen aus dem eigenen Netz oder hinein verwaltet. Er bildet das Tor oder die Schnittstelle zum Internet. Daher muss man in der Konfiguration des TCP/IP-Protokolls wissen, wer diese Aufgabe

übernimmt. Diesen Rechner nennt man das *Gateway*, was aber letztlich eben *Eingangstor des eigenen Netzes* bedeutet.

V-EXP01: Vergleich der Internet-Einstellungen

Suchen Sie einmal in Ihrem Rechner die Einstellungen für den Internetzugang und vergleichen Sie die besprochenen Begriffe mit den gewünschten Einstellungen. Schreiben Sie eine kurze Webseite, die Ihre Internet-Einstellungen beschreibt und fügen Sie ein Bild hinzu, das die Einstellungen wiedergibt. Nutzen Sie die Möglichkeiten Ihres Rechners, eine Bildschirmkopie zu machen.

Prüfen Sie auch die Webseite, die mit Hilfe des Editors verändert wurde. Sehen Sie sich den Quelltext an und vergleichen den Quelltext mit den bisher besprochenen Spielregeln.

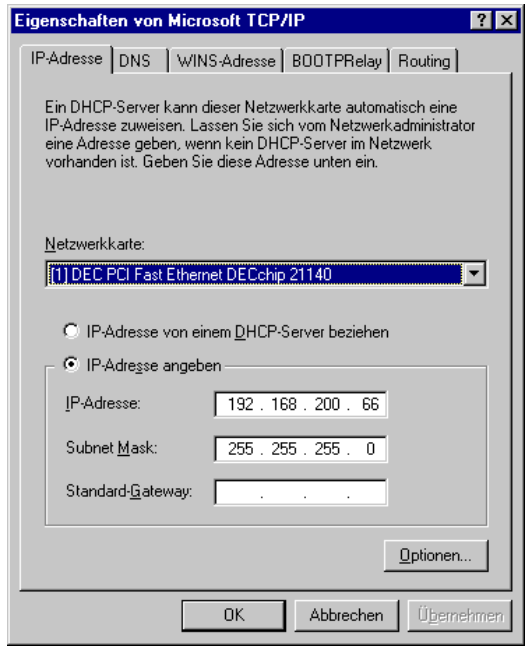

Abbildung 11.1: Netzwerkeinstellungen

Mit diesen kurzen Begriffserklärungen sollten wir genug über die Adressierungsverfahren im Internet wissen, um uns nun dem WWW zuwenden zu können.

11.2 Das WWW – World Wide Web

Man kann an den Begriff WWW aus einer technischen Sicht und aus einer konzeptionellen Sicht herangehen.

Die technische Sicht ist dabei einfacher darzustellen. Man kann sagen, dass das Internet die technische Grundlage und einen Satz von Übertragungsprotokollen liefert, mit denen dann Dienste angeboten werden können. Das WWW benutzt das Internet zur Datenübertragung und ein bestimmtes Protokoll zur Kommunikation und Abstimmung zwischen den Servern im Netz und dem eigenen Browser. Das Protokoll heißt *HTTP* (das Hypertext Transfer Protokoll).

Aus technischer Sicht ist damit erst mal alles gesagt.

Der Reiz des WWW kommt aber mehr aus dem Inhalt der Millionen von Webseiten in der ganzen Welt.

Tim Berners-Lee stellte in einem Vorschlag im Mai 1989 fest, dass die Forschungseinrichtung CERN sehr viel Wissen verliert. Nur ein Teil des Personals am CERN ist auf Dauer eingestellt. Viele Wissenschaftler arbeiten nur für einen relativ kurzen Zeitraum für ein bestimmtes Projekt am CERN. Und mit Ihnen verlässt auch wieder viel Wissen das Forschungszentrum. Zusätzlich müssen die nachfolgenden Mitarbeiter erneut in viele CERN-Spielregeln eingewiesen werden und sie müssen sich auch in die Arbeiten ihrer Vorgänger einlesen.

Er schlug ein nicht-hierarchisches System vor, das plattformübergreifend das Wissen konservieren sollte.

Dazu sei es notwendig, die darunter liegende Netzwerk-Technik zu verstecken und die Verweise zu Informationen völlig unstrukturiert zu erlauben. Im Grunde schlug er ein Hypertext-System vor, das Texte und Verweise beinhalten sollte.

Schreibt nun jeder Mitarbeiter seine Texte in der Hypertext-Form, kann er Verweise auf existierende Arbeiten einfügen. Und mit jeder neuen Arbeit wird das Geflecht der Verweise von einem Text auf einen anderen immer vielfältiger.

Damit haben wir aber den Kern des WWW beschrieben. Es ist das Netz der Verweise in den Webseiten, die das WWW formen. Es ist nicht die zugrunde liegende technische Realisierung.

Wenn Sie den Vorschlag von Tim Berners-Lee einmal selbst lesen wollen, finden Sie ihn beim WWW-Consortium unter *www.w3c.org* in der historischen HTML-Abteilung. Der Dateiname ist u.a. PROPOSAL .RTF.

Wenn wir über WWW Informationen und Verweise verlinken wollen, dann brauchen wir eine Möglichkeit, jede gewünschte Datei mit einer Adresse zu versehen und anzusprechen.

11.3 Sprunganweisungen in HTML

Beginnen wir mit einem einfachen Beispiel, um uns in die Adressierung auf dem WWW einzuarbeiten.

Ein Verweis (engl. link) wird mit der `<a>`-Anweisung beschrieben. Das *a* steht dabei für Anker (engl. anchor). Der Text, der zwischen der Start-Marke und der Ende-Marke der `<a>`-Anweisung geschrieben wird, erscheint in vielen Fällen im Browser-Fenster in einer hervorgehobenen Farbe und unterstrichen. In den Optionen des Browsers oder mit speziellen HTML-Anweisungen lässt sich das einstellen.

Wenn der Leser der Seite auf den so hervorgehobenen Text klickt, sollte der Browser dem Verweis folgen. Dazu brauchen wir noch einen Parameter *href*, mit dem man das Ziel des Verweises beschreiben kann.

Momentan soll der Verweis einfach der vollständige Name einer Datei sein.

Ein Verweis könnte wie folgt aussehen:

```
<a href="zweite.html">Mit Klick zur zweiten Datei</a>
```

11.4 Verweise und Farben

Verweise werden im Browser in drei verschiedenen Farben dargestellt. Es gibt Einstellungen für unbesuchte Verweise, besuchte Verweise und gerade angeklickte Verweise.

Die Farben können Sie mit Parametern in der `<body>`-Anweisung vorgeben. Die Namen der Parameter sind: *link* für unbesuchte Verweise, *vlink* für besuchte und *alink* für den gerade aktiv angeklickten Verweis. Die Werte sind wieder die schon verwendeten Farbwerte.

Beispiel:

```
<body link="#FF0000" vlink="#008000" alink="#0A0000">
```

V-EXP02/03: Verweise in HTML (1)

Schreiben Sie zwei HTML-Dateien mit einem beliebigen Inhalt. Jede der Dateien soll aber einen eigenen Inhalt haben, damit es sofort klar ist, welche der beiden Dateien gerade betrachtet wird. Gestalten Sie die beiden Seiten auch optisch unterschiedlich.

Fügen Sie in jede der Dateien einen Verweis auf die jeweils andere Datei ein.

Laden Sie eine Datei in den Browser und klicken Sie auf den Verweis. Es sollte die andere Datei erscheinen. Wenn der Verweis in der zweiten Datei wieder auf die erste verweist, kann man durch einfaches Anklicken die Dateien abwechseln betrachten.

Vielleicht schreiben Sie in jede Datei eine Strophe Ihres Lieblings-Gedichtes.

Verändern Sie in der zweiten Datei die Farben für Verweise.

Abbildung 11.2: Datei mit Verweis

Nun haben wir die Grundlage des WWW erarbeitet. Noch bewegen wir uns nur auf unserem eigenen PC. Da aber die Methode, Verweise zu anderen Seiten einzubauen immer gleich bleibt, gelten die bisherigen Spielregeln auch außerhalb des eigenen PCs. Unsere beiden Sei-

ten könnten also ohne jede Änderung auch einem der Webserver liegen, die die Dateien im WWW für uns bereitstellen. Und sie würden dort genauso funktionieren.

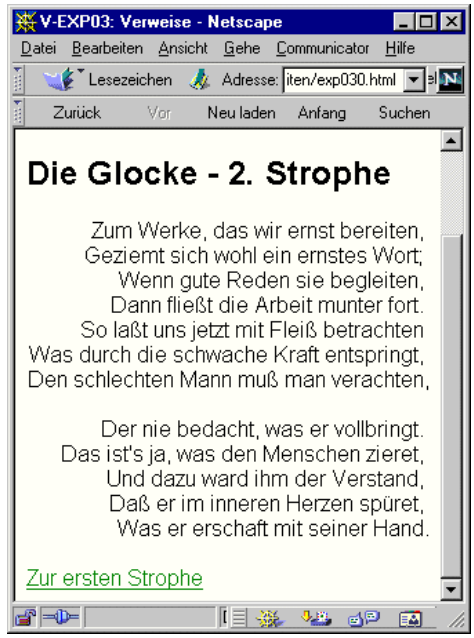

Abbildung 11.3: Rücksprung

Im nächsten Kapitel

Nachdem wir uns nun die Grundlagen des WWW erarbeitet haben, werden wir uns nun daran machen, die anderen Formen der Adressierungen im WWW zu betrachten.

Wir benötigen ja zum Aufbau des weltweiten WWW die Möglichkeit über das Netz beliebige Ressourcen auf einem beliebigen Rechner anzusprechen, der mit dem Netz verbunden ist und Ressourcen im WWW bereitstellt.

11.5 Lösungen

V-EXP01: Vergleich der Internet-Einstellungen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

```
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP01: Verweise 1</title>
</head>
<body>
<h1>
Verweise 1</h1>
TCP/IP ist das Paar von Internet-Protokollen, das wier f&uuml;r den Netzzu-
griff
en&ouml;tigen.
<h2>
TCP/IP Einstellungen</h2>
 <img src="tcp-ip.gif" border="0" height="460" width="371">
</body>
</html>
```

V-EXP02/03: Verweise in HTML (1)

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP02: Verweise</title>
</head>
<body>
<h1>
Verweise in HTML</h1>
Verweise zwischen Web-Seiten
<h2>
Die Glocke - 1. Strophe</h2>
Festgemauert in der Erden
<br>Steht die Form, aus Lehem gebrannt.
<br>Heute mu&szlig; die Glocke werden,
<br>Frisch, Gesellen, seid zur Hand.
<br>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;,
<br>Soll das Werk den Meister loben,
<br>Doch der Segen kommt von oben.
<p><a href="v-exp03.html">Zur zweiten Strophe</a>
</body>
</html>


<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
```

```
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP03: Verweise</title>
</head>
<body link="#FF0000" vlink="#008000" alink="#0A0000">
<h1>
Verweise</h1>
Springen mit den HTML-Seiten
<h2>
Die Glocke - 2. Strophe</h2>
<div align=right>Zum Werke, das wir ernst bereiten,
<br>Geziemt sich wohl ein ernstes Wort;
<br>Wenn gute Reden sie begleiten,
<br>Dann flie&szlig;t die Arbeit munter fort.
<br>So la&szlig;t uns jetzt mit Flei&szlig; betrachten
<br>Was durch die schwache Kraft entspringt,
<br>Den schlechten Mann mu&szlig; man verachten,
<br>Der nie bedacht, was er vollbringt.
<br>Das ist's ja, was den Menschen zieret,
<br>Und dazu ward ihm der Verstand,
<br>Da&szlig; er im inneren Herzen sp&uuml;ret,
<br>Was er erschaft mit seiner Hand.</div>
<a href="v-exp02.html">Zur ersten Strophe</a>
</body>
</html>
```

12 Adressierung im WWW

Die Aufgabe einer Adressierung im WWW ist es, alle freigegebenen Dateien auf den im WWW zusammengeschlossenen Rechner ansprechen zu können. Keine einfache Aufgabe. Die Lösung heißt *URL* (uniform resource locator / einheitliche Ortsangabe für Ressourcen). URLs gibt es in verschiedenen Formen.

Um genau zu sein, müssten wir eigentlich von sogenannten *URI* (uniform resource identifiers) sprechen. Es gab historisch eine Unterscheidung zwischen *URI* und *URL*. Sie wurden aber in einem Internet-Standard (RFC2396) zusammengefasst. Da im WWW-Umfeld der Name *URL* gebräuchlicher ist, bleiben wir auch hier dabei.

12.1 Relative Adressierung im WWW

Die einfachste Adressierung haben wir bereits kennen gelernt. Sie bestand einfach aus der Angabe eines Dateinamens. Der Browser kennt dabei die Adresse der momentan angezeigten Datei. Ein Dateiname ohne weitere Angaben in einem Verweis wird so verstanden, dass die Zieldatei im gleichen Verzeichnis wie die momentan angezeigte Datei liegt. Wenn man in der Adressierung vom Ort der angezeigten Datei ausgeht, spricht man von einer relativen Adressierung.

Will man als Ziel eine Datei außerhalb des momentanen Verzeichnisses angeben, kann man das Verzeichnis mit einem speziellen Verzeichnisnamen wechseln. Zwei einzelne Punkte bedeuten: „gehe zum übergeordneten Verzeichnis“. Die Zieladresse *../ziel.html* bedeutet, dass die gewünschte Datei ZIEL.HTML im übergeordneten Verzeichnis liegt. Als Trennzeichen zwischen den einzelnen Elementen des Suchpfades wird immer der vorwärts gerichtete Schrägstrich verwendet. Das verstehen alle Rechner.

Natürlich kann man die Angabe „gehe ins übergeordnete Verzeichnis“ auch wiederholen, wenn die Datei weiter weg liegt.

Machen wir noch ein kleines Beispiel. Was bedeutet *../bilder/auf.gif*? Zuerst gehen wir ein Verzeichnis nach oben, wechseln dann in das Verzeichnis BILDER und zeigen auf die Datei AUF.GIF. Die Datei liegt also in einem Verzeichnis neben dem momentanen. Wenn es bei der relativen Adressierung komplizierter wird, dann hilft die Darstellung des Verzeichnisbaums.

V-EXP04: relative Adresse von Dateien (Seiten, Ressourcen)

Legen Sie neben Ihrem Arbeitsverzeichnis ein weiteres Verzeichnis für Bilder an.

Sie sollten im Datei-Explorer das Arbeitsverzeichnis und das Bilder-Verzeichnis im gleichen übergeordneten Verzeichnis sehen können. Speichern Sie nun eine Bilddatei in das neue Bilder-Verzeichnis. Statt von einem Verzeichnis spricht man auch von einem Ordner oder (engl.) von einem Directory.

Schreiben Sie nun eine HTML-Datei und speichern diese im Arbeitsverzeichnis. In der HTML-Datei soll ein Link auf das Bild im Nachbarverzeichnis zeigen.

Beim Klicken auf den Verweis wird im Browser das Bild dargestellt.

Abbildung 12.1: Bild mit relativer Adresse

Wenn wir eine Gruppe von Dateien erstellen, die einmal im WWW publiziert werden sollen, dann sollten alle Verweise zwischen den Dateien immer mit relativen Pfadangaben erfolgen. Eine solche Gruppe von Dateien können Sie lokal erstellen und danach gemeinsam auf einen Server im Web stellen. Und alle Verweise zwischen den Seiten funktionieren nach wie vor.

12.2 Absolute Adressierung im WWW

Will man auf eine beliebige Ressource (Datei) zugreifen, gibt man meist drei grundlegende Informationen an:

- die Zugriffsmethode,
- den Server und evtl. seinen Port
- Pfad und Dateinamen auf dem Server.

Ein Beispiel ist *http://www.walter-herglotz.de/index.html*.

Die Zugriffsmethode für HTML-Dateien ist das Transportprotokoll *http*, das wir schon kurz kennen gelernt haben. Mit Hilfe dieses Protokolls kommunizieren Browser und der Webserver. Daher nennt man auch den Webserver gelegentlich *http*-Server oder unter Linux *http*-Dämon. Und so heißt die Programmdatei des unter Linux meist benutzten Webservers *Apache* eben *httpd* (http-Dämon). Unter Linux gibt es gute Dämonen, die im Hintergrund ihre Dienste anbieten.

Ein weiterer Dienst, den Ihr Browser mit hoher Wahrscheinlichkeit nutzen kann, ist *FTP* (file transfer protocol / Dateiübertragungs-Protokoll). Beginnt eine Adresse mit *ftp://*, dann soll die Datei, auf die der Verweis zeigt, mit einem anderen Protokoll übertragen werden. Dazu ist natürlich auf der Seite des Servers ein weiteres Protokoll-Server-Programm (der FTP-Server oder FTP-Dämon) notwendig.

Diese Dienste oder Protokoll-Server laufen alle auf dem gleichen Rechner. Es muss also eine Möglichkeit geben, das richtige Programm mit dem richtigen Protokoll zu erreichen. Man hat dazu sogenannte Ports (Dienst-Nummern) eingerichtet. Jeder Dienst hat dabei eine standardisierte Port-Nummer. Für *FTP* ist die Nummer *21*, für *HTTP* ist es die *80*. Spricht man einen bestimmten Server mit einer definierten Port-Nummer an, dann wartet auf dem Server meist das richtige Programm auf Daten, sofern der entsprechende Protokoll-Server (oder auch Dienst genannt) gestartet wurde. Wenn der Zielserver mit der normalen Port-Nummer angesprochen werden soll, dann kann diese Nummer entfallen. Will man die Nummer angeben, wird sie nach einem Doppelpunkt in dezimaler Schreibweise an den Servernamen angefügt.

Beispiel:

http://www.walter-herglotz.de:80/index.html

12.3 Domänen

Den Server findet man mit Hilfe der so genannten Domänen-Adressierung. Eine Domäne (engl. domain) ist ein Verwaltungsbezirk. Die Welt wird stufenweise in immer feinere Verwaltungsbezirke eingeteilt. In der ersten Stufe, der groben Unterteilung, gibt es zwei verschiedene Arten von Verwaltungsbezirken. Die ursprüngliche Einteilung der amerikanischen Entwickler geschah nach Arten von Netzbetreibern.

So gibt es Netze

- der Regierung (gov / government)
- von Universitäten und anderen Bildungsstätten (edu / education)
- von nicht-kommerziellen Organisationen (org / organisation)
- von Firmen (com / companies)
- von Netzbetreibern (net / networks)

In den meisten anderen Ländern werden internationale Länderkürzel verwendet. Für Deutschland steht *de*, für Frankreich *fr*. Und so weiter.

Diese Abkürzungen bilden die erste Stufe der Unterteilungen des weltweiten Netzes. Der zumeist verwendete Begriff ist auch *first level domain* (erste Stufe der Einteilung in Verwaltungsbezirke).

In der zweiten Stufe wird dann die jeweilige Domäne wiederum unterteilt. Die zweite Stufe bilden oft Firmen, Städte oder Organisationen. Schließlich kann man dann innerhalb von der zweiten Stufe weitere Unterteilungen in einer dritten und vierten usw. Stufe vornehmen.

Und schließlich brauchen wir einen Namen für einen Rechner. Webserver heißen oft *www*. Aber jeder andere Name wäre auch recht.

Einen bestimmten Rechner findet man nun irgendwo auf der Welt, indem man hinter seinen Namen die Liste der zuständigen Verwaltungsbezirke vom niedrigsten bis zur ersten Stufe (der first level domain) jeweils mit einem Punkt getrennt anfügt.

Eine einfache Rechner-Adresse ist dann z.B. *http://www.cnn.com*.

Auf dem Server wird für den HTTP-Server ein Verzeichnis im Verzeichnisbaum reserviert. Ab diesem Verzeichnis steht Platz für die Da-

teien zur Verfügung, die man im Internet sehen kann. Das Start-Verzeichnis nennt man die Wurzel und sie wird durch einen Schrägstrich am Ende des Rechnernamens benannt.

Ein Beispiel: *http://www.intel.com/*. In diesem Verzeichnis können dann Dateien und wiederum Verzeichnisse liegen.

Die weitere Adressierung geschieht wie auf den üblichen PCs auch. Wie erwähnt verwendet man im WWW nur vorwärts gerichtete Schrägstriche zur Trennung von Verzeichnis- und Dateinamen.

12.4 Standard-Annahmen in der Adressierung

In der Adressierung gibt es eine Reihe von Standard-Einstellungen. Zuerst kann bei der Adress-Eingabe im Browser die Protokoll-Angabe entfallen. Der Browser ergänzt das Protokoll automatisch.

Oft wird auch die Angabe einer Datei fehlen. Jeder Server hat eine Standard-Datei, die er anzeigt, wenn keine spezielle Datei gewünscht wird. Der Name ist oft INDEX.HTML (kleingeschrieben) auf Unix/Linux-Rechnern oder DEFAULT.HTM auf Windows-Rechnern.

Jeder, der im WWW einen eigenen Domänennamen betreibt, muss die passenden Start-Datei bereit stellen.

Und schließlich kann man in der Domänenverwaltung festlegen, dass auch der Rechnername fehlen kann. Dann wird innerhalb der Domäne ein Standard-Rechner angesprochen.

Unser kürzestes Beispiel zur Eingabe im Browser ist damit *cnn.com*.

V-EXP05: Absoluter URL

Schreiben Sie eine HTML-Datei, die einen Verweis auf eine beliebige andere Datei im WWW beinhaltet. Benutzen Sie die volle Adressierung ohne Annahme von Standardvorgaben (incl. der Port-Angabe für HTTP).

Falls Sie keine besondere Vorliebe für eine spezielle Datei haben, können Sie die Startseite des Webservers des Autors benutzen. (*http://www.walter-herglotz.de/index.html*).

Natürlich kann die Aufgabe nur funktionieren, wenn Sie mit dem Internet verbunden sind.

Abbildung 12.2: Absolute Adressierung im WWW

12.5 Namen und IP-Nummern verwalten

Wie wir bereits im vorhergegangenen Kapitel gesehen haben, muss jeder Rechner unter einer eindeutigen Rechnernummer angesprochen werden. In diesem Kapitel haben wir über den Zugriff mit Domänen-Namen gesprochen. Es muss nun noch Instanzen geben, die Rechner und Domänen-Namen auf IP-Adressen umsetzen.

Dazu gibt es einen zentralen Rechner für das ganze Internet am *NIC* (network information center / Netzwerk Informations-Zentrum in den USA) und jeweils einen Rechner innerhalb jeder dieser Domänen. Er ist jeweils die Verwaltungsinstanz für Domänennamen oder auch Rechnernamen in den unteren Verwaltungsbezirken.

In Deutschland gibt es beispielsweise das *DENIC* (Network information center for DE / Netzwerk Informationszentrum für den Bereich DE). Hier gibt es einen Katalog aller Verwaltungsbezirke innerhalb des DE-Bezirks. Dieser Katalog ist die Grundlage für den deutschen *DNS*-Service (der domain name service für DE).

Nehmen wir einmal ein extremes Beispiel an. Sie fordern mit Hilfe Ihres Browsers die beschriebene Startseite vom Webrechner aus der Domäne des Autors an. Im Grunde würde nun der DNS-Dienst des Pro-

viders in den USA nachfragen, welcher Rechner für den Verwaltungsbezirk *de* zuständig ist. Danach würde eine Anfrage an den DENIC geschickt, wer den Bezirk *walter-herglotz* verwaltet und schließlich würde dann die nächste Anfrage an den DNS-Dienst von *walter-herglotz.de* gehen und um die IP-Adresse des Rechners *www* bitten.

Die Auflösung der Namen funktioniert im Prinzip tatsächlich so, wenn auch Zwischenspeicher die allermeisten Anfragen puffern. Man muss also nicht bei jedem Zugriff einmal in den USA anfragen. Das wäre das Ende des Internets.

Diese Verwaltung erspart dem Benutzer eine Menge Tipparbeit, da die Namensauflösungen automatisch erfolgen. Vielleicht kennen Sie als Gegenstück die X.400-Mail-Adressierung, wo jede Detailinformation des Empfängers in der Adresse angegeben werden musste. Meist haben die Visitenkarten gar nicht ausgereicht, um die Adresse unterzubringen.

12.6 Verweise innerhalb einer Datei

Auch innerhalb einer (genügend großen) Datei kann man springen. Setzt man an den Anfang ein Inhaltsverzeichnis, kann man den Beginn jedes Kapitels per Mausklick auf einen Verweis erreichen.

Allerdings müssen wir hier noch Sprungmarken einführen. Bisher hatten wir immer Dateinamen, um einen Verweis auszuführen. Diese Dateinamen wurden entweder durch relative oder absolute URLs ausgedrückt.

Für einen Sprung innerhalb einer Datei müssen wir zuerst den möglichen Zielpunkten Namen geben.

Dazu dient ebenfalls die `<a>`-Anweisung, die dann allerdings einen Namens-Parameter benutzt.

Der Parameter *name* definiert einen Namen für die Sprungmarke.

Beispiel:

```
<a name="kapitel_4">Beginn des Kapitels 4</a>
```

Der Inhaltstext wird meist entfallen. Bei den Namen für Sprungmarken muss man auf Groß- und Kleinschreibung achten.

Benutzen Sie für Anweisungen, Dateinamen und Sprungmarkennamen ausschließlich kleine Buchstaben.

Im Inhaltsverzeichnis kann man dann zur gerade definierten Marke mit Hilfe des üblichen Verweises springen. Wir müssen dabei jedoch angeben, dass es sich um einen lokalen Namen anstelle des bisher benutzten Dateinamens handelt.

Beispiel:

```
<a href="#kapitel_4"> Kapitel 4: Über Sprünge in der Seite</a>
```

V-EXP06: Springen innerhalb einer Datei

Schreiben Sie eine HTML-Datei, die einen längeren Text enthält. Natürlich können auch Bilder vorkommen. Die Datei sollte mindestens so groß sein, dass sie nicht auf einmal auf den Bildschirm passt.

Fügen Sie kurz vor Ende der Datei eine Sprungmarke ein. Und schreiben Sie dann noch am Anfang der Datei einen Verweis auf das Sprungziel.

Falls Sie bereits umfangreichere Dateien besitzen, könnten Sie auf diese Art ein komplettes Inhaltsverzeichnis an den Anfang stellen.

Noch etwas: es ist übrigens ein guter Schreibstil, nach jedem größeren Absatz einen Rücksprung an den Anfang der Seite vorzusehen. So geben Sie dem Benutzer jederzeit eine Möglichkeit zu navigieren.

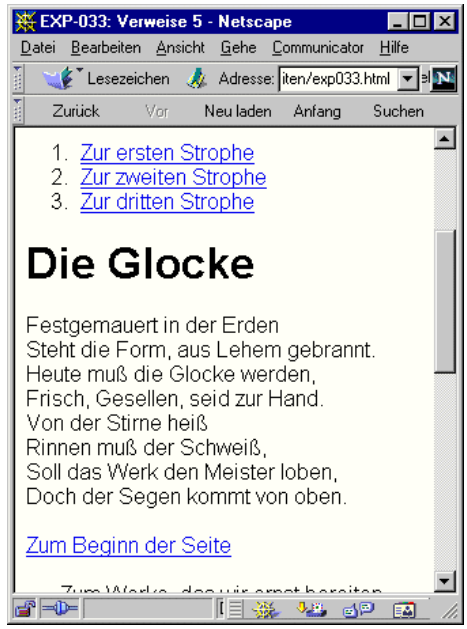

Abbildung 12.3: Springen innerhalb der Seite

Damit haben wir nun alle Adressierungsmöglichkeiten eines URLs zusammen.

Wir können nunmehr eine Transportmethode auswählen (bisher http oder ftp), einen beliebigen HTTP-Server in der ganzen Welt ansprechen, dort in einem Unterverzeichnis eine bestimmte Datei anfordern und schließlich dem Browser bitten, die Datei an einer ganz bestimmten Stelle anzuzeigen.

Die vollständige Syntax dafür ist:

Protokoll://Server:Port/Teil_Pfad/Dateiname.Kennung#Sprungmarke

12.7 Mailadressierung

Nehmen wir einmal an, Sie möchten eine größere Arbeit im Internet veröffentlichen. Damit ein Leser Ihnen Rückmeldungen geben kann, soll Ihre E-Mail-Adresse mit bekanntgegeben werden. Im einfachsten Fall schreiben Sie die E-Mail-Adresse als Text auf die Startseite Ihrer Arbeit.

Es wäre bestimmt schöner, wenn man dem Leser die Rückmeldung erleichtern könnte.

Dazu gibt es eine spezielle Methode: *mailto*.

Man kann in einer HTML-Datei einen Verweis einfügen, der als *href*-Wert das Protokoll *mailto* verwendet und nach einem Doppelpunkt einfach Ihre eigene Mailadresse angibt.

Hier würde der Browser versuchen, das Mailprogramm zu starten, das auf dem Rechner des Lesers installiert ist. Beim Start würde dann das Mailprogramm gleich den Adressaten als Empfänger erhalten. Tippfehler sind dann bei der Adresse nicht mehr möglich. Auch die *Betreff*-Zeile könnte gleich mit ausgefüllt werden.

Der eigentlichen Adresse kann man nach einem Fragezeichen weitere Informationen mitgeben, die man sonst auch beim Schreiben einer E-Mail ausfüllen könnte. Beispiele sind hier *cc* für einen weiteren Empfänger und *subject* für den Betreff. Das ist nicht standardisiert, sollte aber funktionieren.

Beispiel 1:

```
<a href="mailto:egon.meier@musterdomain.de">Schick mir eine Mail</a>
```

Beispiel 2:

```
<a href="mailto:egon.meier@musterdomain.de? cc=peter.huber@xyz.com">Mail
an 2 </a>
```

Beispiel 3:

```
<a href="mailto:emaier@domain.de?subject=Nette Rückmeldung">Mail mit
Betreff</a>
```

Beispiel 4:

```
<a href="mailto:emaier@mdomain.de?body=Text der Nachricht">Mail mit
Inhalt</a>
```

Die Angaben können auch kombiniert werden. Dazu schließen Sie die nächste Vorgabe wiederum mit einem *?* an.

V-EXP07: Angabe der eigenen Mailadresse

Verändern Sie eine vorhandene HTML-Datei und ergänzen Sie die Datei um einen Verweis auf die eigene Mailadresse und experimentieren Sie auch mit den beschriebenen Beispielen.

Betrachten Sie die Datei im Browser und probieren die vier Beispiele aus. Startet jeweils Ihr Mailprogramm mit den richtigen Vorgaben?

Abbildung 12.4: Webseite mit Mail-Verweisen

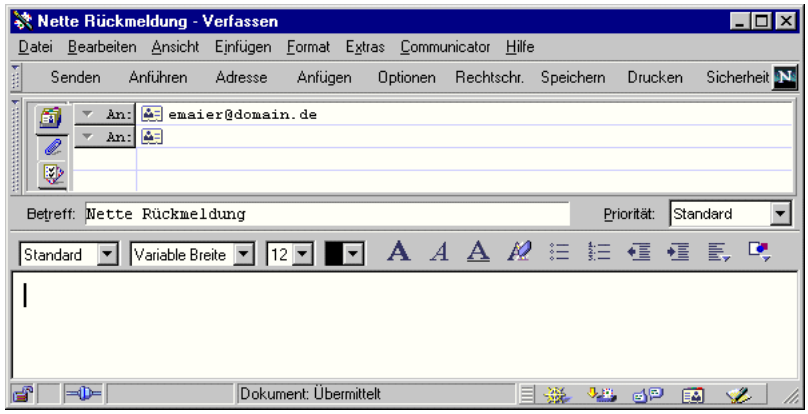

Abbildung 12.5: Gestartetes Mail-Programm mit Vorgabe

Im nächsten Kapitel

Unsere bisherigen Verweise arbeiteten alle mit textlichen Informationen. Der Besucher unserer Webseite muss dazu den zugehörigen Text lesen, um den passenden Link auszuwählen. In vielen Fällen wäre auch eine graphische Unterstützung für Sprünge sehr sinnvoll.

In HTML gibt es eine sehr gelungene Kombination aus Bildern und Verweisen. Man kann aus einem Bild heraus auf viele verschiedene Ziele deuten. Anders als bei einem Textverweis, der nur zu einem Ziel verweisen kann, kann man in HTML die Fläche des Bildes benutzen, um bestimmten Flächen bestimmte Ziele zuzuordnen.

Damit lassen sich viele Themen sehr leicht darstellen und bedienbar machen. Das kann ein Familienphoto sein oder auch eine Explosionszeichnung in einer technischen Dokumentation.

12.8 Lösungen

V-EXP04: relative Adresse von Dateien (Seiten, Ressourcen)

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP04: Verweise</title>
</head>
<body>
<h1>
```

```
HTML-Verweise</h1>
Relativer Zugriff auf eine Datei/Seite
<br>Bildanzeige mit relativem Pfad<b></b>
<p><b>(../exp-bilder/normal.gif)</b><b></b>
<center>
<p><a href="../exp-bilder/normal.gif">Bild anzeigen</a></center>
</body>
</html>
```

V-EXP05: Absoluter URL

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP05: Verweise</title>
</head>
<body>
<h1>
HTML-Verweise</h1>
Absoluter Zugriff auf eine Datei/Seite
<h2>
Aufruf einer Internet-Seite</h2>
<center><a href="http://www.walter-herglotz.de:80/index.html">Start-Seite
eines Web-Auftritts anzeigen</a><a href="http://www.walter-herglotz.de:80/
index.html"></a>
<p>http://www.walter-herglotz.de:80/index.html
<br>(Der Port 80 kann entfallen, ebenso die Dateiangabe.)</center>
</body>
</html>
```

V-EXP06: Springen innerhalb einer Datei

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP06: Verweise</title>
</head>
<body>
<a NAME="anfang"></a>
<h1>
HTML-Verweise und Sprungmarken</h1>
Springen innerhalb einer Datei mit Sprungmarken und lokalen Verweisen
```

```
<h2>
Sprungmarken</h2>
<ol>
<li>
<a href="#strophe01">Zur ersten Strophe</a></li>
<li>
<a href="#strophe02">Zur zweiten Strophe</a></li>
<li>
<a href="#strophe03">Zur dritten Strophe</a></li>
</ol>
<h1>
Die Glocke</h1>
Festgemauert in der Erden
<br><a NAME="strophe01"></a>Steht die Form, aus Lehem gebrannt.
<br>Heute mu&szlig; die Glocke werden,
<br>Frisch, Gesellen, seid zur Hand.
<br>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;,,
<br>Soll das Werk den Meister loben,
<br>Doch der Segen kommt von oben.
<p><a href="#anfang">Zum Beginn der Seite</a>
<center>
<p><a NAME="strophe02"></a>Zum Werke, das wir ernst bereiten,
<br>Geziemt sich wohl ein ernstes Wort;
<br>Wenn gute Reden sie begleiten,
<br>Dann flie&szlig;t die Arbeit munter fort.
<br>So la&szlig;t uns jetzt mit Flei&szlig; betrachten
<br>Was durch die schwache Kraft entspringt,
<br>Den schlechten Mann mu&szlig; man verachten,
<br>Der nie bedacht, was er vollbringt.
<br>Das ist's ja, was den Menschen zieret,
<br>Und dazu ward ihm der Verstand,
<br>Da&szlig; er im inneren Herzen sp&uuml;ret,
<br>Was er erschaft mit seiner Hand.</center>
<p><a href="#anfang">Zum Beginn der Seite</a>
<div align=right>
<p><a NAME="strophe03"></a>Nehmet Holz vom Fichtenstamme,
<br>Doch recht trocken la&szlig;t es sein,
<br>Da&szlig; die eingepre&szlig;te Flamme
<br>Schlage zu dem Schwalch hinein.
<br>Kocht des Kupfers Brei,
<br>Schnell das Zinn herbei,
<br>Da&szlig; die z&auml;he Glockenspeise
<br>Flie&szlig;e nach der rechten Weise.</div>
<p><a href="#anfang">Zum Beginn der Seite</a>
</body>
</html>
```

V-EXP07: Angabe der eigenen Mailadresse

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>V-EXP07: Verweise</title>
</head>
<body>
<h1>
HTML-Verweise</h1>
Senden einer eMail
<h2>
Aufruf des eMail-Programms</h2>
<center><a href="mailto:egon.meier@musterdomain.de">Schick mir eine Mail</a>
<br><a href="mailto:egon.meier@musterdo-
main.de?cc=peter.huber@xyz.com">Mail
an 2</a>
<br><a href="mailto:emaier@domain.de?subject=Nette Rückmeldung">Mail mit
Betreff</a>
<br><a href="mailto:emaier@mdomain.de?body=Text der Nachricht">Mail mit
Inhalt</a><a href="mailto:emaier@mdomain.de?body=Text der Nachricht"></a>
<p>z.B. mailto:egon.meier@musterdomain.de</center>
</body>
</html>
```

13 Sensitive Bilder

Haben Sie irgendwo ein Familienphoto, auf dem anlässlich einer wichtigen Feier fast alle aus der Familie zusammengekommen sind?

Wenn jetzt jeder noch eine eigene HTML-Seite irgendwo auf einem Webserver hätte, dann könnte man doch von jedem Mitglied der Familie ein wenig mehr lernen. Nur müsste man irgendwie von der graphischen Information hin zu den einzelnen Webseiten finden.

HTML bietet dazu die Möglichkeit der sensitiven Bilder. Der Betrachter fährt mit der Maus über eine beliebige Stelle und klickt darauf. Damit markiert man den gewünschten Punkt. Die Stellung der Maus liefert dazu die X- und Y-Koordinaten relativ zum Bild. Mit dieser Information kann man nun eine bestimmte Webseite auswählen oder andere Aktionen einleiten.

Man kann entweder das ganze Bild sensitiv machen oder über das Bild eine Art Folie legen, auf der die sensitiven Flächen markiert sind. Die erste Variante benötigt die Unterstützung des Webservers, die zweite kann im Browser abgearbeitet werden. Wenden wir uns daher erst einmal dem zweiten Fall zu.

13.1 Bilder zum Anklicken

Die grundlegende Version eines anklickbaren Bildes erhalten wir, wenn wir ein Bild alleine oder zusätzlich zu einem Text in den Inhalt der `<a>`-Anweisung stellen.

D-EXP01: Bilder als Anklickbereich

Schreiben Sie eine HTML-Datei, die ein Bild als Inhalt einer `<a>`-Anweisung enthält. Klicken Sie im Browser auf das Bild und springen Sie damit zu der Zieldatei.

Fügen Sie zusätzlich zum Bild noch Text ein.

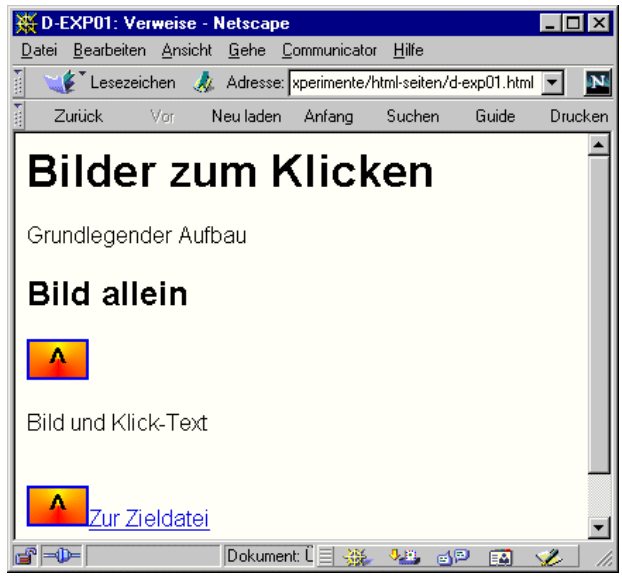

Abbildung 13.1: Anklickbare Bilder

13.2 Sensitive Bilder mit Auswertung durch den Browser

Von einem beliebigen Bild benötigen wir die Größe in Pixel. Das Dateiformat ist dabei unerheblich. Laden Sie das Bild in Ihr Graphik-Bearbeitungsprogramm. Dort gibt es üblicherweise eine Informationsfunktion, die die Eigenschaften des Bildes detailliert berichtet.

Um die sensitiven Bereiche zu definieren, brauchen wir eine entsprechende Tabelle, die sowohl die sensitiven Bereiche definiert und die gewünschten URLs zuordnet. Es gibt unterschiedliche Formen für die sensitiven Bereiche. Die Formen sind Kreise, Rechtecke und Polygone. Damit sollten sich alle Kopfformen der lieben Verwandten darstellen lassen.

Die Zuordnungstabelle heißt auf englisch *map* und so heißt auch die HTML-Anweisung. Die Zuordnungstabelle wird mit der `<map>`-Anweisung beschrieben. Start- und Ende-Marke sind notwendig. Als Parameter benötigen wir unbedingt eine Namensangabe mit *name*. Wie alle in einer Datei definierten Namen ist auch der Name der Zuordnungstabelle empfindlich auf Groß- und Kleinschreibung. Eine solche Zuordnungstabelle kann man sich als eine transparente Folie vor-

stellen, auf der die verschiedenen Bereiche mit einem Filzstift aufgezeichnet wurden.

Den Namen der Zuordnungstabelle werden wir dann später beim Bild angeben, um – bildlich gesprochen – die Zuordnungsfolie über das Bild zu legen. Die gleiche Zuordnungstabelle könnte durchaus mit verschiedenen Bildern genutzt werden.

Der Ausgangspunkt des Koordinatensystems ist die linke obere Ecke. Die X-Werte wachsen nach rechts, die Y-Werte wachsen nach unten. Es gibt nur positive Werte für die Koordinaten.

Ein Rechteck wird mit zwei Punkten definiert: der linken oberen Ecke und der rechten unteren. Jeder Punkt kann mit einem X- und Y-Wert beschrieben werden. Wir brauchen also vier Koordinatenwerte für ein Rechteck.

Die jeweiligen Koordinaten werden hier im Beispiel durch den Zusatz *lo* für links oben und *ru* für rechts unten näher gekennzeichnet. Im wirklichen Eintrag stehen dann natürlich die passenden Zahlenwerte. Der Eintrag in der Zuordnung (engl. map) ist:

```
<area shape="rect" coords="xlo, ylo, xru, yru" href="URL 1">
```

Einen Kreis kann man mit dem Mittelpunkt und dem Radius des Kreises beschreiben.

```
<area shape="circle" coords="xmp, ymp, r" href="URL 2">
```

Die dritte geometrische Figur ist das Polygon. Es werden durch eine beliebig lange Kette von Koordinatenangaben die Punkte im Polygon beschrieben. Die Umrandung finden wir, indem wir uns vom Startpunkt aus zu dem jeweils nächsten Punkt eine Linie denken und vom letzten angegebenen Punkt die Linie zum Anfangspunkt weiterführen. Damit schließen wir gedanklich die Form des Polygons.

```
<area shape="poly" coords="x1, y1, x2, y2, x3, y3" href="URL 32>
```

Gängige Browser verstehen auch die ausgeschriebenen Namen für die Formen. Verwenden sollten Sie diese aber nicht.

Was wir gerade besprechen nennt man *CSIM* (client side image maps / Bildzuordnungen auf dem Browser). Neben diesen Zuordnungen kann man auch Zuordnungen mit Hilfe eines Programmes auf dem Server machen. Doch dazu später mehr.

Die Eigenschaften der Zuordnungen im Browser sind:

- einfach mit HTML zu gestalten
- es wird nur der Browser benötigt, um die Verzweigung zu machen
- der Browser kann als Vorausschau die angebotenen Verweise in der Statuszeile zeigen
- die Methode eignet sich eher für gröbere Auflösungen, wenn der Cursor nicht exakt positioniert sein muss.

D-EXP02: Arbeiten mit sensitiven Bildern und der Browserauflösung

Schreiben Sie eine HTML-Datei, in der ein beliebiges Bild bekannter Größe geladen wird.

Definieren Sie in einer Zuordnungstabelle einige sensitive Stellen. Nutzen Sie jede der drei Möglichkeiten für die Form mindestens einmal.

Die Koordinaten der benötigten Punkte können Sie auf verschiedene Arten ermitteln:

- eventuell durch Berechnung (wenn das ganze Bild 400 Pixel breit ist, dann sind 10% davon...)
- mit einem Graphikprogramm, das die Koordinaten anzeigt
- einen „großen“ HTML-Editor
- indem Sie einen Trick verwenden (der gleich beschrieben wird).

Ergänzen Sie in der `<img>`-Anweisung den Parameter *usemap*, der als Wert den Namen der Zuordnungstabelle erhält. Für Zuordnungstabellen, die in der gleichen Datei liegen, müssen Sie dem Tabellennamen wie bei allen lokalen Namen ein # voranstellen.

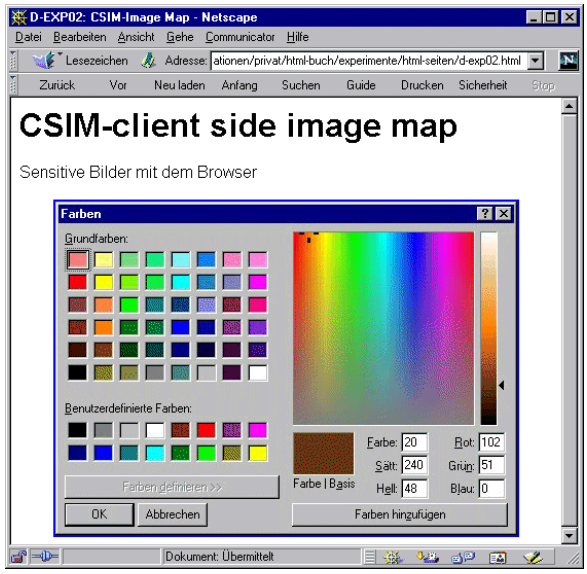

Abbildung 13.2: Bildzuordnungen im Browser

13.3 Trick zur Ermittlung der Koordinaten

Bei der Ermittlung der Koordinaten kann ein kleiner Trick helfen.

Ändern wir für die Zeit der Erstellung der Zuordnungstabelle die `<img>`-Anweisung ein wenig. Zuerst sollte sie von einer `<a>`-Anweisung eingerahmt werden. Als Sprungziel für den *href*-Parameter kann man einen Phantasienamen (XXX.HTML) angeben.

Und als zweites benutzen wir anstelle des *usemap*-Parameters mit dem Namen der Zuordnungstabelle als Wert nur den Parameter *ismap* ohne weitere Angaben.

Der Trick bedeutet, dass wir zur Ermittlung der Koordinaten aus einem sensitiven Bild mit Auflösung durch den Browser nun ein sensitives Bild mit Auflösung durch den Server gemacht haben. Wir haben damit zwar dem Buch etwas vorgegriffen, aber sehen dafür nun in der Statuszeile die Koordinaten angezeigt. Fahren Sie mit der Maus über das Bild, können Sie die jeweils zugehörigen Koordinaten einfach ablesen und in die Tabelle eintragen. Praktisch – nicht wahr?

Haben wir die Zuordnungstabelle vollständig aufgebaut, können wir die Änderungen wieder rückgängig machen, also wieder *usemap* mit der Angabe der Zuordnungstabelle einfügen und den Verweis entfernen.

Nun sollte es klappen. In der Statuszeile sollte nun das jeweilige Sprungziel angezeigt werden, wenn wir über eine sensitive Stelle fahren. (siehe auch das Experiment zu den serverseitigen sensitiven Bildern.)

Abbildung 13.3: Koordinatenangaben in der Statuszeile

Im nächsten Kapitel

Die vielen Elemente der HTML-Sprache müssen zu einer ganzen Seite mit einem ansprechenden Aussehen zusammengestellt werden. Surfen Sie am besten einmal durch das Netz und besuchen verschiedene Webseiten. Vergleichen Sie vielleicht nach Branchen. Firmen, die in einer Geschäftswelt zuhause sind, die stark auf Prestige achtet, werden aufwendige gestaltete Seiten mit Animationen und graphischer Gestaltung anbieten.

Auf der anderen Seite der Skala der Möglichkeiten stehen dann oft Veröffentlichungen von Universitäten, bei denen der Inhalt im Vordergrund steht.

Einen Mittelweg im Layout bietet HTML mit seinen Tabellen an. Und die werden wir nun im Detail untersuchen.

13.4 Lösungen

D-EXP01: Bilder als Anklickbereich

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>D-EXP01: Verweise</title>
```

```
</head>
<body>
<h1>
Bilder zum Klicken</h1>
Grundlegender Aufbau
<h2>
Bild allein</h2>
<a href="g-exp08.html"><img SRC="auf.gif" NOSAVE height=25 width=40></a>
<p>Bild und Klick-Text
<br> 
<p><a href="c-exp01.html"><img SRC="auf.gif" NOSAVE height=25 width=40></
a><a href="c-exp01.html">Zur
Zieldatei</a>
<p>
<hr WIDTH="100%">
</body>
</html>
```

D-EXP02: Arbeiten mit sensitiven Bildern und der Browserauflösung

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>D-EXP02: CSIM-Image Map</title>
</head>
<body>
<h1>
CSIM-client side image map</h1>
Sensitive Bilder mit dem Browser
<center>
<p><map name="csim1"><area shape="rect" coords="15, 50, 33, 65" href="l-
exp01.html"><area shape="circle" coords="262,247,20" href="l-
exp02.html"><area shape="poly" coords="231,34,406,34,405,213,236,217"
href="l-exp03.html"></map><img SRC="farbsucher.jpg" NOSAVE usemap="#csim1"
height=324 width=449></center>
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

14 Tabellen

Tabellen spielen in HTML eine wichtige Rolle. Man kann sie natürlich wie gewohnt als wirkliche Tabelle einsetzen, um Übersichten zu gestalten. Aber Tabellen haben in HTML noch die weit öfter genutzte Funktion, ein Seitenlayout festzulegen. Aber fangen wir zuerst mit der bekannten Tabelle an.

14.1 Tabellen zur Informationsdarstellung

Eine Tabelle ist eine rechteckige Sammlung von Containern für weitere Inhalte. Die Container sind die Zellen, die am Schnittpunkt von Zeilen und Spalten liegen.

Beginnen wir mit der Grundversion der Tabelle. Die `<table>`-Anweisung umrahmt die Ansammlung von Zeilen und Spalten. Eine Zeile wird dabei durch die `<tr>`-Anweisung beschrieben. Die Spalten ergeben sich implizit durch die Anzahl der in den Zeilen enthaltenen Datenelemente, die mit der `<td>`-Anweisung beschrieben werden. Innerhalb der `<td>`-Anweisungen stehen dann die eigentlichen Inhalte. Sollte einmal eine Zelle momentan noch leer bleiben, dann kann ein Platzhalter eingesetzt werden. Der Platzhalter ist * *. Es gibt also erst einmal keine explizite Angabe für die Anzahl der Spalten.

Der Browser muss also zuerst die gesamte Tabelle einlesen, um das wirkliche Erscheinungsbild festlegen zu können.

Alle drei bisher erwähnten Befehle `<table>`, `<tr>` und `<td>` arbeiten wie gewohnt mit einer Start- und einer Ende-Marke.

K-EXP01: Einfache Preistabelle

Schreiben Sie eine einfache Tabelle mit drei Spalten. In der ersten Spalte sollen einige Güter des täglichen Bedarfs stehen, in der zweiten Spalte die übliche Mengeneinheit und in der dritten der Preis für eine Mengeneinheit.

Um die Tabelle leichter im Browser sehen zu können, sollten Sie in der `<table>`-Anweisung den Parameter *border* angeben, ohne jedoch einen Wert zuzuweisen.

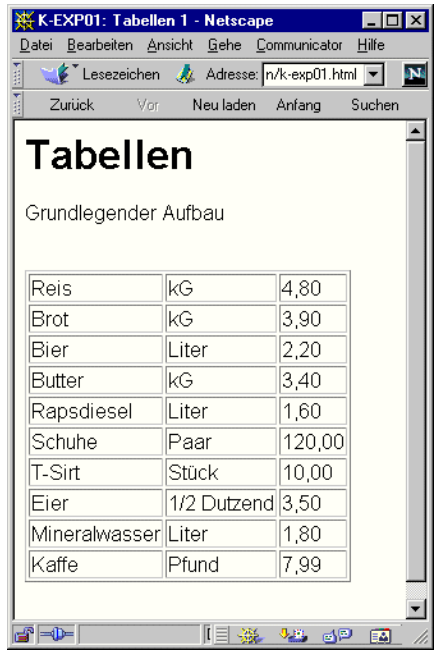

Abbildung 14.1: Einfache Informationstabelle

Tabellen stellen den Browser vor das Problem, dass in jeder Zeile die Anzahl der spezifizierten Spalten variieren können. Er kann damit erst dann die Tabelle anzeigen, wenn er sie vollständig gelesen hat. Das ist für den Browser technisch aufwendig und für den Leser entstehen Wartezeiten am Bildschirm.

Es wurde daher von den Browserherstellern am Standard vorbei ein Parameter für die `<table>`-Anweisung eingeführt, den die wichtigsten Browser heute immer noch verstehen.

Der Parameter ist *cols*. Man gibt ihm als Wert die Anzahl der Spalten mit. Nur im Netscape-Browser bedeutet die Anwesenheit des Parame-

ters *cols* auch, dass alle Spalten die gleiche Breite einnehmen. Da der Parameter *cols* von den Browsern unterschiedlich interpretiert wird, sollte man ihn besser nicht verwenden.

K-EXP02: Eigenschaften der ganzen Tabelle

Ergänzen Sie das erste Tabellenbeispiel um einige Parameter. Mit dem Parameter *bgcolor* lässt sich die Hintergrundfarbe festlegen. Die Farbe wird wie bei der Schriftfarbe mit einem Namen oder der RGB-Angabe mit Hex-Werten vorgegeben.

Mit *align* kann die Tabelle im umgebenden Block positioniert werden. Und mit *width* und *height* lässt sich die gewünschte Breite und Höhe vorgeben. Natürlich würde durch eine solche Vorgabe nicht die Ausgabe von Text in den einzelnen Zellen unterdrückt, sondern die Tabelle trotz der Größenangabe erweitert.

Mit *bordercolor* stellen Sie die Farbe der Tabellenumrandung ein. Der Rahmen erscheint nur, wenn der Parameter *border* alleine oder mit einem Wert größer als *0* angegeben wird.

Den Abstand des Zelleninhaltes bis zum Tabellengitter kann man mit *cellpadding* einstellen, den Abstand zwischen Zellen justiert man mit *cellspacing*.

Abbildung 14.2: Tabelleneigenschaften

Die optische Erscheinung einer Tabelle kann man durch Ein- und Ausschalten des Tabellengitters, Farben und die Definition der Abstände innerhalb einer Zelle und zwischen den Zellen einstellen. Experimentieren wir noch einmal mit einer eher ungewöhnlichen Einstellung der Zellenabstände.

K-EXP03: Tabelle mit stark getrennten Zellen

Erweitern Sie das bisherige Beispiel um Angaben zum Abstand innerhalb einer Zelle (*cellpadding*) und dem Abstand zwischen den Zellen (*cellspacing*). Wählen Sie insbesondere für den Abstand zwischen den Zellen einen deutlich sichtbaren Wert. Der *border*-Parameter soll entfernt werden.

Wie wird das Aussehen der Tabelle verändert und welche Farbe wird für die Zwischenräume benutzt?

Abbildung 14.3: Tabelle mit schwebenden Zellen

14.2 Spalten- und Zeilennamen und die Beschreibung

Tabellen gibt es, wie im Beispiel, oft als einfache Tabelle, die einer komplexen Liste ähneln. Der Inhalt einer Spalte wird meist durch einzelne Spaltenüberschriften beschrieben. Wir müssten dazu noch

in jede Zelle der ersten Zeile eine kleine Überschrift schreiben können. Diese Einträge können mit der `<th>`-Anweisung erstellt werden. Für Spaltenüberschriften wird die `<td>`-Anweisung durch die `<th>`-Anweisung ersetzt.

Der ganzen Tabelle kann man noch eine Unter- oder Überschrift mit der `<caption>`-Anweisung mitgeben. Diese Anweisung schreibt man unmittelbar nach der Startmarke der `<table>`-Anweisung. Mit dem *align*-Parameter kann man dann die Lage festlegen. Im Standard sind dabei alle 4 Seiten der Tabelle möglich. Es gibt damit folgende Werte: *top* | *bottom* | *left* | *right*. In den Browsern sind vor allem die Werte für *oben* / *top* und *unten* / *bottom* realisiert. Zumindest für meinen Geschmack ist es dabei ungewöhnlich, dass ohne Vorgabe mit dem *align*-Parameter die Tabelle eine Überschrift erhält. Mir würde eine Unterschrift eher gefallen. Aber das kann man ja einstellen.

K-EXP04: Spaltenüberschriften und Tabellenüberschrift

Ergänzen Sie das bisherige Beispiel um Spaltenüberschriften und eine Tabellenunterschrift.

Wie werden in Ihrem Browser die Überschriften der Spalten angezeigt?

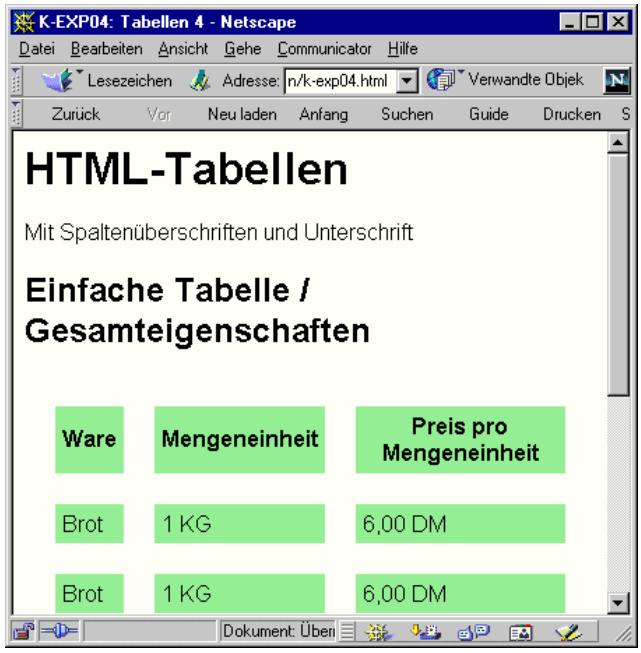

Abbildung 14.4: Spaltenüberschriften

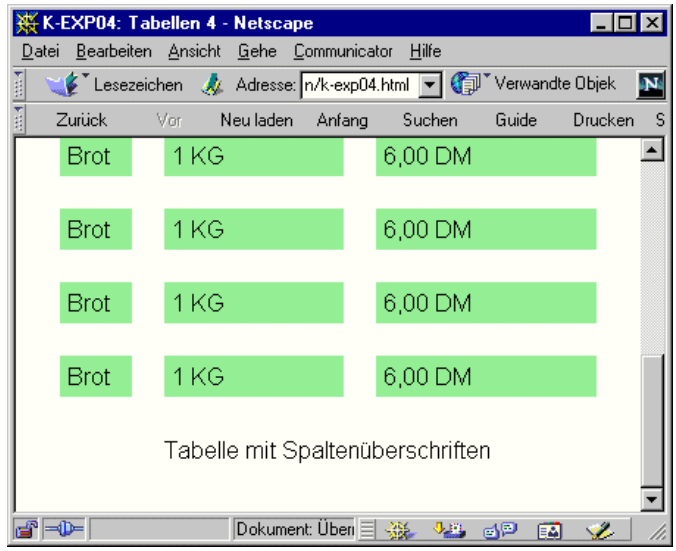

Abbildung 14.5: Tabellenunterschrift

Die Möglichkeit, in einer Tabelle einzelne Zellen mit einem Inhalt zu versehen, der als Überschrift ausgezeichnet wird, lässt sich auch nutzen, um die Idee einer Kreuztabelle in HTML darzustellen.

Ergänzt man die Auszeichnung der Spalten durch eine Auszeichnung der Zeilen hat man die gewünschte Kreuztabelle dargestellt. Dazu wird dann der jeweils erste Eintrag einer Zeile mit der `<th>`-Anweisung ausgezeichnet.

K-EXP05: Kreuztabelle

Erstellen Sie eine Kreuztabelle. Die Spalten sollen einige bekannte Orte, die Zeilen die (angenommen) mittleren Temperaturen der Monate zeigen.

Die Tabelle soll auch eine Beschriftung unterhalb der Tabelle erhalten.

Mit diesen Experimenten zur Tabellendarstellung haben die üblichen Tabellenformen abgedeckt und haben damit schon fast das Ende des Kapitels über Tabellen erreicht. Aber nur fast. Denn es fehlt noch eine der großen Aufgaben der Tabellen in HTML.

Abbildung 14.6: Kreuztabelle

14.3 Tabellen als Layout-Hilfe

Tabellen lassen sich sehr elegant auch zur Gestaltung des Layouts benutzen.

Wenn wir die Breite der Tabelle auf 100% des umgebenden Containers setzen, dann passt sich die Tabelle automatisch der Breite an. Jede der Zellen können Sie als eigene kleine Webseite betrachten.

Eine Zelle, die mit der `<td>`- oder `<th>`-Anweisung aufgebaut wird, kann ebenfalls eine Reihe von Parametern zur Steuerung akzeptieren.

- `bgcolor=#RGB` (Hintergrundfarbe einer Zelle)
- `align=left|center|right` (horizontale Ausrichtung des Inhaltes)
- `colspan=n` (Angabe, wie viele Spalten die Zelle umfasst)
- `rowspan=n` (Angabe, wie viele Zeilen die Zelle umfasst)
- `nowrap` (kein Umbruch des Inhaltstextes)
- `valign=bottom|middle|top` (vertikale Ausrichtung des Inhaltes)

Damit die Steuerung von Zellen wirksam wird, müssen in manchen Browsern Zellen einen Text oder ein anderes Element enthalten. Sollten Sie die Zelle absichtlich leer lassen wollen, dann geben Sie wenigstens ein Leerzeichen ein. Vollständig leere Zellen werden sonst nicht korrekt angezeigt.

Bei den Sonderzeichen haben wir gesehen, dass es ein Leerzeichen gibt, dass zwischen Wörtern stehen kann und einen Umbruch zwischen den Wörtern verhindert. Dieses Leerzeichen wird auch vom Browser nicht entfernt. Benutzen Sie ein oder mehrere * *, wenn eine Zelle leer sein soll.

Tabellen stehen in einem Container. Meist ist dies der Container, der mit der `<body>`-Anweisung eingerichtet wurde. Eine einzelne Zelle kann nun ebenfalls als Container wirken. Setzt man innerhalb einer `<body>`-Anweisung einen graphischen Querstrich (`<hr>`) mit 100%-Breite ein, wird er die gesamte Browser-Breite ausfüllen. Setzt man aber die gleiche Anweisung in eine Zelle ein, füllt der graphische Querstrich nur die Zellenbreite.

K-EXP06: Tabellenzellen als Container

Fügen Sie in eine beliebige Zelle ein Bild ein. Positionieren Sie es sowohl horizontal als auch vertikal in der Mitte der Tabellenzelle.

Gestalten Sie jede Zelle, so dass die ganze Seite als eine schön gestaltete Einstiegsseite dienen könnte. Nutzen Sie auch die Möglichkeit, einzelnen Zellen eine andere Hintergrundfarbe zu geben.

Setzen Sie am Schluss den Wert des Parameter *border* und *cellspacing* auf *0*.

Wie erscheint das Bild in Ihrem Lieblingsbrowser? Können Sie es auch in einem anderen Browser betrachten?

Wenn unsere Seite fertig ist, dann können wir den Rahmen und den Zwischenraum zwischen den Zellen abschalten, um einen einheitlichen Eindruck zu erwecken.

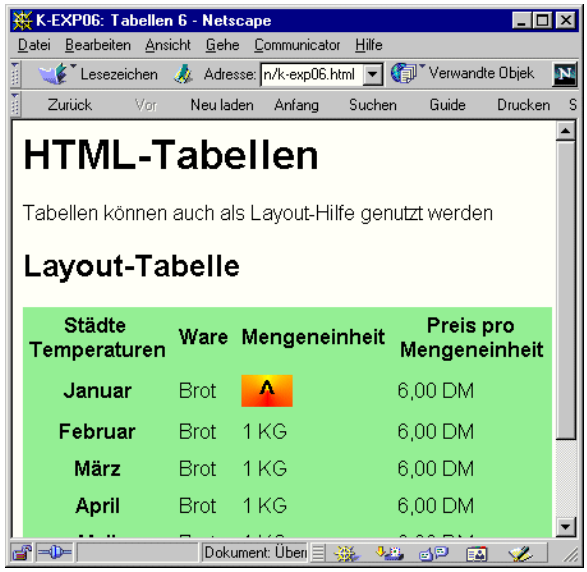

Abbildung 14.7: Bild in der Mitte der Zelle

K-EXP07: Unterschiedlich große Zellen

Schreiben Sie eine HTML-Seite, die mehrere Spalten und Zeilen umfasst. Eine Zelle soll nun vergrößert werden, so dass sie jeweils zwei Spalten und Zeilen umfasst. Sollten schon Zellen vorhanden sein, z.B. durch einen HTML-Editor, dann sollten die nun überflüssigen Zellen entfernt werden.

Platzieren Sie in die Mitte der großen Zelle ein Bild. Die Ausrichtung soll horizontal und vertikal erfolgen.

Weisen Sie einzelnen Zellen unterschiedliche Hintergrundfarben zu. Verwenden Sie keinen *border*-Parameter.

Der Abstand zwischen den Zellen müsste evtl. auf *0* gesetzt werden, damit die Farbe der Seite nicht durchscheint.

Wir geraten hier langsam aber sicher in gefährliche Gebilde. Browser neigen dazu, die Anzeige der Seiten nach eigenen Spielregeln zu gestalten. Der Grund dafür kann sein, dass nichts über die Auslegung in den verschiedenen Versionen des Standards festgelegt wurde oder dass eben spezielle Parameter (wie der *cols*-Parameter) unterschiedlich interpretiert werden.

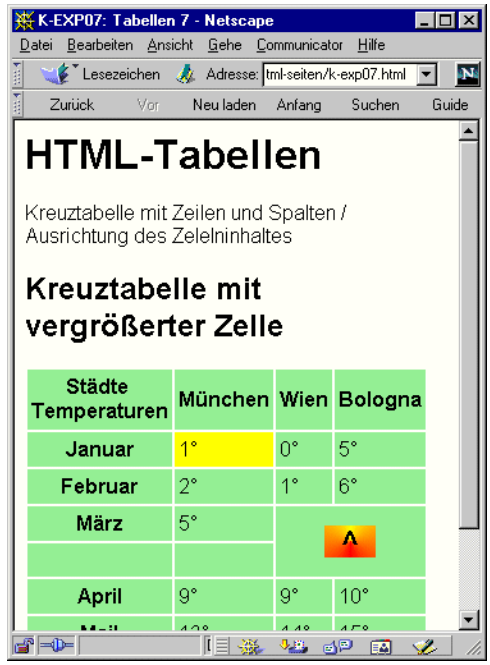

Abbildung 14.8: Vergrößerte Zellen

Als Ausweg aus den verschiedenen Interpretationen der Browser bleibt uns nur, unsere Seiten nach bestem Wissen zu gestalten und in verschiedenen Browsern und verschiedenen Versionen zu testen. Dabei wird es sicher sinnvoll sein, mehrere „alte“ Browser (z.B. Netscape 2.0) auszuprobieren. Nicht jeder Benutzer wird immer die neuesten Browser benutzen.

Um eine möglichst gleichartige Darstellung in verschiedenen Produkten zu erreichen, sollte man das Aussehen weitestgehend festlegen. Die Größe einer Tabelle sollte daher möglichst mit *width* und *height* vorgegeben werden. Nutzen Sie also möglichste viele Parameter für die Einstellungen.

14.4 Design einer Startseite

Obwohl man nach Möglichkeit das Erscheinungsbild der Seite immer gegen ältere Versionen und verschiedene Hersteller testen sollte, braucht man keine Angst zu haben, neue Parameter oder Anweisungen zu verwenden. Wir haben ja schon wiederholt besprochen, dass von dem Browser unbekannte Parameter und Anweisungen einfach überlesen werden.

Und schließlich noch ein Tipp für die Startseite eines Webauftrittes: verwenden Sie nicht zu viele Stilelemente. Es kann sonst unangenehm auf den Besucher wirken. Wenn Sie eine Tabelle benutzen, nutzen Sie die Seite nicht ganz aus (vielleicht nur zu 90%). Und füllen Sie auch die einzelnen Zellen nicht zu sehr. Nur so kann sich das Bild an die vorhandene Bildschirmgröße beim Betrachter anpassen.

Es gibt außerdem eine kleine Spielregel beim Aufbau der Seiten. Oberhalb des ersten HTML-Elementes einer Seite existiert immer eine einzelne Leerzeile, um ein wenig Abstand vom Fensterbeginn zu haben. Setzen Sie eine Tabelle auf 100%-Höhe, bedeutet dies, dass die Tabelle die Fenstergröße annimmt und zusätzlich die Leerzeile angezeigt wird. Sie sehen damit immer den Schiebebalken an der rechten Seite am Browserfenster. Verringern Sie die Höhe etwas verschwindet der Schiebebalken.

K-EXP08: Design einer Startseite

Schreiben Sie eine Startseite für einen gedachten oder wirklichen Auftritt im WWW.

Abbildung 14.9: Design einer einfachen Startseite

Nutzen Sie eine Tabelle, um 4 Zellen zu definieren: eine kleine in der oberen linken Ecke, eine schmale über den Rest der Seite, ein schmale senkrechte Zelle in der zweiten Zeile und schließlich eine große Zelle in der rechten unteren Ecke. Setzen Sie dazu einfach die Breite und Höhe der ersten kleinen Zelle auf 10%.

Beachten Sie auch den Tipp zur Gesamtgröße der Tabelle.

14.5 Erweiterte Gestaltung von Tabellen oder Zellen

Tabellen oder Zellen in Tabellen können, wie wir gesehen haben, mit Farben hinterlegt werden. Dies ist eine graphisch einfache und sehr schnelle Variante, da der Browser das Ausfüllen mit Farben übernehmen kann.

Für anspruchsvollere Hintergründe können Bilder benutzt werden, um den Hintergrund zu gestalten. Beachten Sie aber, dass kontrastreiche Hintergrundbilder mit kräftigen Farben den Vordergrund fast verschwinden lassen. Je dezenter der Hintergrund ist, desto eher wird er den Leser der Seite ansprechen.

K-EXP09/10: Tabellen und Zellen mit Hintergrund-Graphiken

Ergänzen Sie ein bisheriges Tabellenbeispiel um je ein Hintergrundbild für die ganze Tabelle und für eine einzelne Zelle.

Erstellen Sie zwei Varianten: einmal mit sehr stark strukturierten Bildern und einmal mit dezenten Bildern. Suchen Sie bei Bedarf im WWW nach Bildern.

Das Ergebnis kann bei sehr kontrastreichen Bildern so aussehen:

Abbildung 14.10: Hintergrundbilder in Tabellen

Mit richtig gewählten kontrastarmen und dezenten Hintergrundbildern ergibt sich ein weit besserer Effekt.

Abbildung 14.11: Dezente Hintergrundbilder

14.6 Präzise Positionierung von Zellen

Bisher haben wir die Zellen möglichst nicht ganz gefüllt, um dem Browser die Möglichkeit zu geben, sich an die verschiedensten Auflösungen anzupassen.

In verschiedenen Webauftritten werden Sie als Benutzer gefragt, welche Auflösung Sie verwenden. Die Seiten sind hier genau auf die Auflösung des Leserbildschirms angepasst.

Der Trick, um exakt ausgerichtete Seiten aufzubauen, ist recht einfach. Innerhalb einer oder mehrerer Zellen werden Texte oder graphische Elemente untergebracht, die eine feste Größe besitzen. Die Zelle kann nicht kleiner als die darin enthaltenen Elemente werden.

Bei graphischen Elementen existieren Parameter, um die Breite anzugeben. Bei Texten kann man einen Text einfügen, auch wenn er nur aus festen Leerzeichen besteht, und durch den Parameter *nowrap* einen Zeilenumbruch verhindern. Auch so lassen sich Minimalgrößen definieren.

K-EXP11: Definierte Größen für Zellen

Nutzen Sie ein bisheriges Tabellenbeispiel und fügen in alle Zellen einer Zeile graphische Trennlinien mit `<hr>` ein. Jede Trennlinie soll eine feste Breite in Pixel erhalten.

Abbildung 14.12: Feste Minimal-Größen von Zellen

Laden Sie die Seite in den Browser und beschreiben Sie, wie sich die Tabelle bei einem immer kleineren Fenster verhält.

Im nächsten Kapitel

Mit den Tabellen haben wir die einfachste Variante der Seitengestaltung kennen gelernt.

Darüber hinaus kann man die ganze Seite fest unterteilen und in jeden der Teile eine eigene HTML-Seite laden. Gerade für Benutzerführungen, die dem Benutzer auch bei wechselnden Seiten immer die gleiche Navigation im Webauftritt ermöglichen sollen, wird dies verwendet. Die Technik heißt *Seitenrahmen / frames*.

14.7 Lösungen

K-EXP01: Einfache Preistabelle

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP01: Tabellen 1</title>
</head>
<body>
<h1>
Tabellen</h1>
Grundlegender Aufbau
<br> 
<br> 
<table BORDER >
<tr>
<td>Reis</td>
<td>kG</td>
<td>4,80</td>
</tr>
<tr>
<td>Brot</td>
<td>kG</td>
<td>3,90</td>
</tr>
<tr>
<td>Bier</td>
<td>Liter</td>
```

```
<td>2,20</td>
</tr>
<tr>
<td>Butter</td>
<td>kG</td>
<td>3,40</td>
</tr>
<tr>
<td>Rapsdiesel</td>
<td>Liter</td>
<td>1,60</td>
</tr>
<tr>
<td>Schuhe</td>
<td>Paar</td>
<td>120,00</td>
</tr>
<tr>
<td>T-Sirt</td>
<td>St&uuml;ck</td>
<td>10,00</td>
</tr>
<tr>
<td>Eier</td>
<td>1/2 Dutzend</td>
<td>3,50</td>
</tr>
<tr>
<td>Mineralwasser</td>
<td>Liter</td>
<td>1,80</td>
</tr>
<tr>
<td>Kaffe</td>
<td>Pfund</td>
<td>7,99</td>
</tr>
</table>
<br> 
</body>
</html>
```

K-EXP02: Eigenschaften der ganzen Tabelle

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
```

```
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP02: Tabellen 2</title>
</head>
<body>
<h1>
HTML-Tabellen</h1>
Grundlegender Aufbau
<h2>
Einfache Tabelle / Gesamteigenschaften</h2>
<table BORDER=4 BGCOLOR="#90EE90" bordercolor="red" >
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
</table>
</body>
</html>
```

K-EXP03: Tabelle mit stark getrennten Zellen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP03: Tabellen 3</title>
</head>
<body>
<h1>
HTML-Tabellen</h1>
Grundlegender Aufbau, mit "cellspacing" ohne "border"
<h2>
Einfache Tabelle / Gesamteigenschaften</h2>
<table CELLSPACING=20 CELLPADDING=5 BGCOLOR="#90EE90" bordercolor="red" >
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
```

```
</table>
</body>
</html>
```

K-EXP04: Spaltenüberschriften und Tabellenüberschrift

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP04: Tabellen 4</title>
</head>
<body>
<h1>
HTML-Tabellen</h1>
Mit Spalten&uuml;berschriften und Unterschrift
<h2>
Einfache Tabelle / Gesamteigenschaften</h2>
<table CELLSPACING=20 CELLPADDING=5 BGCOLOR="#90EE90" bordercolor="red" >
<tr>
<th>Ware</th>
<th>Mengeneinheit</th>
<th>Preis pro Mengeneinheit</th>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
```

```
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<caption align="bottom">Tabelle mit Spalten&uuml;berschriften</caption>
</table>
</body>
</html>
```

K-EXP05: Kreuztabelle

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP05: Tabellen 5</title>
</head>
<body>
<h1>
HTML-Tabellen</h1>
Kreuztabelle mit Zeilen und Spalten
<h2>
Kreuztabelle</h2>
<table CELLSPACING=5 CELLPADDING=3 COLS=4 BGCOLOR="#90EE90" borderco-
lor="red" >
<tr>
<th>St&auml;dte
<br>Temperaturen</th>
<th>M&uuml;nchen</th>
<th>Wien</th>
<th>Bologna</th>
</tr>
<tr>
<th>Januar</th>
<td>1&deg;</td>
<td>0&deg;</td>
<td>5&deg;</td>
</tr>
<tr>
<th>Februar</th>
```

```
<td>2&deg;</td>
<td>1&deg;</td>
<td>6&deg;</td>
</tr>
<tr>
<th>M&auml;rz</th>
<td>5&deg;</td>
<td>4&deg;</td>
<td>7&deg;</td>
</tr>
<tr>
<th>April</th>
<td>9&deg;</td>
<td>9&deg;</td>
<td>10&deg;</td>
</tr>
<tr>
<th>Mail</th>
<td>13&deg;</td>
<td>14&deg;</td>
<td>15&deg;</td>
</tr>
<tr>
<th>Juni</th>
<td>16&deg;</td>
<td>17&deg;</td>
<td>18&deg;</td>
</tr>
<caption ALIGN=BOTTOM>Kreuztabelle</caption>
</table>
</body>
</html>
```

K-EXP06: Tabellenzellen als Container

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP06: Tabellen 6</title>
</head>
<body>
<h1>
HTML-Tabellen</h1>
Tabellen k&ouml;nnen auch als Layout-Hilfe genutzt werden
```

```
<h2>
Layout-Tabelle</h2>
<table BORDER=0 CELLSPACING=0 CELLPADDING=5 WIDTH="100%" BGCOLOR="#90EE90"
bordercolor="red" >
<tr>
<th>St&auml;dte
<br>Temperaturen</th>
<th>Ware</th>
<th>Mengeneinheit</th>
<th>Preis pro Mengeneinheit</th>
</tr>
<tr>
<th>Januar</th>
<td>Brot</td>
<td ALIGN=CENTER VALIGN=CENTER><img SRC="auf.gif" NOSAVE height=27
width=44></td>
<td>6,00 DM</td>
</tr>
<tr>
<th>Februar</th>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<th>M&auml;rz</th>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<th>April</th>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<th>Mail</th>
<td>Brot</td>
<td>1 KG</td>
<td>6,00 DM</td>
</tr>
<tr>
<th>Juni</th>
<td>Brot</td>
<td>1 KG</td>
```

```
<td>6,00 DM</td>
</tr>
<caption ALIGN=BOTTOM>Kreuztabelle</caption>
</table>
</body>
</html>
```

K-EXP07: Unterschiedlich große Zellen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>K-EXP07: Tabellen 7</title>
</head>
<body>
<h1>
HTML-Tabellen</h1>
Kreuztabelle mit Zeilen und Spalten / Ausrichtung des Zelelninhaltes
<h2>
Kreuztabelle mit vergr&ouml;&szlig;erter Zelle</h2>
<table CELLSPACING=0 CELLPADDING=3 COLS=4 WIDTH="90%" HEIGHT="70%" BGCO-
LOR="#90EE90" bordercolor="red" >
<tr>
<th>St&auml;dte
<br>Temperaturen</th>
<th>M&uuml;nchen</th>
<th>Wien</th>
<th>Bologna</th>
</tr>
<tr>
<th>Januar</th>
<td BGCOLOR="#FFFF00">1&deg;</td>
<td>0&deg;</td>
<td>5&deg;</td>
</tr>
<tr>
<th>Februar</th>
<td>2&deg;</td>
<td>1&deg;</td>
<td>6&deg;</td>
</tr>
<tr>
<th>M&auml;rz</th>
<td>5&deg;</td>
```

```
<td ALIGN=CENTER VALIGN=CENTER COLSPAN="2" ROWSPAN="2"><img SRC="auf.gif"
NOSAVE height=25 width=40></td>
</tr>
<tr>
<td> </td>
<td> </td>
</tr>
<tr>
<th>April</th>
<td>9&deg;</td>
<td>9&deg;</td>
<td>10&deg;</td>
</tr>
<tr>
<th>Mail</th>
<td>13&deg;</td>
<td>14&deg;</td>
<td>15&deg;</td>
</tr>
<tr>
<th>Juni</th>
<td>16&deg;</td>
<td>17&deg;</td>
<td BGCOLOR="#33FF33">18&deg;</td>
</tr>
<caption ALIGN=BOTTOM>Kreuztabelle</caption>
</table>
</body>
</html>
```

K-EXP08: Design einer Startseite

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="Author" content="Walter Herglotz">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Description" content="Tabellen in HTML">
   <title>K-EXP08:  Formatierung mit Tabellen</title>
</head>
<body >
 
<table BORDER CELLPADDING=4 WIDTH="90%" HEIGHT="90%" >
<tr>
<td WIDTH="10%" HEIGHT="10%" BGCOLOR="#CCCCCC">
<center><img SRC="auf.gif" NOSAVE height=25 width=40></center>
```

```
</td>
<td WIDTH="90%">
<center><a NAME="seitenanfang"></a><font color="#FF0000"><font
size=+2>Gro&szlig;e
Navigationsleiste</font></font></center>
</td>
</tr>
<tr>
<td WIDTH="10%" HEIGHT="90%">Kleine 
<br>Navigationsleitste</td>
<td WIDTH="90%" HEIGHT="90%">
<li>
<b><font size=+1>Inhalt der Seite</font></b></li>
<br>
<hr WIDTH="100%"></td>
</tr>
</table>
</body>
</html>
```

K-EXP09/10: Tabellen und Zellen mit Hintergrund-Graphiken

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="Author" content="Walter Herglotz">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Description" content="Tabellen in HTML">
   <title>K-EXP09:  Formatierung mit Tabellen</title>
</head>
<body >
 
<table BORDER CELLPADDING=4 WIDTH="90%" HEIGHT="90%" BACKGROUND="b-exp01-
1.gif" NOSAVE >
<tr>
<td WIDTH="10%" HEIGHT="10%" BGCOLOR="#CCCCCC">
<center><img SRC="auf.gif" NOSAVE height=25 width=40></center>
</td>
<td WIDTH="90%">
<center><a NAME="seitenanfang"></a><font color="#FF0000"><font
size=+2>Gro&szlig;e
Navigationsleiste</font></font></center>
</td>
</tr>
<tr>
<td WIDTH="10%" HEIGHT="90%">Kleine 
```

```
<br>Navigationsleitste</td>
<td WIDTH="90%" HEIGHT="90%" BACKGROUND="auf.gif" NOSAVE>
<li>
<b><font size=+1>Inhalt der Seite</font></b></li>
<br>
<hr WIDTH="100%"></td>
</tr>
</table>
</body>
</html>

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="Author" content="Walter Herglotz">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Description" content="Tabellen in HTML">
   <title>K-EXP10:  Formatierung mit Tabellen</title>
</head>
<body>
 
<table BORDER CELLPADDING=4 WIDTH="90%" HEIGHT="90%" BACK-
GROUND="hintergrund2.gif" NOSAVE >
<tr>
<td WIDTH="10%" HEIGHT="10%" BGCOLOR="#CCCCCC">
<center><img SRC="auf.gif" NOSAVE height=25 width=40></center>
</td>
<td WIDTH="90%">
<center><a NAME="seitenanfang"></a><font color="#FF0000"><font
size=+2>Gro&szlig;e
Navigationsleiste</font></font></center>
</td>
</tr>
<tr>
<td WIDTH="10%" HEIGHT="90%">Kleine 
<br>Navigationsleitste</td>
<td WIDTH="90%" HEIGHT="90%" BACKGROUND="hintergrund1.gif" NOSAVE>
<li>
<b><font size=+1>Inhalt der Seite</font></b></li>
<br>
<hr WIDTH="100%"></td>
</tr>
</table>
</body>
</html>
```

K-EXP11: Definierte Größen für Zellen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
   <meta name="Author" content="Walter Herglotz">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Description" content="Tabellen in HTML">
   <title>K-EXP11:  Formatierung mit Tabellen</title>
</head>
<body >
 
<table BORDER CELLPADDING=4 WIDTH="90%" HEIGHT="90%" BACK-
GROUND="hintergrund2.gif" NOSAVE >
<tr>
<td WIDTH="10%" HEIGHT="10%" BGCOLOR="#CCCCCC">
<center><img SRC="auf.gif" NOSAVE height=25 width=40></center>
</td>
<td WIDTH="90%">
<center><a NAME="seitenanfang"></a><font color="#FF0000"><font
size=+2>Gro&szlig;e
Navigationsleiste</font></font></center>
</td>
</tr>
<tr>
<td WIDTH="10%" HEIGHT="90%">Kleine 
<br>Navigationsleitste
<hr WIDTH="120"></td>
<td WIDTH="90%" HEIGHT="90%" BACKGROUND="hintergrund1.gif" NOSAVE>
<li>
<b><font size=+1>Inhalt der Seite</font></b></li>
<br>
<hr WIDTH="500"></td>
</tr>
</table>
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

15 Seitengestaltung mit Rahmen

Benutzer und Autoren lassen sich in verschiedene Gruppen einteilen. Je sachorientierter die Gruppe ist, wie es z.B. bei wissenschaftlichen Veröffentlichungen der Fall ist, desto schlichter kann eine Seite gestaltet sein. Umgekehrt gilt vermutlich auch, dass ein Massenpublikum eine schön gestaltete Seite mit einfachen Navigationsmöglichkeiten erwartet.

Da uns Menüs im Sinne einer klassischen graphischen Benutzeroberfläche fehlen, wurden verschiedene Konzepte entwickelt, um dem Benutzer die Navigation durch einen Auftritt im Web zu erleichtern.

Ein Konzept, das heute oft in Verbindung mit dynamisch erzeugten Webseiten genutzt wird, sind die besprochenen Tabellen. In jeder Seite werden dann Haupt- und Detailnavigation in einer Kopfzeile und in einer Seitenspalte wiederholt und jeweils angepasst. Bei einer manuellen Erstellung würde der Verwaltungsaufwand jeden Rahmen sprengen, aber wenn die Seiten von Programmen generiert werden, sieht die Situation anders aus. Ein Grund für Tabellen ist die meist gute Geschwindigkeit.

15.1 Rahmen (Frames)

Ein anderes Konzept arbeitet mit einer Einteilung des vorhandenen Bildschirms in Rahmen. Dies kann im ersten Moment fast wie die Zellen einer Tabelle aussehen. Es gibt dabei aber wesentliche Unterschiede. Die Rahmen werden mit unterschiedlichen HTML-Dateien gefüllt. Sind also in einer Rahmengruppe drei Rahmen vorhanden, werden auch drei Webseiten geladen.

Ein wichtiges Merkmal ist dabei, dass sich Rahmen in ihrer Größe nicht verändern und der Inhalt nur auf einen Benutzerbefehl hin ausgetauscht wird. Damit kann man z.B. recht einfach eine feste Na-

vigationsleiste im Kopf der Seite einrichten, die dem Benutzer immer zur Verfügung steht.

Jedem Rahmen gibt man einen Namen und kann dann beim Klicken auf einzelne Verweise gezielt Dateien in bestimmten Rahmen laden.

Frames bieten eine oft elegante Art, eine gute Benutzerkommunikation aufzubauen. Allerdings wird dies durch leicht erhöhte Ladezeiten und eine manchmal schwierige Verwaltung der gleichzeitig angezeigten Inhalte bezahlt.

Eine Zeitlang waren Frames sehr in Mode. Man konnte kaum eine Webseite aufrufen, ohne in einem Wust von Frames zu geraten. Die überschäumende Begeisterung (engl. hype) hat etwas zu Gunsten von dynamisch erzeugten Tabellen nachgelassen.

15.2 Die Frames Steuerdatei

Sehen wir uns den Aufbau einer Einstiegsseite in einen Webauftritt mit Frames (Rahmen) im Beispiel an. Diese Datei wird als Steuerdatei fungieren.

An die Stelle der bisherigen `<body>`-Anweisung tritt die Angabe der Rahmengruppe mit der `<frameset>`-Anweisung. Dazu muss das gesamte Fenster des Benutzers in rechteckige Bereiche untergliedert werden. Im Beispiel teilen wir nach Zeilen (*rows*). Die erste Zeile soll dabei 65 Pixel hoch sein und die zweite Zeile nimmt den Rest der Seite ein. Ein Sternchen ist wie so oft der geeignete Platzhalter.

Im einfachsten Fall würden nun zwei Frames mit der `<frame>`-Anweisung folgen, die die beiden Zeilen füllen. Hier gibt es zuerst eine einfache Zeile und danach folgt erneut eine Rahmengruppe, die den Platz der zweiten Zeile einnimmt. Der zur Verfügung stehende Bereich wird nun senkrecht in Spalten (*cols/columns*) eingeteilt. Die Angabe der Größen ist jetzt relativ in Prozent angegeben.

Mit der Ende-Marke der `<frameset>`-Anweisung schließt man die gesamte Gruppe. Die Hauptaufgabe der `<frameset>`-Anweisung ist schlicht die Einteilung des Darstellungsfensters im Browser. Zusätzlich geben Parameter Vorgaben für alle Rahmen innerhalb der Gruppe.

In unserer Steuerdatei gibt es momentan keine `<body>`-Anweisung.

Beispiel:

```
<frameset rows="65,*" >
<frame src= "thtml1-22header.html" name="f1" scrolling="no" framebor-
der="0" noresize>
<frameset cols = "15%,85%" >
<frame src= "thtml1-22nav.html" name="f2" frameborder="0"noresize>
<frame src= "thtml1-22main.html" name="fmain" frameborder="0" noresize>
</frameset>
```

Der Bildschirm ist damit in drei Bereiche untergliedert und kann damit drei HTML-Dateien gleichzeitig anzeigen. Es müssen keine HTML-Dateien sein. Es könnten auch Bilder sein. Aber das würde der Idee widersprechen, die Rahmen auch zur Navigation für den Benutzer zu verwenden.

Sehen wir uns die möglichen Parameter an.

Mit *frameborder* kann man einem Rahmen eine Umrandung geben. Ein Wert von *0* schaltet den Rahmen ab. Ganz wichtig für jeden Rahmen ist der mit *name* angegebene Name. Ohne einen individuellen Namen kann man keine weiteren Inhalte in den Rahmen hinein nachladen.

Das ursprünglich in der `<frameset>`-Anweisung festgelegte Größenverhältnis kann durch den Benutzer durch Ziehen an Kanten verändern werden. Mit *noresize* nimmt man dem Benutzer die Möglichkeit, die Größenverhältnisse anzupassen.

Der Browser wird bei Bedarf an den Kanten der Rahmen Schiebeleisten anbringen, um auch größere Inhalte zugänglich zu machen. Mit dem Parameter *scrolling* kann man den Browser darüber hinaus anweisen, immer oder nie die Schiebeleisten einzublenden. Die Werte sind dazu *yes* | *no*.

Und schließlich geben wir mit *src* (source / Quelle) an, welche Datei am Anfang zu laden ist.

R-EXP01: Grundlegende Seite mit Rahmen

Schreiben Sie eine HTML-Datei, die das Benutzerfenster mit Rahmen unterteilt. Es sollen drei Rahmen werden. Ein querliegender Rahmen soll dabei die Hauptnavigationsleiste enthalten, darunter liegen ein schmaler Rahmen für die Detailnavigation und der Hauptrahmen zur Anzeige der Inhalte.

Laden Sie in jeden der drei Rahmen eine passende HTML-Seite.

Auf die Angabe von Verweisen verzichten wir noch einen Moment.

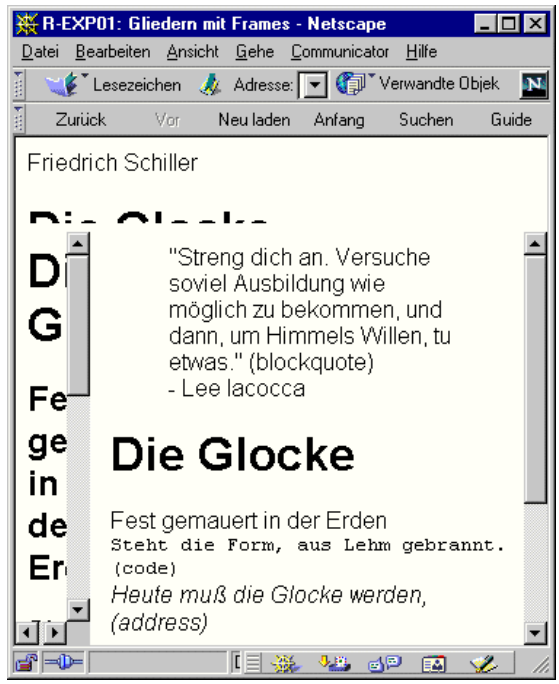

Abbildung 15.1: Seitenaufbau mit Rahmen

Das Ergebnis zeigt einen kleinen Ausschnitt der Randbedingungen, die wir bei Rahmen beachten sollten. Der obere, querliegende Rahmen kann die Datei gar nicht anzeigen, da sie zu groß ist. Hätten wir ein Bild in den oberen Rahmen eingefügt, dann würde sich auch hier die Frage stellen, ob das Bild angezeigt werden kann.

Im Beispiel wurden 65 Pixel für den oberen Rand reserviert.

Ist das Bild größer, wird es abgeschnitten. Ein ganz ähnlicher Effekt ergibt sich, wenn man den querliegenden Rahmen mit einem prozentualen Anteil an der Seitenhöhe spezifiziert und ein Bild lädt, das gut hineinpasst. Hat man die Seite bei einer Auflösung von 800x600 Pixel erstellt, wird man sich vielleicht einen Moment wundern, wenn man die gleiche Seite mit geringerer Auflösung betrachtet. Wieder sind die Chancen gut, dass das gleich groß gebliebene Bild nicht mehr in den kleiner gewordenen Rahmen passt.

Die Bemerkungen sollen Sie keineswegs von den Rahmen abschrecken. Aber sie sollten hier sehr auf ein solides Design achten und die Seiten sicherheitshalber unter verschiedenen Auflösungen und Browsern testen.

15.3 Verweise und Rahmen

Die Aufteilung des Anzeigefensters im Browser stellt uns nun vor die Aufgabe, dem Benutzer gezielt die Möglichkeit zu geben, durch Anklicken eines Verweises, eine Datei in einem bestimmten Fenster anzuzeigen. Dazu muss die Funktionalität der Verweise ausgebaut werden.

Nehmen wir einmal an, dass in einem senkrechten Rahmen links eine Liste mit Verweisen auf die möglichen Dateien steht. Das Menü soll immer sichtbar bleiben und die angeforderten Inhalte sollen im Hauptrahmen der Gruppe angezeigt werden.

Der neue Parameter für den Verweis im `<a>`-Befehl heißt *target* (Ziel). Er kann den frei vergebenen Namen aus der Rahmendefinition erhalten.

Hier setzt man bei der Verwendung kein „#“-Zeichen vor den Namen. Verwechslungen sind hier nicht möglich.

```
<a href="url" target="Rahmenname">Klicktext</a>.
```

R-EXP02: Rahmen mit Verweisen

Ändern Sie die bisherige Rahmenaufgabe ab. Im Detailnavigationsfenster soll nun eine Datei mit einer Anzahl von Verweisen stehen. Alle angeklickten Verweise sollen dazu führen, dass die Inhalte im Hauptrahmen angezeigt werden.

Die Navigationsdatei soll Verweise zu allen Experimenten eines beliebigen Kapitels enthalten, die eben dann im Hauptfenster angezeigt werden.

Mit unserem Beispiel erhalten wir schnell eine sehr hilfreiche Struktur. wenn wir für jedes Kapitel eine Navigationsdatei schreiben, dann haben wir immer ein Kapitel im schnellen Zugriff. Nun müssen wir nur noch die Navigation einrichten, die uns zu den Kapitel führt.

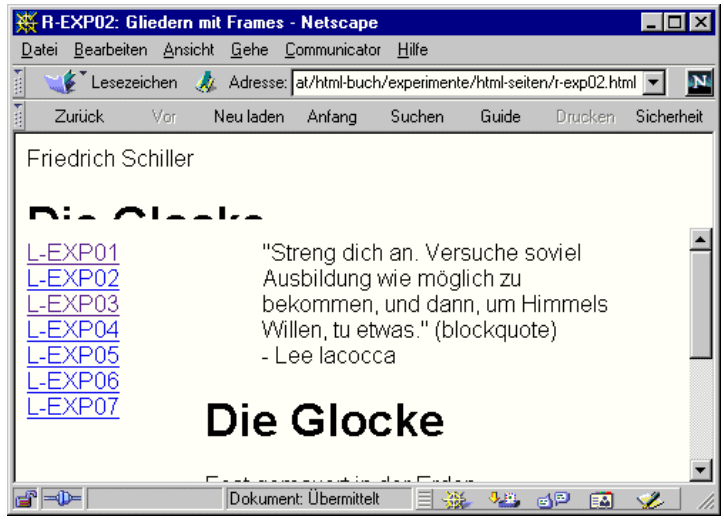

Abbildung 15.2: Detailnavigation mit Rahmen

R-EXP03:Unterteilung in Haupt- und Detailnavigation

Ergänzen Sie nun unsere Rahmenkonstruktion mit einer Hauptnavigations-Ebene. Die Datei im oben querliegenden Rahmen soll auf die Detailnavigationsdateien der einzelnen Kapitel verweisen. Im Kopf steht also eine Anzahl von Verweisen – mindestens einer pro Kapitel.

Natürlich ist es für das Verständnis nicht notwendig, jetzt alle Dateien zu schreiben. Es würde der eine Verweis auf die im vorhergegangenen Kapitel angelegte Detailnavigationsdatei genügen.

Einen Unterschied zum vorhergegangenen Experiment müssen wir allerdings machen. Sowohl die Detailnavigation als auch der Hauptrahmen sollen am Anfang mit einer leeren Seite gefüllt werden.

Das Ergebnis sollten wir uns schrittweise ansehen. Zuerst schauen wir auf den Urzustand nach dem Laden.

Da wir sowohl im Hauptrahmen als auch im Detailnavigationsrahmen jeweils eine leere HTML-Seite geladen haben, erscheint dort kein Inhalt. Nur in der Hauptnavigationsleiste sind zwei Verweise zu sehen.

Abbildung 15.3: Rahmen nach dem Laden

Nun klicken wir auf den TABELLEN-Verweis.

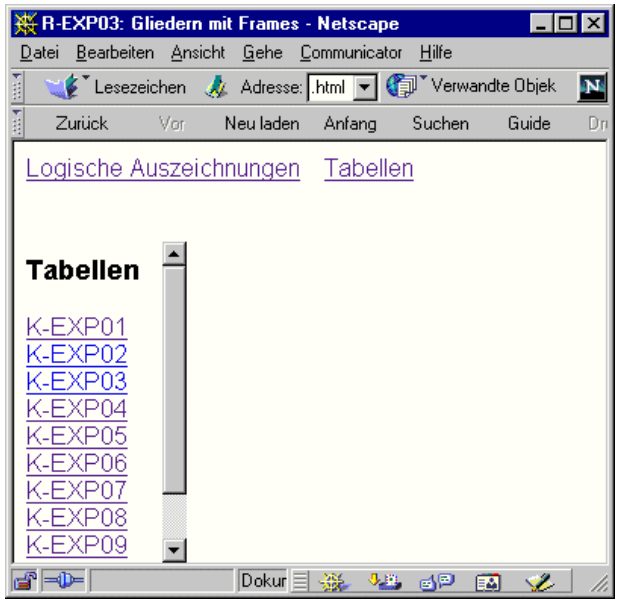

Abbildung 15.4: Mit geladener Detailnavigation

Der Hauptrahmen bleibt auch hier noch leer. Und nun klicken wir ein zweites Mal. Nun aber klicken wir auf einen Verweis in der Detailnavigation.

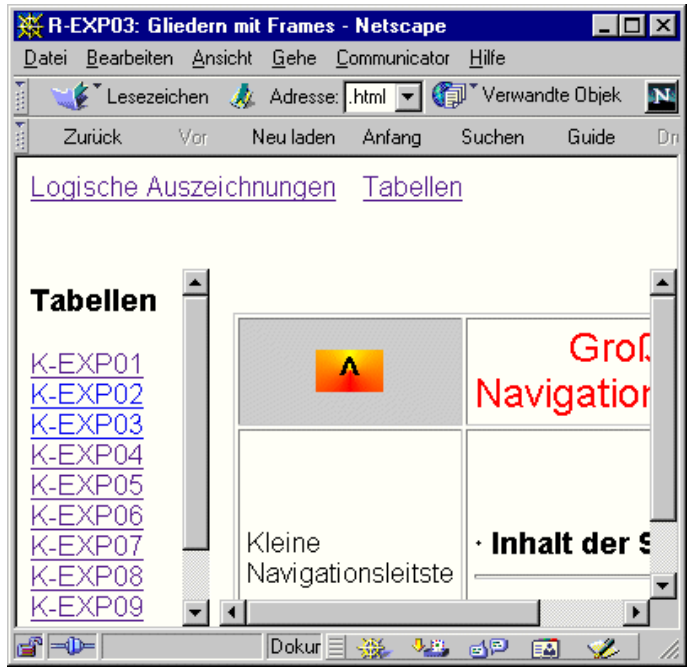

Abbildung 15.5: Nach dem zweiten Klick

Nach zwei Klicks haben wir also unsere gewünschte Datei gefunden.

Gehen Sie an dieser Stelle noch einmal zurück in die Hauptnavigation und klicken Sie einen anderen Eintrag an. Wenn alles richtig geschrieben wurde, dann steht nun im Rahmen der Detailnavigation eine neue Auswahlliste zur Verfügung. Nur der Inhalt des Hauptfensters passt irgendwie nicht mehr zu unserer Detailauswahl.

Da wir immer nur eine Datei durch einen einzelnen Klick in HTML aufrufen können, ist der alte Inhalt des Hauptfensters stehen geblieben und wird erst durch den zweiten Klick in der Detailnavigation überschrieben.

Nicht sehr schön, aber auch mit HTML nicht zu vermeiden.

Diese Art von Gestaltungsproblemen können wir allein mit HTML nicht mehr beheben. Wenn man mit einem Mausklick in der Hauptnavigation erreichen will, dass gleichzeitig eine Navigationsdatei in der Detailauswahl angezeigt und im Hauptfenster die Startseite des Themas geladen wird, dann kann dies nur noch mit Programmierung geschehen.

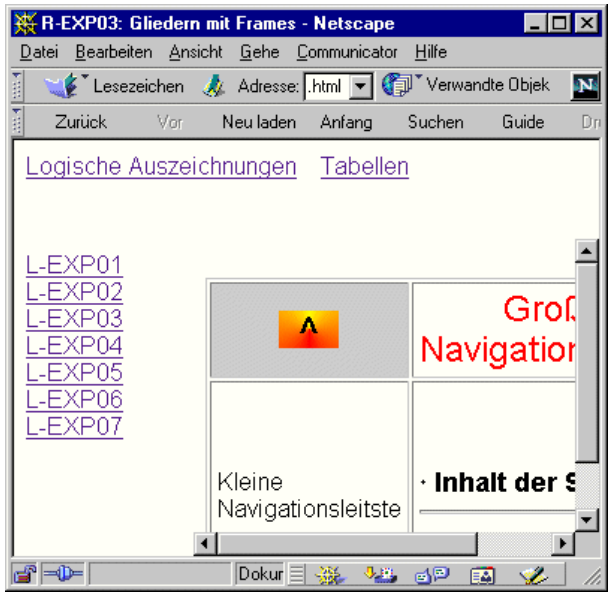

Abbildung 15.6: Unzusammenhängende Anzeige

Zwar geht die Programmierung über den Themenumfang des Buches hinaus, aber ein paar Experimente werden wir uns im Kapitel zu Javascript dennoch ansehen.

15.4 Rahmensteuerung

Rahmen benötigen noch spezielle Steuerungselemente. Wenn man eine normale HTML-Datei mit Verweisen in den Hauptrahmen lädt und dann darin auf einen Verweis klickt, wird auch die Zieldatei wieder im Hauptrahmen angezeigt.

Hoffentlich ist das nicht wieder eine Webseite mit Rahmen. Denn dann würde der verfügbare Darstellungsbereich immer kleiner.

Wir brauchen Steuerungen, um den Rahmen beim Verlassen zumindest auflösen zu können, denn wenn wir den Webauftritt mit Rahmen verlassen und einen ganz anderen Webauftritt besuchen wollen, dann können wir die alte Einteilung des Bildschirms nicht mehr brauchen.

Dazu wurden vier spezielle Namen für Ziele definiert. Die Namen beginnen mit einem Unterstrich:

- `_self` im eigenen Rahmen anzeigen (Vorgabewert)
- `_top` im Gesamt-Fenster ohne Rahmen anzeigen (alle framesets werden aufgelöst)
- `_blank` in einem neuen, leeren Fenster anzeigen (wird auch bei unbekanntem Namen benutzt)
- `_parent` in dem Fenster, das bisher den momentanen Frameset angezeigt hat. Die Frames werden aufgelöst. (bei verschachtelten Frames) Bei nicht verschachtelten Rahmen haben damit *_top* und *_parent* die gleiche Wirkung.

Mit *_self* wird das Ziel im eigenen Rahmen/frame angezeigt (also in dem Rahmen, in dem der Verweis steht), mit *_top* der Rahmenaufbau vollständig verlassen. Sollten derzeit verschachtelte Frames angezeigt werden, werden alle aufgelöst.

Immer, wenn Sie auf eine externe Seite verweisen, sollte daher *_top* als Ziel angegeben werden. (Sonst gibt es ein Fenster im Fenster im Fenster ...). Weiter gibt es *_parent*. Dies stellt die gewünschte Seite in der Art dar, die vor dem Aufruf des `<frameset>` gültig war. In verschachtelten Framesets wird also nur der innerste aufgelöst. Die Angabe *_blank* öffnet ein neues Fenster.

Vielleicht erahnen Sie an dieser Stelle, dass man mit der Vielzahl der Möglichkeiten auch gute Chancen hat, die Bedienerführung zumindest unglücklich zu gestalten. Bei jedem Verweis muss man ganz präzise entscheiden, welche Auswirkungen dies auf die ganze Rahmengruppe hat. Es gibt immer wieder unangenehme Beispiele für unpräzises Design im WWW zu finden.

R-EXP04: Auflösen von Rahmen

Schreiben Sie eine neue HTML-Datei mit Rahmen oder ändern Sie ein vorhandene so ab, dass in der Detailnavigationsleiste Einträge stehen, die die vier speziellen Namen verwenden.

Wenn Sie dies möchten, können Sie das Beispiel ausbauen und eine Rahmengruppe (*frameset*) innerhalb des bisherigen Hauptrahmens definieren. Dann kann man auch den *_parent*-Namen zeigen.

In der Lösung begnügen wir uns mit der unverschachtelten Variante.

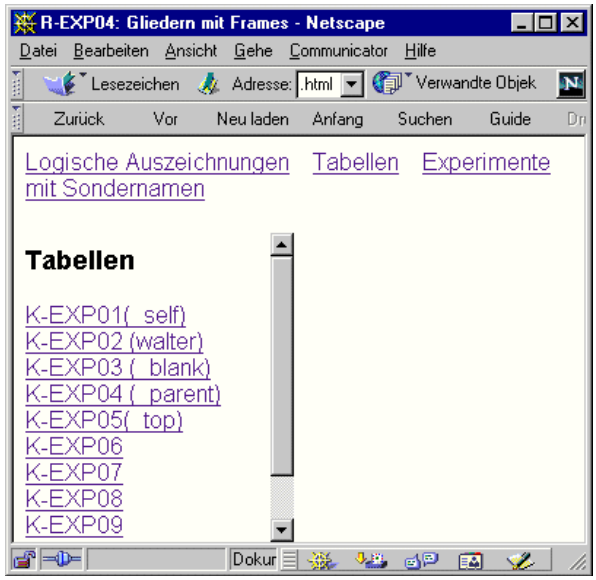

Abbildung 15.7: Experimente mit Sondernamen

15.5 Browser ohne Rahmen

Es gibt sie noch, die Browser, die keine Rahmen anzeigen können. Für diese Browser kann man eine eigene HTML-Seite in die Rahmen-Steuerdatei einfügen.

Dazu wird nach der Rahmengruppe, also nach der Ende-Marke der `<frameset>`-Anweisung die Anweisung `<noframes>` geschrieben, die ihrerseits dann wieder einen vollständigen `<body>`-Block enthalten kann. Meist beschränkt man sich hier auf die knappe Mitteilung, dass keine Rahmen/Frames angezeigt werden können und der Benutzer doch bitte dafür Verständnis haben soll.

Gute Webseiten bieten einen Verweis auf eine Version ohne Rahmen an. Aber das ist doch ein Zusatzaufwand, den nicht alle erbringen können.

R-EXP05: Information für Browser ohne Rahmen

Erweitern Sie eine bestehende Webseite mit Rahmensteuerung um die notwendigen Teile für Browser ohne Rahmen.

Die Lösung kann man vermutlich nur schwer testen, außer man besorgt sich von den Webauftritten der Hersteller einen garantiert alten Browser.

Ob uns da der Text-Browser *Lynx* helfen kann?

```
<frame src= "r-exp03-leer.html" name="detailnavigation" frameborc
<frame src= "r-exp03-leer.html" name="main" frameborder="0">
</frameset>
<noframes>
<body>
Kann leider die Frames nicht anzeigen.<br>
Können Sie sich einen aktuellen Browser besorgen?<br>
</body>
</noframes>

</frameset>
</html>
```

Abbildung 15.8: Angaben für Browser ohne Rahmen

Diese Angaben für Browser ohne die Fähigkeit, Rahmen anzuzeigen, werden Sie im Normalfall selbst nicht sehen. Aber es ist sicher vernünftig, zu anderen höflich zu sein.

15.6 Standardvorgabe des Zielfensters

Es gibt noch eine Möglichkeit, sich das Leben etwas leichter zu machen. Wenn wir die Navigationsleiste schreiben und alle Ziele immer im gleichen Fenster des Rahmensatzes anzeigen wollen, können wir im Kopf der Seite den zumeist gewünschten Fensternamen vorgeben.

Beispiel:

```
<head>
<base target="hauptfenster">
...
</head>
```

Damit können dann die einzelnen Zielangaben in den Verweisen entfallen.

Im nächsten Kapitel

Mit den Strukturierungsmöglichkeiten der Rahmen haben wir ein mächtiges Werkzeug kennen gelernt, das es uns erlaubt, eine große Anzahl von Seiten zu verwalten und dem Benutzer zu erschließen.

Aber wir erreichen auch eine Art Schallmauer, die uns die Grenzen der Sprachmöglichkeiten aufzeigt. Das Beispiel war hier die gleichzeitige Veränderung der Inhalte zweier Fenster.

HTML-Seiten stehen ja nicht alleine. Es sind immer mindestens zwei Beteiligte beim Betrachten einer Webseiten vorhanden: der Server, der die Dateien schickt und sie vielleicht sogar vorher analysiert, und der Browser, der die Seiten anzeigt.

Wir wollen im nächsten Kapitel einen Schritt über die Möglichkeiten der einzelnen HTML-Datei hinausgehen und ein erstes Zusammenspiel zwischen Server und Browser kennen lernen.

15.7 Lösungen

R-EXP01: Grundlegende Seite mit Rahmen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP01: Gliedern mit Frames</title>
</head>
<frameset rows="65,*" >
<frame src="l-exp01.html" name="masternavigation" scrolling="no" framebor-
der="0" noresize>
<frameset cols = "15%,85%" >
<frame src= "l-exp02.html" name="detailnavigation" frameborder="0">
<frame src= "l-exp03.html" name="main" frameborder="0">
</frameset>
</frameset>
</html>
```

R-EXP02: Rahmen mit Verweisen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP02: Gliedern mit Frames</title>
</head>
<frameset rows="65,*" >
<frame src="l-exp01.html" name="masternavigation" scrolling="no" framebor-
der="0" noresize>
<frameset cols = "25%,75%" >
<frame src= "r-exp02-dnav.html" name="detailnavigation" frameborder="0">
<frame src= "l-exp03.html" name="main" frameborder="0">
```

```
</frameset>
</frameset>
</html>
```

R-EXP02-DNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP02-dnav: Gliedern mit Frames</title>
</head>
<body>
<a href="l-exp01.html"  target="main">L-EXP01</a> <br>
<a href="l-exp02.html"  target="main">L-EXP02</a><br>
<a href="l-exp03.html"  target="main">L-EXP03</a><br>
<a href="l-exp04.html"  target="main">L-EXP04</a><br>
<a href="l-exp05.html"  target="main">L-EXP05</a><br>
<a href="l-exp06.html"  target="main">L-EXP06</a><br>
<a href="l-exp07.html"  target="main">L-EXP07</a><br>
</body>
</html>
```

R-EXP03:Unterteilung in Haupt- und Detailnavigation

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP03: Gliedern mit Frames</title>
</head>
<frameset rows="65,*" >
<frame src="r-exp03-mnav.html" name="masternavigation" scrolling="no" fra-
meborder="0" noresize>
<frameset cols = "50%,50%" >
<frame src= "r-exp03-leer.html" name="detailnavigation" frameborder="0">
<frame src= "r-exp03-leer.html" name="main" frameborder="0">
</frameset>
</frameset>
</html>
```

R-EXP03-MNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP04 Master Navigation</title>
</head>
<body>
<a href="r-exp02-dnav.html" target= "detailnavigation">Logische Auszeich-
nungen</a>   
<a href="r-exp03-dnav.html"  target= "detailnavigation">Tabellen</a>
  
</body>
</html>
```

R-EXP03-DNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.73 [en] (WinNT; U) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP03-dnav</title>
</head>
<body>
<h3>
Tabellen</h3>
<a href="k-exp01.html" target="main">K-EXP01</a>
<br><a href="k-exp02.html" target="main">K-EXP02</a>
<br><a href="k-exp03.html" target="main">K-EXP03</a>
<br><a href="k-exp04.html" target="main">K-EXP04</a>
<br><a href="k-exp05.html" target="main">K-EXP05</a>
<br><a href="k-exp06.html" target="main">K-EXP06</a>
<br><a href="k-exp07.html" target="main">K-EXP07</a>
<br><a href="k-exp08.html" target="main">K-EXP08</a>
<br><a href="k-exp09.html" target="main">K-EXP09</a>
<br><a href="k-exp10.html" target="main">K-EXP10</a>
<br><a href="k-exp11.html" target="main">K-EXP11</a>
</body>
</html>
```

R-EXP03-LEER:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.73 [en] (WinNT; U) [Netscape]">
   <title>R-EXP03-leer</title>
</head>
<body>
 
<br> 
</body>
</html>
```

R-EXP04: Auflösen von Rahmen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP04: Gliedern mit Frames</title>
</head>
<frameset rows="65,*" >
<frame src="r-exp04-mnav.html" name="masternavigation" scrolling="no" fra-
meborder="0" noresize>
<frameset cols = "50%,50%" >
<frame src= "r-exp03-leer.html" name="detailnavigation" frameborder="0">
<frame src= "r-exp03-leer.html" name="main" frameborder="0">
</frameset>
</frameset>
</html>
```

R-EXP04-MNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP04 Master Navigation</title>
</head>
<body>
<a href="r-exp02-dnav.html"  target="detailnavigation">Logische Auszeich-
nungen</a>   
```

```
<a href="r-exp03-dnav.html"  target="detailnavigation">Tabellen</a>
  
<a href = "r-exp04-dnav.html" target= "detailnavigation"> Experimente mit
Sondernamen</a>
</body>
</html>
```

R-EXP04-DNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP04-dnav</title>
</head>
<body>
<h3>Tabellen</h3>
<a href="k-exp01.html"  target="_self">K-EXP01(_self)</a> <br>
<a href="k-exp02.html"  target="walter">K-EXP02 (walter)</a> <br>
<a href="k-exp03.html"  target="_blank">K-EXP03 (_blank)</a> <br>
<a href="k-exp04.html"  target="_parent">K-EXP04 (_parent)</a> <br>
<a href="k-exp05.html"  target="_top">K-EXP05(_top)</a> <br>
<a href="k-exp06.html"  target="main">K-EXP06</a> <br>
<a href="k-exp07.html"  target="main">K-EXP07</a> <br>
<a href="k-exp08.html"  target="main">K-EXP08</a> <br>
<a href="k-exp09.html"  target="main">K-EXP09</a> <br>
<a href="k-exp10.html"  target="main">K-EXP10</a> <br>
<a href="k-exp11.html"  target="main">K-EXP11</a> <br>
</body>
</html>
```

R-EXP05: Information für Browser ohne Rahmen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP05: Gliedern mit Frames</title>
</head>
<frameset rows="65,*" >
<frame src="r-exp04-mnav.html" name="masternavigation" scrolling="no" fra-
meborder="0" noresize>
<frameset cols = "50%,50%" >
<frame src= "r-exp03-leer.html" name="detailnavigation" frameborder="0">
```

```
<frame src= "r-exp03-leer.html" name="main" frameborder="0">
</frameset>
<noframes>
<body>
Sie können leider keine Frames anzeigen.<br>
Können Sie sich einen neuen Browser besorgen?
</body>
</frameset>
</html>
```

16 Programme auf dem Webserver

In den vergangenen Kapiteln haben wir in vielen Experimenten die vielen Möglichkeiten und Eigenschaften der Sprache HTML dargestellt. Dabei haben wir HTML-Dateien, Bilder und andere fertige Dateien auf einem Webserver angesprochen.

Alle Dateien waren Datendateien. Darüber hinaus gibt es noch eine zweite große Gruppe: die Programmdateien. Wir können mit HTML-Mitteln auch auf Programmdateien auf den Webservern verweisen.

Man kann sich leicht vorstellen, dass dem jeweiligen Verwalter eines Webservers gar nicht wohl wäre, wenn jeder fremde Benutzer beliebige Programme starten könnte. So gibt es unter Linux/Unix einen Befehl der unwiederbringlich alle Dateien in allen Verzeichnissen auf diesem Rechner löscht. Den sollte man wohl besser nicht der Allgemeinheit zur Verfügung stellen.

Die Benutzung der Programmdateien wird also sinnvollerweise deutlich eingeschränkt. Die meisten einfachen Webauftritte dürfen überhaupt keine Programme benutzen. Manchmal werden einige wenige Programme, wie ein Aufrufzähler oder ein Gästebuch vom Provider zur Verfügung gestellt.

Betrachten wir daher für den Moment einen privaten Webserver oder einen etwas aufwendigeren Webserver, wie er auch zur Unterstützung des Buches zur Verfügung steht.

Auf den meisten Webservern (den Maschinen) gibt es für den HTTP-Server (dem Server-Programm) einen reservierten Bereich im Dateisystem. Und innerhalb dieses reservierten Bereichs existiert ein Unterverzeichnis für alle Daten, die veröffentlicht werden sollen und ein Verzeichnis für Programme, die aus dem Internet startbar gemacht werden sollen. Das Verzeichnis heißt unter Linux/Unix meist CGI-BIN. Der Name leitet sich von *CGI* (common gateway interface / allgemeine Standardschnittstelle) ab. Gemeint ist die Verbindung

zwischen dem jeweiligen HTTP-Server und den damit aufgerufenen Programmen.

Wie fast immer gibt es verschiedene Erweiterungen von kleineren und größeren Firmen. Wir wollen hier bei der allgemeinen und offenen Schnittstelle bleiben.

Statt eines Dateinamens kann nun auch im *href*-Parameter des Verweises ein Programmname auftauchen. Unter manchen Betriebssystemen kann man an der Dateikennung (z.B. *.EXE) erkennen, dass es sich um ein Programm handelt. Unter Linux/Unix bestimmt man mit Hilfe eines Dateiattributes, ob eine Datei ausgeführt werden kann oder nicht. Aber meist kann man am Pfad (/CGI-BIN/) erkennen, dass der Verweis auf ein Programm zeigt.

16.1 Programme und Interaktivität

Wozu sollten wir auf einem fernen Rechner möglicherweise am anderen Ende der Welt ein Programm starten können? Nun, dies ist die Voraussetzung für Interaktivität. Wenn wir in irgendeiner Form Daten bereitstellen, wollen wir vermutlich eine schnelle Antwort darauf erhalten. Das kann eine Bestellung sein oder die Reservierung eines Tickets oder einfach das Klicken mit der Maus auf eine Landkarte.

Ich kenne eine Firma, deren Filialen über ganz Deutschland verteilt liegen. Diese Firma bietet auf ihrer Webseite eine Karte von Deutschland an. Klickt man so ungefähr auf den eigenen Wohnort, dann erscheint als Rückmeldung die Adresse der nächsten Filiale und die Entfernung zum Wohnort. Ich habe das als eine elegante und kundenfreundliche Lösung empfunden.

Allerdings erfordert diese Lösung die Möglichkeit, den Klickpunkt des Benutzers zu erfragen, die Werte zum Server zu schicken, diese dort auszuwerten und eine individuelle Antwort an den Benutzer zurück zu liefern. Das sind eine ganze Menge neue Anforderungen an eine Webseite.

16.2 Serverseitige sensitive Bilder

Benutzen wir als Einstieg in das Thema *Webserver und Programme* das gerade erwähnte Beispiel.

Dazu benötigen wir zuerst wieder ein Bild. Dieses Bild wird als Ganzes sensitiv gemacht. Das unterscheidet dieses Verfahren vom bisher benutzten Verfahren der Zuordnungstabelle in den browserseitigen sensitiven Bildern.

Das Verfahren heißt auf englisch *server side image maps*.

Um ein Bild anklickbar zu machen, genügt es wie bisher auch, das Bild innerhalb eines Verweises anzugeben. Vor und hinter die `<img>`-Anweisung stellen wir die Start- und Ende-Marke des Verweises. Allerdings soll der Verweis auf eine Programmdatei zeigen.

Nun brauchen wir nur noch eine Möglichkeit, unseren Klickpunkt mit seinen Koordinaten an das im Verweis angegebene Programm zu senden. Dazu ergänzen wir die `<img>`-Anweisung um den Parameter *ismap* ohne einen zusätzlichen Wert. Aus unserem Bild ist eine Zuordnungsvorlage geworden. Wir können durch einen Klick eine Information an das Programm liefern, auf das wir im Verweis zeigen. Das Programm wird dann die Zuordnung vornehmen oder eine andere sinnvolle Auswertung treffen.

Das ist das gleiche Verfahren, das wir schon als „Trick" bei den browserseitigen sensitiven Bildern gesehen haben. Der Unterschied ist nur, dass wir nicht, wie im Trick, auf eine beliebige HTML-Datei verweisen, sondern auf ein Programm.

Schreiben wir nun zuerst einmal eine solche Datei.

C-EXP01: Serverseitige sensitive Bilder

Erstellen Sie eine HTML-Datei, in der ein Bild als Vorlage für eine Auswertung auf dem Server verwendet wird. Laden Sie ein Bild eines beliebigen unterstützten Formates. Dieses Bild soll als Zuordnungsvorlage definiert werden.

Das Bild soll als Verweis dienen. Der Verweis soll auf ein Auswerteprogramm zeigen. Sie sehen bei Bewegungen mit der Maus über dem Bild die Koordinaten in der Statuszeile angezeigt. Dies gilt zumindest im heutigen Netscape und Microsoft Browser.

Verweisen Sie bei Bedarf auf das Programm *http://www.walter-herglotz.de/programme/echo_var.php3*.

Eine Frage bleibt wahrscheinlich noch offen: auf welches Programm sollen Sie verweisen? Wenn Sie der Eigentümer des Rechners im Netz sind, dann können Sie das gewünschte Programm in das erwähnte Verzeichnis /CGI-BIN stellen, wenn Sie es geschrieben haben und natürlich in Ihren Webseiten darauf verweisen. Sollten Sie – was wahrscheinlich die Regel ist – keinen eigenen Webserver betreiben, dann müssten Sie die Hilfe eines geeigneten Webservers in Anspruch nehmen.

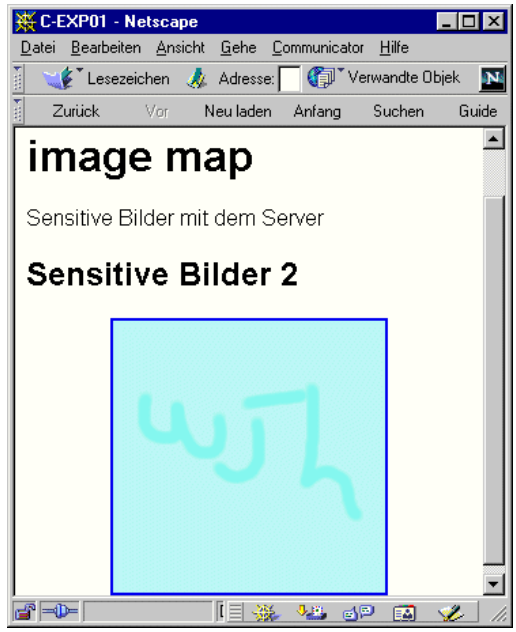

Abbildung 16.1: Sensitive Bilder mit Serverunterstützung

Um Ihnen das Experimentieren zu erleichtern, habe ich ein kleines Programm geschrieben und auf dem Supportserver für das Buch zur Verfügung gestellt. Sie können es gerne benutzen. Aus technischen Gründen liegt es allerdings nicht in dem besprochenen Verzeichnis CGI-BIN sondern unter PROGRAMME. Der Grund ist, dass es ein Script der Sprache PHP3 zur Generierung dynamischer Webseiten ist. Und Scripte, die Webseiten dynamisch erzeugen, können überall liegen.

Für den Moment können Sie als Testprogramm die folgende Adresse benutzen:

http://www.walter-herglotz.de/programme/echo_var.php3
(bitte alles klein schreiben).

Bitte haben Sie Verständnis, wenn sich diese Adresse einmal ändern sollte. Informationen erhalten Sie dann sehr wahrscheinlich direkt auf der Homepage der Webadresse.

Das Echo-Programm wird Ihnen die wichtigsten Informationen zurückmelden, die Sie bei einem Experiment an den Server übermittelt haben.

Für das Beispiel der serverseitigen Bildauswertung sind dies insbesondere die Koordinaten des Klickpunktes im Bild. Auf den Punkt genau könnte das angesprochene Programm auf Ihre Eingabe reagieren.

Was aber ein reales Programm mit den Informationen anfängt, die es vom Webserver erhält, ist natürlich seine Angelegenheit. In unserem Fall sehen Sie nur die Rückmeldung der Koordinaten.

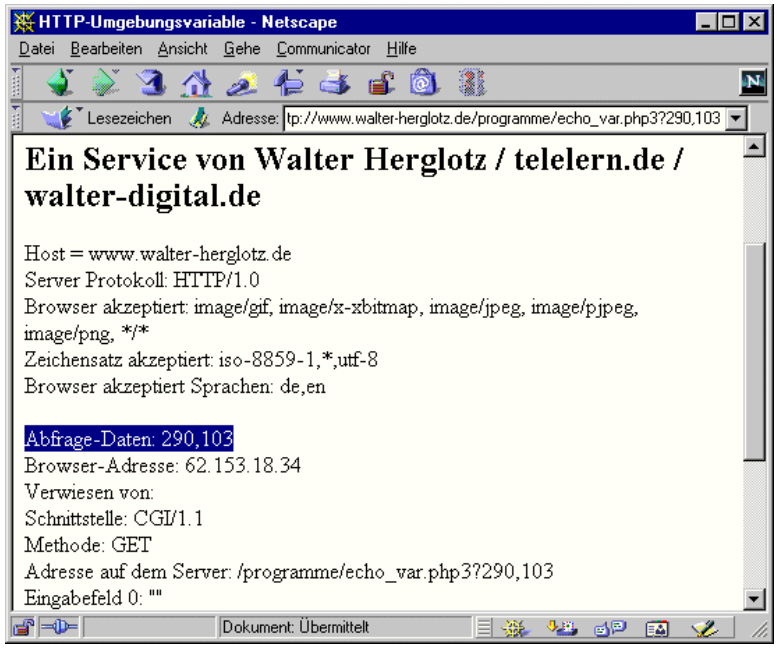

Abbildung 16.2: Rückmeldung des Echo-Programmes

Im Bild sehen Sie einen Teil der Rückmeldung. Letztlich sind dies die Inhalte einiger Umgebungs- und Programmvariablen. Im ersten Teil finden Sie Informationen zum Server und zum Browser. Der Browser meldet, welche Bildformate, Sprachen und Zeichensätze er unterstützt.

Danach kommt die für uns wichtige Information der Abfragedaten. Es sind im Bild „290,103“. Es ist leicht zu erraten. Das sind die Koordinaten des Klicks. Etwas weiter unten erkennen wir die Methode: hier *get*. Aber das können Sie auch selbst ausprobieren.

Im nächsten Kapitel

Mit den anklickbaren Bildern haben wir zum ersten Mal nicht auf Daten sondern auf Programme gezeigt. Hinter dieser Möglichkeit verbirgt sich die ganze Vielfalt der Interaktivität, die mit Browsern möglich ist.

Im nächsten Kapitel betrachten wir einmal alle vorhandenen Eingabeanweisungen in HTML.

16.3 Lösungen

C-EXP01: Serverseitige sensitive Bilder

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>C-EXP01</title>
</head>
<body>
<h1>
SSIM - server side image map</h1>
Sensitive Bilder mit dem Server
<h2>
Sensitive Bilder 2</h2>
<center><a href="http://www.walter-herglotz.de/programme/
echo_var.php3"><img ISMAP SRC="../exp-bilder/hintergrund1.gif" NOSAVE
height=200 width=200></a></center>
</body>
</html>
```

17 Interaktivität und Formulare

Der erste Traum der WWW-Entwickler ist über alle Maßen in Erfüllung gegangen. Sie wollten dem Informationsverlust durch Personalwechsel und durch abgekapselte Strukturen mit Hilfe einer weitgehend offenen Informationspolitik entgegenwirken. Ihr Mittel war die technische Veröffentlichung und die Verbindung untereinander durch Verweise. Wir wissen längst, was daraus geworden ist. Wie viele Millionen von Webseiten uns in der ganzen Welt zur Verfügung stehen, können wir nur erahnen.

Für jeden, der an irgendetwas interessiert ist, bietet das WWW und die anderen Dienste des Internets eine gigantische Fülle an wertvollen Informationen. Allerdings wird niemand ohne eigenes Interesse lernen, die Spreu vom Weizen zu trennen.

Interessieren Sie Sprachen? Hunderte von Zeitungen in vielen Sprachen sind nur einen Mausklick entfernt. Und es sind auch die Sprachen aller unserer Nachbarländer. Wäre es nicht toll, viele von Ihnen zu besuchen und vielleicht sogar die eine oder andere Seite in der jeweiligen Originalsprache zu lesen?

Mit der Möglichkeit im Netz, die Welt zu entdecken, haben wir nur einen Aspekt erfasst. Gut, es ist viel schneller im Zugriff, als die Zeitungskioske in den großen Bahnhöfen. Aber vielleicht bemüht sich jemand, der eine französische Zeitung am Kiosk gekauft hat, mehr um den Inhalt, als ein flüchtiger Besucher einer französischen Zeitungswebseite.

Was Kioske und Büchereien nicht so einfach leisten können, ist der umgekehrte Weg. Nahezu jeder, der surft, kann auch persönlich für die Welt publizieren.

Die nächste große Eigenschaft ist die Fähigkeit, mit HTML-Seiten Interaktivität ins Haus zu bringen. Diese Fähigkeit führt letztlich dazu, dass ein Browser in der Lage ist, als Schnittstelle zwischen dem Benutzer auf seinem Rechner und einem Programm auf einem ganz anderen Rechner im Netz zu dienen.

Skeptiker werden vielleicht befürchten, dass die Zeit der zentralen Großrechner wieder kommt. Vielleicht haben sie sogar recht.

17.1 Formulare

Interaktive Seiten benutzen Formulare für alle Benutzereingaben. Ein Formular ist eine Zusammenstellung von bisher bekannten HTML-Elementen wie einfacher Text, Tabellen oder Bilder zusammen mit Eingabemöglichkeiten für den Benutzer.

Die Eingabeelemente in HTML nutzen die gleichen Eingabemöglichkeiten wie auch andere graphische Benutzeroberflächen. Dazu zählen verschiedene Eingabefelder, Auswahlmöglichkeiten (1 aus n), Listenauswahl und Auswahlfelder.

Die Formularanweisung heißt `<form>`.

Wir wollen hier einmal eine grundlegende Seite mit Formularen besprechen. Für die Interaktivität brauchen wir nun wieder sowohl unseren Browser, als auch einen Server mit der Erlaubnis, Programme abzulegen. Und natürlich müssten wir auch die Auswerteprogramme schreiben. Aber das geht deutlich über die Intention dieses Buches hinaus, Ihnen ein solides Grundverständnis von HTML und Spaß an der Praxis zu vermitteln.

Bleiben wir also hier bei der Darstellung der Formular-Elemente und betrachten uns das Ergebnis, wenn Sie es wünschen, mit dem Echo-Programm auf dem Buchserver *http://www.walter-herglotz.de/programme/echo_var.php3*. Wenn Sie selbst einen Server zur Verfügung haben, z.B. auf einer Linux-Maschine, stehen Ihnen natürlich alle Programmiermöglichkeiten offen.

Zu jedem Formular gibt man in einem URL eine Aktion an. Diese Aktion erhält dann das Ergebnis des Formulars zugestellt. Als Aktion trägt man in den allermeisten Fällen den Namen eines Programmes oder eines Scriptes ein. In unserem Testfall ist das das Echo-Programm auf dem Buchserver. Für große Anwendungen wird als Ziel meist ein Programm dienen. Ansonsten genügen Scripten.

Der Unterschied ist für den Autor von Webseiten erst einmal belanglos. Programme können auf den Server direkt gestartet werden., Skripte benötigen einen laufenden Interpreter, der das Skript liest und Schritt für Schritt ausführt. Wichtig wird der Unterschied erst, wenn Sie Tausende von Anfragen in der Stunde auf Ihrer Webseite erwarten.

Weiter müssen wir aus einer von zwei Methoden wählen, wie dem Zielprogramm die eingegebenen Daten übergeben werden. Es gibt *get* und *post*.

Die *get*-Methode stellt dem aufgerufenen Programm die Daten als Inhalt einer Umgebungsvariablen zur Verfügung. *post* kann Daten über den Standardeingang des aufgerufenen Programmes übernehmen. Nutzen Sie daher *post* für größere Datenmengen. Falls Sie nicht selbst programmieren, dann verwenden Sie einfach das Echo-Programm und die Methode *post*.

Schließlich gibt es noch den Parameter *enctype* (encryption type / Verschlüsselungsart). Der Name des Parameters geht weit über das hinaus, was tatsächlich passiert. Man kann damit festlegen, in welcher Darstellung die Daten übertragen werden, nicht aber eine Verschlüsselung. Der Wert des Parameters besteht aus einem MIME-Type (MIME-Typ). Man kann mit ihm die Kodierung des Formularinhaltes steuern. Im Normalfall werden die Werte in einer 7-Bit-Darstellung mit Ersetzungszeichen übertragen.

Diese Darstellung heißt *application/x-www-form-urlencoded*. Diese Darstellung ist für eine Übertragung mit *get* notwendig. Im Falle einer *post*-Übertragung kann man auch als Wert *text/plain* benutzen. Dann werden die Daten ohne Kodierung übertragen. Zur Übertragung von Dateien benötigen wir noch einen weiteren Darstellungstyp: *multipart/form-data*.

17.2 Aufbau von Formularen

Formulare bestehen aus Grundkomponenten der Fensterprogrammierung. Dazu gehören Eingabefelder, Auswahlfelder oder Möglichkeiten zum Ankreuzen. Schließlich gibt es noch die Möglichkeit, einen umfangreichen Text einzugeben.

Das Eingabe-Element `<input>` besteht immer aus der Art der Eingabe, den dazu notwendigen Parametern, einen Namen für diese Eingabe und einen Ergebniswert. Das auswertende Programm erhält dann die paarweise Angabe `Name="Ergebniswert"` und kann so die einzelnen Teile eines Formulars auseinander halten.

Zwischen den eigentlichen Eingabefeldern kann beliebiger Text stehen. Auch können die Eingabefelder z.B. über Tabellenelemente verteilt werden.

17.3 Eingabe-Möglichkeiten

17.3.1 1 aus n - Auswahl

```
Sauerbraten <input type="radio" name="eingabe1" value="Braten" checked>
Nudelauflauf <input type="radio" name="eingabe1" value="Nudeln" >
Halbes Huhn<input type="radio" name="eingabe1" value="Huhn" >
Eieromlett <input type="radio" name="eingabe1" value="Eier" >
...
```

Im Beispiel werden die Namen von verschiedenen Gerichten am Bildschirm angezeigt. Hinter jedem Namen steht eine Eingabeanweisung. Der Browser wird logisch alle Eingaben mit gleichem Namen zu einer Gruppe zusammenfassen. Hier wurde der Gruppenname *eingabe1* verwendet. Unser Echo-Programm kann ja nicht erraten, welche Namen Sie möglicherweise benutzen wollen. Daher sollten Sie nach Möglichkeit die Namen des Echo-Programms benutzten. Es gibt alle Eingaben mit den Namen *eingabe0* bis *eingabe9* zurück.

Für jede einzelne Eingabemöglichkeit schreibt man eine `<input>`-Anweisung. Die `<input>`-Anweisung gehört zu den wenigen HTML-Anweisungen, die keine Ende-Marke benutzen.

Im Beispiel steht außerhalb der Anweisung normaler Text. Der wird wie gewohnt am Bildschirm angezeigt und kann dem Benutzer mitteilen, was er denn nun hier eingeben soll.

Hier handelt es sich um eine 1 aus n-Auswahl. Sie erkennen dies am gleichen Namen für alle Eingaben und natürlich am Eingabetyp *radio*. Die guten alten Radios hatten ein kleines Tastenklavier zur Auswahl des Frequenzbandes. Drückte man auf eine Taste, sprang die bisher selektierte heraus. Es kann immer nur eine Taste aktiv sein.

Das Ereignis wird dann in Form eines Namenwertepaares übertragen. Beispiel: *eingabe1="Huhn"*. Mit der Parameter *checked* (aktiviert) gibt man an, welcher Wert als Vorgabe angezeigt und benutzt werden soll. Die Vorgabe kann fehlen.

Noch ein Tipp zur Gestaltung. Solche Abfragen wird man am einfachsten in Tabellen verpacken, um eine ansprechende Seite zu erhalten.

G-EXP01: Formular mit einer 1 aus n-Auswahl

Schreiben Sie eine HTML-Seite mit einem Formular. Als Übertragungsmethode wählen Sie *post*. Eine bestimmte Darstellungsart braucht nicht angegeben zu werden.

Probieren Sie das Verhalten des Formulares aus, wenn keine Default-Einstellung gewählt wurde (mit *checked*).

Was passiert beim ersten Anzeigen der Auswahl, was passiert beim Zurücksetzen mit dem *reset*-Element und was wird übertragen, wenn keinerlei Auswahl getroffen wurde?

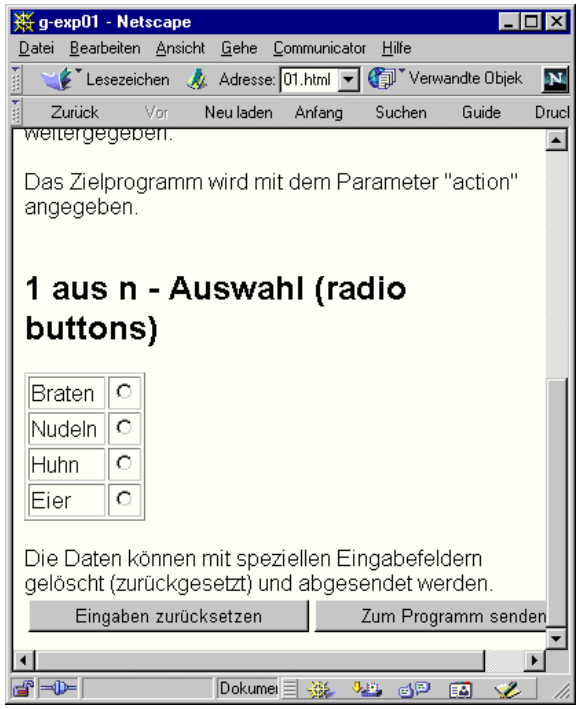

Abbildung 17.1: Radio-Box (1 aus n)

17.3.2 n aus m - Auswahl

```
<UL>
<LI>Wein<input type="checkbox" name="eingabe2"> </LI>
<LI>Bier<input type="checkbox" name="eingabe3"> </LI>
<LI>Wasser<input type="checkbox" name="eingabe4"> </LI>
</UL>
```

Diese Eingabemöglichkeit benutzt einzelne kleine Markierungsboxen, die man getrennt anklicken kann. Mit dem Typ *checkbox* werden einige (m) Auswahlmöglichkeiten angeboten. Der Benutzer kann dann einzelne oder auch mehrere (oder alle) auswählen. Meist würde man bessere Namen verwenden (z.B. *wein*, *bier*, *wasser*). Nur für das Echo-Programm wurden die wenig aussagekräftigen Namen *eingabe2* etc. benutzt.

G-EXP02: Einzelauswahl (n aus m)

Schreiben Sie eine HTML-Datei mit einem einfachen Formular. Es sollen fünf Getränke auswählbar sein. Das Formular soll wieder mit *post* und ohne spezielle Darstellung an das Echo-Programm gesendet werden.

Beim Betrachten im Browser wählen Sie Wein und Wasser (oder eine andere Kombination Ihrer Wahl) aus und schicken das Formular ab.

Was wird übertragen, was wird nicht übertragen?

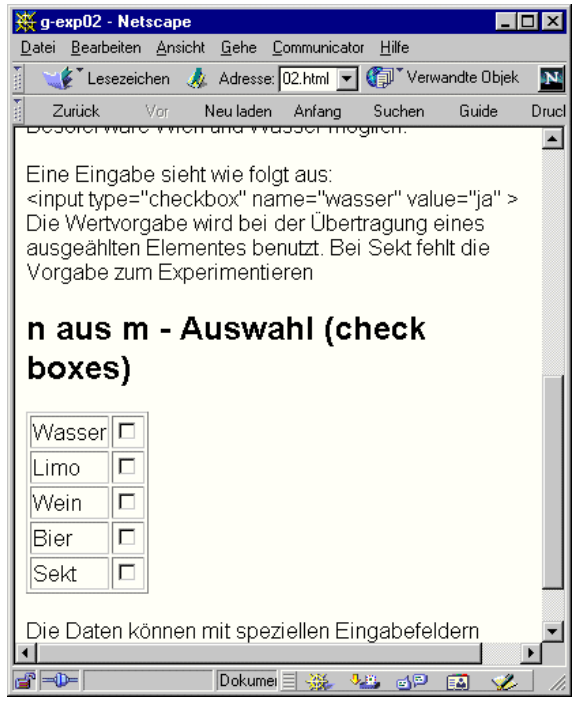

Abbildung 17.2: Einzeln auswählbar (n aus m)

17.3.3 Texteingaben

Für Texteingaben mit kurzen einzeiligen Texten (Name etc.) gibt es Textfelder.

```
<input type="text" name="eingabe5" value="walter">
```

Hier wird der Browser ein Eingabefeld öffnen und bereits mit der Standardvorgabe (hier: „walter“) vorbesetzen. Im Normalfall wird die Standardvorgabe selten benutzt.

Für Texteingaben eignen sich noch zwei Parameter. Der eine ist *maxlength* (maximale Eingabelänge). Hier kann man angeben, wie viele Zeichen der Benutzer sinnvollerweise angeben kann. Denken Sie als Beispiel an eine Eingabemaske für eine Datenbank. Es macht keinen Sinn, mehr Zeichen zu holen, als in der Datenbank gespeichert werden können.

Mit dem Parameter *size* (Größe) kann man die sichtbare Größe des Eingabefeldes angeben. Ob dies allerdings immer korrekt vom Browser über alle Schriftarten und Schriftgrößen eingehalten werden kann, ist fraglich. Im Zweifel geben Sie lieber ein paar Zeichen mehr für *size* an.

Ein weiterer Typ kann hier *password* statt *text* sein. Dann werden beim Eingeben nur Sternchen angezeigt. Der angegebene Wert dient wieder zur Vorbesetzung. Allerdings gilt hier, wie für alle anderen Eingaben auch, dass keine Verschlüsselung erfolgt. Ihre Daten laufen ungesichert über das Netz. Nur wenn die Seiten generell über ein Verschlüsselungsverfahren übertragen werden, kann eine gewisse Sicherheit geboten werden.

G-EXP03: Eingabe mit einzeiligen Textfeldern

Schreiben Sie eine HTML-Datei mit einem Formular, das vier einzeilige Texteingaben erwartet. Fragen Sie nach Namen, Vornamen, Anmeldenamen (login) und Passwort. Das Passwort soll nicht angezeigt werden. Auf Wertvorgaben sollten Sie besser verzichten.

Für längere mehrzeilige Texte eignet sich besser ein großes Textfeld. Es fällt aus dem Rahmen der bisher besprochenen `<input>`-Anweisungen, wird aber ansonsten sehr ähnlich benutzt und behandelt.

Anders als bisher ist insbesondere der Umgang mit einer Textvorgabe. Es fehlt der *value*-Parameter. Stattdessen können Sie zwischen der Start-Marke und der Ende-Marke den gewünschten Vorbesetzungstext angeben oder auch weglassen. Wegen dieser Variante der Text-

vorgabe erwartet die `<textarea>`-Anweisung auch die schließende Marke.

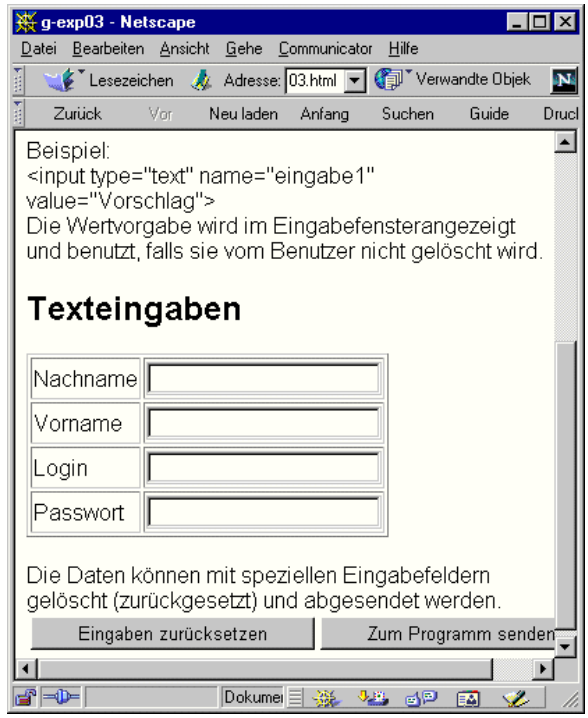

Abbildung 17.3: Einzeilige Texteingaben

```
<textarea rows="10" cols="60" name="eingabe7"></textarea>
```

Hier können Sie einen kleinen Roman eingeben. Die Parameter legen die Größe des Anzeigebereiches fest [hier: 10 rows (Zeilen) und 60 cols (columns/Spalten)]. Trotz der doch sehr großen Eingabemöglichkeiten steht der *maxlength*-Parameter leider nicht zur Verfügung.

G-EXP04: Formular mit mehrzeiligem Eingabefeld

Schreiben Sie eine HTML-Datei mit einem Formular, dass ein mehrzeiliges Eingabefeld beinhaltet. Da die eingegebene Datenmenge evtl. größer sein kann, als es manche Server mit der *get*-Methode vertragen, sollten Sie hier unbedingt mit *post* arbeiten.

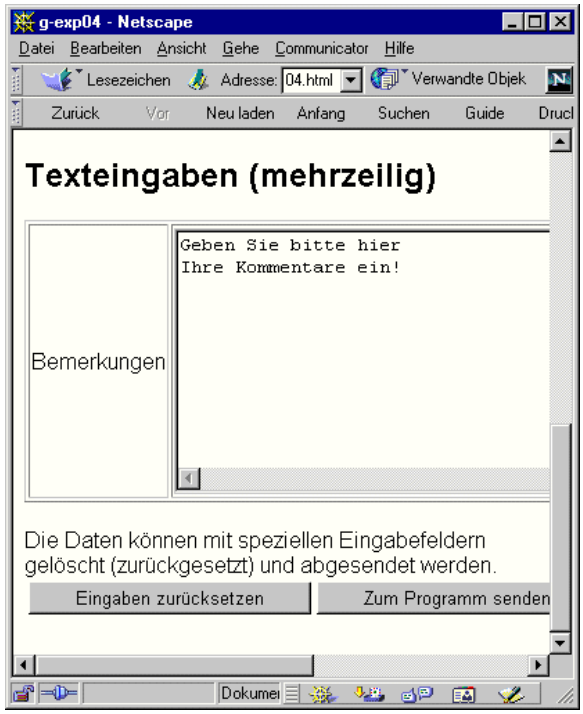

Abbildung 17.4: Mehrzeilige Eingaben

17.3.4 Auswahlboxen

Benutzer einer graphischen Oberfläche sind es gewohnt, dass ihnen bei Entscheidungen eine Vorgabeliste präsentiert wird, aus der sie auswählen. Diese Listen kann man mit Auswahlboxen realisieren. Dabei gibt es zwei Stile. Der erste Stil zeigt dabei einen Teil der Liste an, der zweite nur genau einen Eintrag.

Wohin wollen Sie denn?

```
<select name="eingabe8" size="4">
<option>Shanghai</option>
<option>Wuxi</option>
<option>Hongkong</option>
<option>Singapore</option>
<option>Havanna</option>
<option>Prag</option>
<option>Pilsen</option>
</select>
```

G-EXP05: Mehrfach-Auswahlbox

Schreiben Sie eine HTML-Datei mit einer Mehrfach-Auswahl-Box. Dabei stellen Sie entweder eine Frage Ihrer Wahl mit den gewünschten Optionen oder bilden unser kleines Reiseziel-Beispiel nach.

Der *size*-Parameter muss hier größer als *1* sein.

Für den Fall, dass der Benutzer gleichzeitig mehrere Optionen aktivieren können soll, geben Sie zusätzlich in der `<select>`-Anweisung den Parameter *multiple* ohne weiteren Wert an.

Die beiden Stile können durch die Angabe der Anzeigegröße *size* gesteuert werden. Der Wert 1 gibt eine einzeilige Auswahl an, ein anderer Wert zeigt dann die entsprechende Anzahl von Zeilen zur Auswahl.

Im Falle einer Mehrfachanzeige kann man noch entscheiden, ob der Benutzer die Möglichkeit haben soll, mehr als einen Eintrag auszuwählen. Dazu dient der zusätzliche Parameter *multiple* (mehrfach). Wie der Benutzer hier mehrere Einträge selektiert ist abhängig vom Betriebssystem. Versuchen Sie dazu gleichzeitig die `Command`-Taste und einen passenden Mausklick einzugeben, wenn Sie vor einem Mac sitzen oder nutzen Sie die Kombination mit `Strg` und der Maustaste.

G-EXP06: Einfach-Auswahlbox

Ändern Sie das vorhergegangene Beispiel ab. Die Größe soll nur noch *1* betragen und ein evtl. vorhandener *multiple*-Parameter sollte entfernt werden.

Schreiben Sie also eine einzeilige Auswahlbox.

Mit Hilfe der einzeiligen Auswahl kann man praktisch den gleichen Effekt erzielen, wie mit Radio-Boxen (1 aus n). Es ist sicher eine persönliche Geschmacksfrage, welche Variante man benutzt.

Es gibt natürlich Unterschiede. Die Standardvorgabe ist bei der einzeiligen Auswahlbox immer der erste angezeigte Eintrag. Deshalb sieht man häufig einzeilige Auswahlboxen, die einen speziellen Eintrag an den Anfang setzen: z.B. BITTE AUSWÄHLEN oder SELECT. Der Sinn ist dabei herauszufinden, ob der Benutzer der Seite tatsächlich gewählt hat.

Abbildung 17.5: Einzeilige Auswahl

17.4 Formular abschicken

Hat der Benutzer das Formular fertig ausgefüllt, dann kann er sein Ergebnis zum Server schicken. Hierzu gibt es mehrere Möglichkeiten. Die älteste und immer noch gängigste ist ein einfacher Knopf (engl. button). Klickt der Benutzer darauf, wird der Inhalt des Formulars zu dem Programm geschickt, dessen URL im *action*-Parameter angegeben wurde.

Zum Server schicken:

```
<input type="submit" name="eingabe9" value="Absenden">
```

Der *name*-Parameter kann dabei entfallen. Will man aber noch eine Zusatzinformation beim Benutzer abholen, könnte man mehrere Sendeknöpfe in das Formular einbauen und am Namen entscheiden, welcher Sendeknopf gedrückt wurde. Das kommt eher selten vor.

Der Vorgabewert hat hier eine andere Funktion. Er bestimmt die Beschriftung auf dem Sendeknopf. Wenn Sie eine deutsche Beschriftung wünschen, werden Sie den Vorgabewert in *value* immer angeben müssen.

Formular löschen:

```
<input type="reset" value= "Formular löschen">
```

Meist bietet man dem Benutzer noch eine Möglichkeit, grobe Tippfehler dadurch zu beheben, dass er das ganze Formular löscht. Allerdings bedeutet hier LÖSCHEN nicht unbedingt alle Felder zu leeren, sondern sie auf den Anfangswert zu setzen. Hat man als Beispiel in einem Auswahlfeld eine Vorgabe gemacht (*checked*), dann wird die Eingabe des Benutzers verworfen und wieder die Vorgabe aktiviert.

In den Bildern zu den einzelnen Experimenten waren die beiden Knöpfe zum Senden und Zurücksetzen jeweils schon gezeigt.

17.5 Formulare mit nur einer Eingabe

Ein Sonderfall liegt vor, wenn im ganzen Formular nur eine einzige einzeilige Texteingabe vorhanden ist. Hier würde der Browser das Formular bereits nach dem Textabschluss mit ENTER/RETURN/EINGABE-Taste senden.

G-EXP07: Formular mit einzelner Texteingabe

Schreiben Sie eine HTML-Datei mit einem Formular, das nur eine einzeilige Texteingabe benutzt.

Beobachten Sie das Verhalten nach der Eingabe der ENTER/RETURN/EINGABE-Taste.

Wo werden solche Eingabeformulare benutzt?

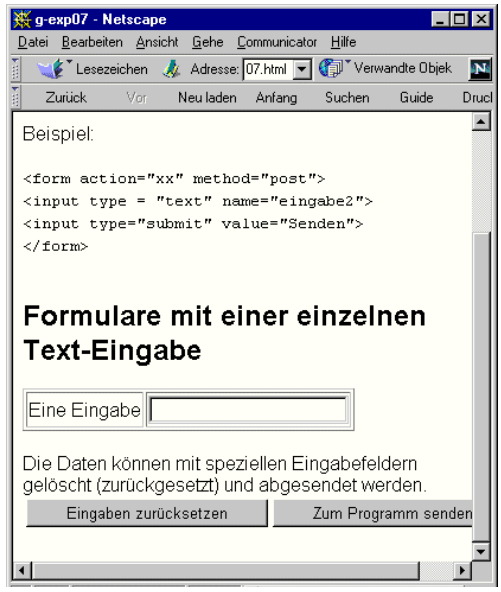

Abbildung 17.6: Formulare mit einer Texteingabe

17.6 Vier abschließende Varianten

Inzwischen haben wir die wichtigsten Eingabemöglichkeiten kennen gelernt. Es bleiben noch vier interessante Varianten zu besprechen.

Die erste ist die einfachste. Man kann den Browser anweisen, selbst einen graphischen Knopf zu erzeugen. Diese Knöpfe werden jedoch nicht mit HTML-Mitteln, sondern mit Hilfe einer eingebauten Programmierung abgefragt. Daher werden wir diese Variante erst ein wenig später im Kapitel über Javascript besprechen.

Die zweite Variante ist im ersten Moment vielleicht etwas mysteriös. Es gibt den Typ *hidden* – auf Deutsch *verborgen*. Sind es nicht eigenartige Eingabefelder, die im Verborgenen existieren? Sie haben aber einen Namen (*name*) und einen passenden Wert (*value*). Meist werden diese Felder als Bindeglied zwischen verschiedenen Seiten einer größeren Präsentation verwendet. Man spricht auch von einer Sitzungsidentifikation (engl. session id).

Gerade bei neueren datenbankgestützten Webauftritten werden Sie diese verborgenen `<input>`-Anweisungen finden. Hier werden die einzelnen Seiten automatisch per Programm generiert. Alle Informationen, die die Verwaltung der Webseiten auf dem Server benötigt, werden in den verborgenen Namen/Werte-Paaren sicher über alle Seiten eines Besuches auf einer Webseite mitgeführt. Da sie aber verborgen sind, kann man sie nicht in einem Bild zeigen.

17.7 Bilder als Sendeknopf

Die dritte Variante nutzt eine Graphik als Sendeknopf. Statt eines konventionellen Sendeknopfs kann man auch eine Graphik einsetzen. Die Graphik wirkt dabei wie ein sensitives Bild. Neben der Funktion, ein Formular abzusenden, übermittelt der Klick auf eine Sendegraphik an den Server zusätzlich die Koordinaten des Klickpunktes.

Die Eingabeanweisung `<input>` wird dazu als *type*-Parameter den Wert *image* verwenden. Dieser Parameter impliziert den *submit*-Wert. Nun brauchen wir nur noch die Angabe des Bildes, dass wir wie bei der `<img>`-Anweisung mit Hilfe des *src*-Parameters angeben.

Die Eingabe erhält, wie andere Eingabefelder auch, einen Namen. Aus dem Namen werden zwei neue Namen generiert. Der Browser hängt an den Stammnamen einmal ein *.x* und einmal ein *.y* an. Mit diesen Namen werden dann die Klick-Koordinaten transportiert.

Das Serverprogramm kann also auf den Punkt genau feststellen, wo der Benutzer geklickt hat.

Vielleicht wird in Zukunft diese Variante der `<input>`-Anweisung die bisherige Steuerung der serverseitigen sensitiven Bilder mit der `<img>`-Anweisung ablösen.

G-EXP08: Graphischer Sendeknopf

Erstellen Sie ein Formular. Anstelle der `<input type="submit">`- Anweisungen verwenden Sie nun als Typ-Parameter *image*. Vergessen Sie den Namen nicht, denn den braucht der Browser, um die Koordinaten zu übermitteln. Das Bild laden Sie wie gewohnt mit der *src*-Anweisung.

Wenn Sie mit *action* das Echo-Programm des Buchservers nutzen, dann sehen Sie zumindest in den Abfragedaten die Koordinatenwerte angezeigt.

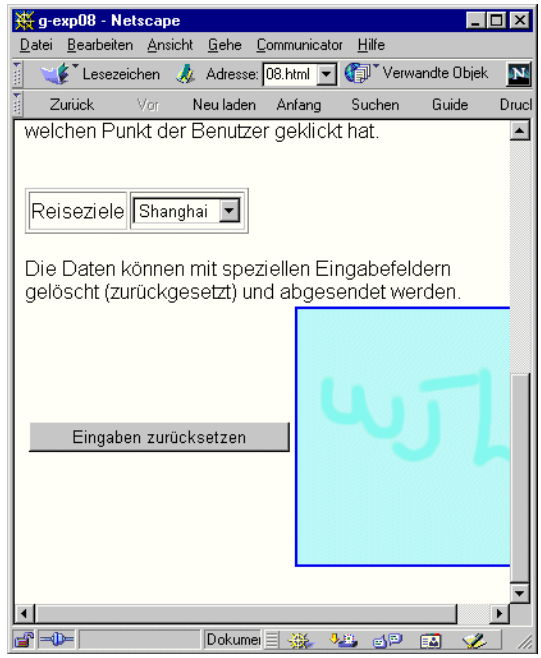

Abbildung 17.7: Graphischer Sendeknopf

17.8 Dateien zum Server schicken

Die vierte noch zu besprechende Variante ist sicher die spektakulärste. Man kann neben den einfachen Texteingaben auch ganze Dateien zum Server befördern. Damit können Graphik-Dateien oder auch herstellerspezifische Dateien vom Benutzer zum Webserver geschickt werden.

Damit kommen wir einem Traum der Webentwickler einen großen Schritt näher. Denn sie wollten ein sehr dynamisches und unkontrolliert wachsendes Netz. Erschrecken Sie nicht. Unkontrolliert bedeutet nicht chaotisch. Denn die neuen Dateien der Benutzer sollten ja in das Netz der Verweise, das WWW, einbezogen werden können. Noch sind die Fähigkeiten der Webserver, automatisch zu publizieren, sehr gering. Aber auch beim Publizieren tut sich etwas.

Kommen wir zuerst zum Datei-Upload zurück (upload: vom Benutzer zum Server).

In der `<form>`-Anweisung benötigen wir einen anderen MIME-Typ, der für die Übertragung von binären Dateien geeignet ist. Darstellungstyp muss nun *multipart/form-data* sein. Die einzelnen Eingaben des Formulars werden damit getrennt übertragen. Die Methode wird natürlich *post* sein, denn *get* kann ja nur kleine Datenmengen bearbeiten.

Die `<input>`-Anweisung verwendet dann innerhalb des Formulars den Wert *file* für den *type*-Parameter. Da es sich erst einmal um ein Texteingabefeld handelt, in dem der Benutzer den Namen evtl. zusammen mit dem lokalen Pfad eingibt oder mit Betriebssystemhilfe sucht, kann der Autor hier auch die üblichen Parameter für Texteingaben wie *size* und *maxlength* verwenden.

Der Browser wird dann die Datei laden und an das Programm übermitteln, das im *action*-Parameter der `<form>`-Anweisung angegeben ist.

Was allerdings das Zielprogramm damit macht, ist seine Sache.

G-EXP09: Datei-Upload Steuerung

Schreiben Sie ein Formular, das eine (sehr kleine) Datei zu einem Server schicken kann. Falls Sie den Buchserver nutzen, denken Sie daran, dass Sie sehr vorsichtig mit der Dateigröße sind.

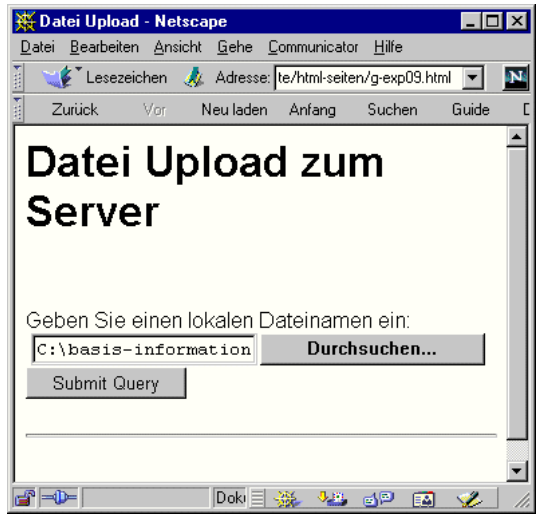

Abbildung 17.8: Datei Upload

17.9 Menüs

Das einzige, was wir vielleicht an dieser Stelle an Eingabemöglichkeiten vermissen, sind Menüs. Diese Menüs lassen sich aber durch Verweise nachbilden.

Dazu kann man beispielweise die Formulare innerhalb einer Rahmengruppe mit den beiden Navigationsleisten nutzen, die wir ja schon diskutiert haben.

17.10 ISINDEX – die erste Suchabfrage

Nachdem wir die heute übliche Methode mit der `<form>`- und der `<input>`-Anweisung kennen gelernt und gesehen haben, wie Benutzereingaben abgeholt und weitergeleitet werden, können wir uns noch einer historischen Anweisung widmen. Es ist vermutlich ganz spannend, hier die ungeheuere Entwicklung zu sehen, die die Sprache HTML in den wenigen Jahren ihrer Existenz genommen hat.

Die `<isindex>`-Anweisung war die erste Möglichkeit, zumindest eine gewisse Interaktivität einzuführen. Obwohl die Anweisung längst ausgedient hat und durch Formulare ersetzt wurde, wollen wir uns doch diese historische Benutzerabfrage einmal ansehen.

Die `<isindex>`-Anweisung steht an einer ungewöhnlichen Stelle – sie steht im Kopf der Seite (innerhalb der `<head>`-Anweisung).

Der Browser reagiert auf eine `<isindex>`-Anweisung durch die Anzeige eines kleinen Eingabefensters, das eingerahmt wird durch je einen graphischen Strich (`<hr>`) über und unter dem Eingabefenster

Die eigentliche Eingabeaufforderung kann durch einen Parameter *prompt* und einem passenden Text als Wert vorgegeben werden. Fehlt die eigene Vorgabe, wird der Browser einen Texte wie *This is a searchable index. Enter search keywords: / Dies ist ein durchsuchbarer Index. Geben Sie die Suchbegriffe ein:* anzeigen.

Sollte eine nähere Bedienungsanleitung notwendig sein, steht der gesamte `<body>`-Bereich zur Verfügung.

Hat nun der Benutzer einen oder mehrere Begriffe in das Eingabefeld eingegeben, dann muss diese Eingabe zur weiteren Verarbeitung an ein Programm auf dem Webserver übergeben werden.

Da die `<isindex>`-Anweisung keine Möglichkeit bietet, das Zielprogramm anzugeben, dem die eingegebenen Daten übermittelt werden sollen, ist vereinbart, dass die Daten einfach zurückgegeben werden.

Damit muss aber auch schon die HTML-Seite von einem Programm generiert werden, den nur dann macht es Sinn, die Eingabe an die Adresse der Seite wieder abzuliefern.

Natürlich kann man zu Testzwecken eine Seite mit `<isindex>`-Anweisung schreiben und von der lokalen Festplatte im Browser aufrufen. Nur werden wir damit keine Reaktion bewirken.

Dies war lange Zeit die einzige, dynamisch erzeugte Art von Seiten im WWW.

Ein solches Programm kann sehr einfach gebaut sein. Für Anwender von Linux kann es auch ein Script sein.

Das Programm wird daher zweimal aufgerufen: das erste Mal erzeugt das Programm die HTML-Seite mit der `<isindex>`-Anweisung, das zweite Mal wertet das Programm die erhaltenen Eingaben aus.

Die Eingaben werden in einer besonderen Form kodiert. Die Art der Kodierung wird URL-Kodierung genannt. Dabei werden Leerzeichen durch ein „+“ ersetzt. Buchstaben außerhalb der 7-Bit-ASCII Werte werden mit einem „%“-Zeichen und dem zweistelligen Zahlenwert des Buchstabens in hexadezimaler Form dargestellt.

Der so kodierte Text wird zusammen mit einem „?“-Zeichen an die Adresse des aufgerufenen Programms angehängt. Der kodierte Text wird dann an den Webserver zurückgesendet, der das „Anhängsel“

abtrennt und dem Programm in der Form eines Aufrufparameters weitergibt.

Das funktioniert genauso, als würde jemand am Webserver sitzen und das angesprochene Programm von der Kommandozeile aus mit dem kodierten Text als Parameter aufrufen.

Diese Methode wird übrigens nur bei der `<isindex>`-Anweisung verwendet, da sie nicht allgemein gültig ist.

G-EXP10: Muster für generiertes ISINDEX-Script

Schreiben Sie eine HTML-Datei, die den `<isindex>`-Aufruf enthält. Experimentieren Sie mit dem *prompt*-Parameter. Die Datei wird vom Browser genauso angezeigt, als wäre der Inhalt von einem Programm auf dem Server gesendet worden.

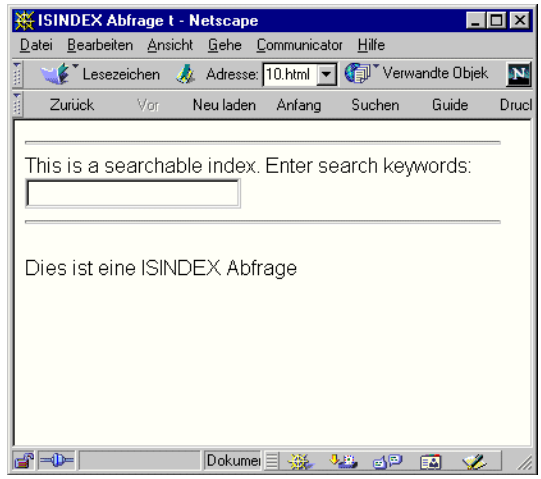

Abbildung 17.9: Beispiel für ISINDEX

Im nächsten Kapitel

Wir haben mit den Eingabemöglichkeiten die Grundlage gesehen, wie man möglicherweise ganze Anwendungen mit Hilfe eines Browsers, eines Webservers und der geeigneten Anwendungsprogramme schreiben kann.

Machen wir einen kurzen Ausflug in die Welt der Systemarchitekturen, um uns einen Eindruck von den möglichen Auswirkungen der WWW-Welt zu verschaffen.

17.11 Lösungen

G-EXP01: Formular mit einer 1 aus n-Auswahl

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.73 [de]C-CCK-MCD DT  (WinNT; U)
   [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp01</title>
</head>
<body>
<h1>
Allgemeines Formular</h1>
Formulare sind Sammlungen von Benutzereingaben, die einem Programm zugesen-
det
werden
<p>Ein Formular wird mit der &lt;form>-Anweisung gebildet. Die Parameter
geben die &Uuml;bertragungsmethode ("method") und das Zielprogramm
("action")
an.
<p>Es gibt zwei Methoden: "get" und "post". "get" wird benutzt, um kleinere
Datenmengen an Programme weiterzugeben. Diese Daten landen dann in einer
Umgebungsvariablen. Sie hei&szlig;t: "QUERY_STRING".
<p>"post" eignet sich f&uuml;r gro&szlig;e Datenmengen. Die Daten werden
vom Web-Server gepuffert und an den Standard-Eingang des aufgerufenen Pro-
gramms
weitergegeben.
<p>Das Zielprogramm wird mit dem Parameter "action" angegeben.
<br> 
<h2>
1 aus n - Auswahl (radio buttons)</h2>
<form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Braten</td>
<td><input type="radio" name="eingabe1" value="Braten" ></td>
</tr>
<tr>
<td>Nudeln </td>
<td><input type="radio" name="eingabe1" value="Nudeln" ></td>
</tr>
<tr>
<td>Huhn</td>
<td><input type="radio" name="eingabe1" value="Huhn" ></td>
```

```
</tr>
<tr>
<td>Eier</td>
<td><input type="radio" name="eingabe1" value="Eier" ></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP02: Einzelauswahl (n aus m)

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.73 [de]C-CCK-MCD DT  (WinNT; U)
   [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp02</title>
</head>
<body>
<h1>
Allgemeines Formular</h1>
Formulare sind Sammlungen von Benutzereingaben, die einem Programm zugesen-
det
werden
<p>Ein Formular wird mit der &lt;form>-Anweisung gebildet. Die Parameter
geben die &Uuml;bertragungsmethode ("method") und das Zielprogramm
("action")
an.
<p>Die Daten landen bei einer &Uuml;bertragung mit "get" in einer Umge-
bungsvariablen.
Sie hei&szlig;t: "QUERY_STRING".
<p>"post" eignet sich f&uuml;r gro&szlig;e Datenmengen. Die Daten werden
vom Web-Server gepuffert und an den Standard-Eingang des aufgerufenen Pro-
gramms
weitergegeben.
```

```
<p>Mit Einzel-Auswahlboxen kann man beliebig viele M&ouml;glichkeiten
gleichzeitig
ausw&auml;hlen. In unserem Besoiel w&auml;re Wien und Wasser m&ouml;glich.
<p>Eine Eingabe sieht wie folgt aus:
<br>&lt;input type="checkbox" name="wasser" value="ja" >
<br>Die Wertvorgabe wird bei der &Uuml;bertragung eines ausge&auml;hlten
Elementes benutzt. Bei Sekt fehlt die Vorgabe zum Experimentieren
<h2>
n aus m - Auswahl (check boxes)</h2>
<form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Wasser</td>
<td><input type="checkbox" name="eingabe0" value="ja" ></td>
</tr>
<tr>
<td>Limo</td>
<td><input type="checkbox" name="eingabe1" value="ja" ></td>
</tr>
<tr>
<td>Wein</td>
<td><input type="checkbox" name="eingabe2" value="ja" ></td>
</tr>
<tr>
<td>Bier</td>
<td><input type="checkbox" name="eingabe3" value="ja" ></td>
</tr>
<tr>
<td>Sekt</td>
<td><input type="checkbox" name="eingabe4" ></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP03: Eingabe mit einzeiligen Textfeldern

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp03</title>
</head>
<body>
<h1>
Allgemeines Formular</h1>
Texteingaben gibt es in drei verschiedenen Formen: als einelne Textzeile
als normaler Text, als einzelnen Textzeile als Passworteingabe und als
mehrzeiliges Eingabefeld.
<p>F&uuml;r die Gr&ouml;&szlig;e des Einagbefeldes benutzt der Browser
Standardvorgaben. Sie k&ouml;nnen die Gr&ouml;&szlig;e des Eingabefeldes
mit "size" steuern. Mit "maxlength" k&ouml;nnen Sie maximale Zahl eingege-
bener
Zeichen festlegen.
<p>Beispiel:
<br>&lt;input type="text" name="eingabe1" value="Vorschlag">
<br>Die Wertvorgabe wird im Eingabefensterangezeigt und benutzt, falls
sie vom Benutzer nicht gel&ouml;scht wird.
<h2>
Texteingaben</h2>
<form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Nachname</td>
<td><input type="text" name="eingabe1" ></td>
</tr>
<tr>
<td>Vorname</td>
<td><input type="text" name="eingabe2" ></td>
</tr>
<tr>
<td>Login</td>
<td><input type="text" name="eingabe3" ></td>
</tr>
<tr>
<td>Passwort</td>
<td><input type="password" name="eingabe4" ></td>
</tr>
</table>
```

```
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP04: Formular mit mehrzeiligem Eingabefeld

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.73 [de]C-CCK-MCD DT  (WinNT; U)
   [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp04</title>
</head>
<body>
<h1>
Allgemeines Formular</h1>
Texteingaben mit mehrzeiligem Eingabefeld
<p>Das mehrzeilige Eingabefeld verh&auml;lt sich etwas anders wie das ein-
zeilige.
Hier kann man die gew&uuml;nschten zeilen und Spalten am Bildschirm ange-
ben.
<p>Zeilenschaltungen werden hier als Text aufgefa&szlig;t und in den Text
&uuml;bernomen. Im Gegensatz dazu schlo&szlig; eine Zeilenschaltung die
Eingabe bei einzeiligen Texteingaben ab.
<p>Beispiel:
<br>&lt;textarea name="eingabe1" cols="70" rows="10" >Bitte geben Sie hier
Ihre Kommentare ein!&lt;/textarea>
<br>Die Wertvorgabe wird im Eingabefensterangezeigt und benutzt, falls
sie vom Benutzer nicht gel&ouml;scht wird.
<p>Noch eine Randbemerkung zum Echo-Programm: wenn Sie das Echo-Programm
der Buch-Web-Seite benutzen, werden in der Anzeige Zeilenschaltungen ent-
fernt.
Sie gelangen zum Server, werden aber in diesem Echo-Programm nicht ange-
zeigt.Au&szlig;er
Sie werfen einen Blick auf die URL-kodierte Information im Pfad.
<h2>
```

```
Texteingaben (mehrzeilig)</h2>
<form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Bemerkungen</td>
<td><textarea name="eingabe1" cols="70" rows="10" >Geben Sie bitte hier
Ihre Kommentare ein!</textarea></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP05: Mehrfach-Auswahlbox

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp05</title>
</head>
<body>
<h1>
Formulare</h1>
Auswahllisten gibt es in zwei Formen: mehrzeilige und einzeilige.
<p>Beispiel:
<br>&lt;select name="eingabe4" size="4">
<br>&lt;option>Shanghai&lt;/option>
<br>&lt;option>Wuxi&lt;/option>
<br>....
<br>&lt;/select>
<br> 
<h2>
Auswahllisten in HTML (mehrzeilig)</h2>
Die mehrzeiligen Auswahllisten erlauben auch die Selektion mehrerer
Eintr&auml;ge.
```

```
Die Tasten- und Mausklick-Kombination h&auml;ngt vom verwendeten Betriebs-
system
ab. Dazu wird bei der &lt;select>-Anweisung der zus&auml;tzliche Parameter
"multiple" angegeben. Vorsicht: das Echo-Programm zeigt nur einen Wert
an.
<br> 
<p><form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Reiseziele</td>
<td><select name="eingabe8" size="4" multiple><option>Shanghai</
option><option>Wuxi</option><option>Peking</option><option>Prag</
option><option>M&uuml;nchen</option><option>Havanna</option><option>Pil-
sen</option></select></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP06: Einfach-Auswahlbox

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp06</title>
</head>
<body>
<h1>
Formulare</h1>
Nebern der mehrzeiligen Auswahlbox gibt es ncoh die einzeilige. Man
mu&szlig;
nur in der &lt;select>-Anweisung den Gr&ouml;&szlig;enwertt auf "1" setzen.
<p>Beispiel:
<p><tt>&lt;select name="eingabe1" size="1"></tt>
```

```
<br><tt>&lt;option>Shanghei&lt;/option</tt>
<br><tt>&lt;option>Wuxi&lt;/option></tt>
<br><tt>...</tt>
<br><tt>&lt;/select></tt>
<br><tt></tt> 
<h2>
Auswahllisten in HTML (einzeilig)</h2>
Auswahllisten werden &auml;hnlich den Mitteln einer GUI-graphischen Benut-
zer
Oberfl&auml;che gestaltet. Hier
<br>haben wir eine einzeilige Auswahlliste geschrieben. Einzeilige Auswahl-
listen
geben immer nur einen Wert zur&uuml;ck.
<br> 
<br> 
<p><form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Reiseziele</td>
<td><select name="eingabe8" size="1"> <option>Shanghai</
option><option>Wuxi</option><option>Peking</option><option>Prag</
option><option>M&uuml;nchen</option><option>Havanna</option><option>Pil-
sen</option></select></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP07: Formular mit einzelner Texteingabe

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp07</title>
```

```
</head>
<body>
<h1>
Formulare</h1>
Ein Formular, dass nur eine einzelne, einzeilige Texteingabe besitzt, kann
nach der Eingabe des Textes auch durch die Enter/Return-Taste abgesendet
werden.
<p>Beispiel:
<p><tt>&lt;form action="xx" method="post"></tt>
<br><tt>&lt;input type = "text" name="eingabe2"></tt>
<br><tt>&lt;input type="submit" value="Senden"></tt>
<br><tt>&lt;/form></tt>
<br> 
<h2>
Formulare mit einer einzelnen Text-Eingabe</h2>
<form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Eine Eingabe</td>
<td><input type="text" name="eingabe1"></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP08: Graphischer Sendeknopf

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>g-exp08</title>
</head>
<body>
<h1>
```

```
Formulare</h1>
Einf&uuml;gen eines graphischen Sendeknopfes
<p>Beispiel:
<br>&lt;form action ="xxx" methode = "post">
<br>...
<br>&lt;input type="image" name="sendegraphik" src="../farbsucher.jpg">
<br>&lt;/form>
<h2>
Graphischer Sendeknopf</h2>
Der Sendekopf kann auch graphisch gestaltet werden. In diesem Fall werden
neben den Formularelementen auch zwei zus&auml;tzliche Koordinaten
&uuml;bertragen.
<br>An den Namen des Sendefeldes werden einmal ein ".x" und einmal ein
".y" angeh&auml;ngt, als Namen &uuml;bertragen und als Wert jeweils durch
die x- oder y-Koordinate erg&auml;nzt.
<p>Das Severprogramm kann also exakt feststellen, auf welchen Punkt der
Benutzer geklickt hat.
<br> 
<p><form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Reiseziele</td>
<td><select name="eingabe8" size="1" ><option>Shanghai</
option><option>Wuxi</option><option>Peking</option><option>Prag</
option><option>M&uuml;nchen</option><option>Havanna</option><option>Pil-
sen</option></select></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="image" name="mein_sendeknopf" src="../exp-bilder/
hintergrund1.gif"></td>
</tr>
</table>
</form>
</body>
</html>
```

G-EXP09: Datei-Upload Steuerung

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
```

```
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="Author" content="Walter Herglotz">
   <meta name="GENERATOR" content="Mozilla/4.73 [de]C-CCK-MCD DT  (WinNT; U)
   [Netscape]">
   <title>Datei Upload</title>
</head>
<body>
<h1>
Datei Upload zum Server</h1>
<p><br><form method="post" action="http://www.walter-herglotz.de/pro-
gramme/echo_var.php3" enctype="multipart/form-data">
<br>Geben Sie einen lokalen Dateinamen ein:<br> <input type="file"
name="eingabe1">
<br><input type="submit" >
<br></form>
<p>
<hr WIDTH="100%">
<br> 
</body>
</html>
```

G-EXP10: Muster für generiertes ISINDEX-Script

```
<html>
<head>
<title>
ISINDEX Abfrage
t</title>
<isindex>
</head>
<body>
<p>Dies ist eine ISINDEX Abfrage
</p>
<body>
</html>
```

Pearson Education

ADDISON-WESLEY

18 IT-Architektekturen mit Browsern und Servern

Vielleicht waren Sie erstaunt, welche Vielfalt HTML für die Interaktion mit dem Benutzer in Formularen und Graphiken zur Verfügung stellt. Letztlich beinhaltet HTML im Zusammenwirken mit Programmen auf dem Server alle grundlegenden Möglichkeiten der Kommunikation mit dem Benutzer, die auch in den üblichen Benutzeroberflächen enthalten sind.

Das hat weitreichende Auswirkungen. Man kann völlig neue IT-Architekturen aufbauen.

PC-Benutzer arbeiten typischerweise mit den großen und flexiblen Möglichkeiten der heutigen Standardprogramme. Aber dies geht nur so lange gut, wie man nicht in den größeren Rahmen einer Firma eingebunden ist. Immer dann, wenn viele Personen auf die gleichen Daten zugreifen müssen, kommen IT-Architekturen ins Spiel. Meist stehen heute alle wichtigen Unternehmensdaten in zentralen Datenbanken. Betrachtet man ganze Firmen, dann wird man sich Gedanken machen müssen, wie die Daten gehalten und wie die Aufgaben und Programme verteilt werden.

18.1 Client - Server - Architektur

Eine Idee war, die Daten zentral auf Datenbankservern zu halten, einige Verarbeitungsschritte dort auch zu programmieren und über das Netz die Client-Rechner anzuschließen, die dann spezialisierte Anwendungen zum Zugriff auf die zentralen Datenbanken ausführten.

Diese Client-Server-Anwendungen litten (und leiden) unter zwei Hauptproblemen: der oft zu geringen Bandbreite des Netzwerks zwischen Client und Server und der unklaren Aufgabenverteilung zwischen den beiden. Oft hat man versucht, möglichst viel Verarbeitungsleistung auf den Client zu legen. Der Preis war dabei die hohe Transportrate zwischen Server und Client.

18.2 Mehrschichtarchitektur

Mit Browsern und Webservern, die heute in jeder größeren Firma als Intranet zur Verfügung stehen, lassen sich nun andere Konzepte verfolgen.

Der Browser realisiert eine schnelle Benutzerschnittstelle auf dem Rechner des Benutzers. Diese Funktion als Benutzerschnittstelle kann man in ihrer Funktion klar abgrenzen. Man nennt dies auch eine Softwareebene oder Schicht.

Die Benutzerschnittstelle oder Benutzerschicht kommuniziert nun mit dem üblichen Protokoll mit einem Webserver, der die Kommunikationssteuerung zwischen der Benutzerebene und den HTML-Seiten und Anwendungsprogrammen auf dem Server übernimmt. Wieder haben wir eine klar abgegrenzte Aufgabe beschrieben.

Als dritte Schicht in der gerade beschriebenen IT-Architektur betrachtet man die hinter dem Webserver liegenden Anwendungsprogramme, die die eigentliche Verarbeitung durchführen können. Man sagt auch, dass diese Programme die Geschäftslogik enthalten.

Und nun bleibt nur noch die vierte Schicht zu erwähnen, die die Datenbanken beinhaltet. In den Datenbanken einer größeren Firma liegen heute alle Geschäftsdaten. Das ist die Voraussetzung, dass mehrere Mitarbeiter mit den gleichen Daten arbeiten können und trotzdem immer ein einheitlicher Datenbestand gewährleistet ist.

Dieses Modell ist ein 4-Schichten (engl. tier) Modell, das eine saubere Trennung der Funktionen vorsieht und auch von mehreren verteilten Rechnern abgearbeitet werden kann. Damit erhalten wir auch eine Lastverteilung.

Wenn ich heute eine Systemarchitektur für eine neue Applikation in Firmen festzulegen hätte, würde ich sicherlich einen Versuch mit mehreren Schichten machen.

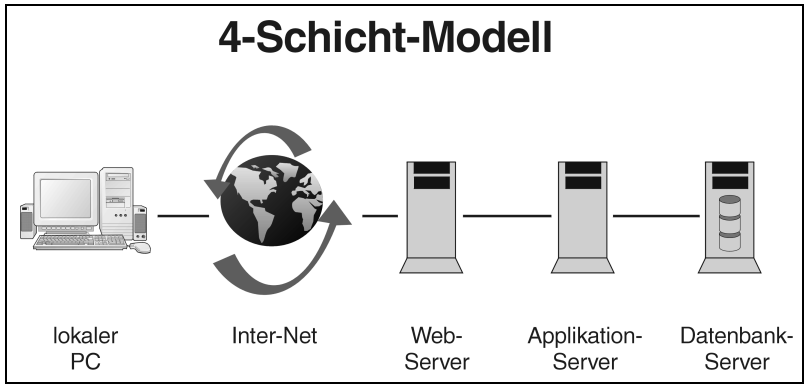

Abbildung 18.1: Das 4-Schicht-Modell

Es gibt dabei noch eine wichtige Randbedingung. In vielen Firmen gibt es ja neben den verwaltenden Tätigkeiten am Schreibtisch noch entwickelnde oder produzierende Abteilungen, die oft die eigentliche Wertschöpfung einer Firma erbringen. Dort werden aber oft spezialisierte Rechner eingesetzt. Für eine CAD-Entwicklung sind dies vielleicht SGI-Rechner, die Schaltungsentwickler verwenden oft Sun-Workstations.

Mit einer browsergestützten Mehrschichtarchitektur kommen auch die produktiven Teile einer Firma in den Genuss des Zugangs zu firmenweiten Verfahren. Systembrüche werden vermieden.

18.3 ASP – eine mögliche Anwendung der Mehrschichtarchitektur

In der Literatur stolpert man immer wieder über die Abkürzung *ASP* (application service provider / Anwendungsprogramm Anbieter). Die Idee dahinter ist einfach.

Ein Anbieter nutzt ein Rechenzentrum, um Dienstleistungsangebote einem breiten Anwenderkreis zur Verfügung zu stellen. Der Nutzer kann sich mit Hilfe seines Browsers beim *ASP* anmelden und den Dienst nutzen, ohne jemals für diesen Dienst eigene Software kaufen zu müssen und, was meist noch teurer ist, sie zu betreuen und zu installieren.

Wenn eine Firma hier die richtigen Dienste findet, kann sie diese Dienste sehr schnell weltweit anbieten und vielleicht genauso schnell wachsen.

Im nächsten Kapitel

Das vorangegangene Kapitel über die Interaktionsmöglichkeiten mit HTML hat die Basis geboten für die Diskussion der Architektur. In den folgenden Kapiteln werden wir weitere Bausteine einer solchen Architektur bereitstellen. Wir benötigen noch Verfahren, unserem Webauftritt ein einheitliches Aussehen zu geben. Und natürlich werden wir die statischen Webseiten um programmierte und damit dynamische Elemente erweitern.

Auf dem beschriebenen Weg betrachten wir nun die Anweisungen, die im Kopf der HTML-Seite stehen können. Einige betreffen die einzelne Seite, andere Gruppen von Seiten und schließlich werden wir auch Angaben zu Vorgaben machen können, die den Stil aller Webseiten eines Webauftrittes beeinflussen.

Wenden wir uns daher nun den Angaben im Kopf einer Seite detailliert zu.

19 Meta-Anweisungen

Im Kopf einer HTML-Datei finden Sie oft weitere Angaben über die einfache Titelangabe hinaus. Wenn Sie einmal im Internet surfen und sich eine beliebige Datei mit Hilfe der Browserfunktion ANSICHT/QUELLTEXT betrachten, dann können Sie oft eine ganze Reihe dieser Meta-Anweisungen sehen.

Meta bedeutet *übergeordnet*. Im Kopf stehen also Informationen über den eigentlichen Inhalt, den wir in der `<body>`-Anweisung zur Verfügung stellen. Einen Eintrag haben wir immer geschrieben, denn er ist Pflicht: den Titel.

Dieser Titel gehört nicht zur eigentlichen Seite und wird deshalb auch außerhalb des Anzeigefensters im Rand angezeigt.

Neben der speziellen Titelangabe gibt es Reihe von weiteren Angaben, die man mit Hilfe der `<meta>`-Anweisung in den Kopfbereichen einfügt. Meta-Informationen fallen in zwei Gruppen: die eine gibt Informationen, die sonst auch über das Verbindungsprotokoll HTTPD zwischen dem Browser und dem Server laufen könnten, die andere beschreibt den Inhalt näher. Die erste Gruppe nutzt den Parameter *http-equiv*, was soviel bedeutet wie *http-equivalent / einem HTTP-Befehl entsprechend*. Die zweite Gruppe verwendet schlicht den Parameter *name*.

Die Informationen stehen den beiden Beteiligten der Kommunikationsverbindung zur Verfügung. Sowohl der Server als auch der Browser kann die Informationen im Kopf lesen und für ihre eigene Steuerung benutzen.

19.1 HTTP-EQUIV-Werte

Ein Meta-Element mit dem Parameter *http-equiv* ergänzt die Informationen, die zwischen dem Server und dem Browser ausgetauscht werden. Wichtige Anwendungsbereiche sind die Möglichkeit, nähere Angaben über den Inhalt und den verwendeten Zeichensatz zu machen, Informationen für Zwischenspeicher (engl. caches) oder Weiterleitungen zu anderen Dokumenten einzurichten.

Werfen wir noch einmal einen Blick auf die schon bekannte Angabe des Zeichensatzes.

Beispiel:

```
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
```

Diese Angabe ist immer zu empfehlen. Vielleicht verhindert sie die unnötige Ersetzung der Umlaute durch Sonderzeichen. Sicher ist die Angabe des Zeichensatzes im internationalen Rahmen sinnvoll.

M-EXP01: Angabe des Dokumententyps und des verwendeten Zeichensatzes

Geben Sie im Kopf einer HTML-Datei den Dateityp und den verwendeten Zeichensatz an. Bereits in den ersten Kapiteln haben wir die Notwendigkeit diskutiert, eine bestimmte Datei auf allen Rechnern gleich darzustellen. Dazu nutzen wir die Möglichkeit, den verwendeten Zeichensatz in der Datei anzugeben.

Die zweite Variante, die wir in der Einleitung besprochen haben, steuert die Zwischenpuffer in der Übertragungsstrecke zwischen dem Server und dem Browserfenster.

Zwischenpuffer werden an vielen Stellen eingesetzt, um die Übertragungskapazität des Internets besser zu nutzen. Der Browser unterhält meist bereits zwei Zwischenpuffer: einen im Hauptspeicher und einen auf der Festplatte. Wenn in einer Seite ein Logo (z.B. ein Firmenzeichen) vorkommt und wir auf die nächste Seite der gleichen Firma wechseln, die wiederum das Logo benutzt, dann holt der Browser das Bild sehr wahrscheinlich aus dem internen Puffer und überträgt es nicht erneut über das Netz.

Ein Zwischenspeicher sitzt vermutlich im internen Netz, wenn Sie in einer etwas größeren Firma arbeiten. Bekannt ist hier als Programm *squid*. Aber auch auf der Seite des Providers sitzen Zwischenspeicher. Wenn Sie sich vorstellen, dass bestimmte Webseiten sehr oft aufgerufen werden (z.B. die großen Magazine oder der Wetterbericht), dann

kann eine Zwischenspeicherung von ein paar Gigabytes beim Provider den gesamten Verkehr deutlich verringern.

Allerdings führt dies dazu, dass die Webseite, die Sie gerade betrachten, gar nicht aktuell sein muss. Stellen Sie sich vor, Sie betrachten eine Seite mit den Lottozahlen, die einen Monat schon im Zwischenpuffer liegt. Vermutlich wären Sie nicht begeistert.

Um solche Pannen zu vermeiden, können wir die Verfallsdauer einer Seite angeben. Die Programme, die die Zwischenspeicherung steuern, werden dann hoffentlich die veralteten Seiten aus ihrem Puffer werfen.

Ein Bespiel:

```
<meta http-equiv="expires" content= "Mon, 16 Apr 2001 19:56:43 GMT">
```

Das Verfallsdatum wird mit Datum und Uhrzeit in einem Standardformat angegeben. Für die Lottozahlen sollte man am höchsten eine Woche lang die Erlaubnis zum Puffern erteilen. Wollen Sie die Zwischenspeicherung verbieten, geben Sie statt des Datums eine *0* an.

Die dritte Variante kennen Sie wahrscheinlich von Weiterleitungen. Klickt man auf eine Seite, deren Adresse nicht mehr aktuell ist, weil sich der Aufbau und der Inhalt von Webauftritten geändert hat, dann blenden moderne Verwaltungsprogramm meist eine Weiterleitung ein. Der meist gleiche Text am Bildschirm heißt dann: *Klicken Sie hier, wenn Ihr Browser keine Weiterleitung unterstützt*.

Im Kopf dieser Hinweisseite steht dann eine Meta-Anweisung, die uns zu einer Zielseite verbindet.

Beispiel:

```
<meta http-equiv="refresh" content="10;url="seite10.html">
```

Nach zehn Sekunden schaltet der Browser zur angegebenen Seite weiter. Etwas ungewöhnlich ist dabei, dass hier nicht *href* sondern *url* verwendet wird.

M-EXP02: Dia-Show

Erstellen Sie mehrere Webseiten mit Bildern. Es könnte eine Präsentation der Produkte Ihrer Firma für eine Messe sein. Diese Webseiten sollen nun in Form einer Dia-Show kontinuierlich angezeigt werden. Die Produkte sollen dabei 20 Sekunden angezeigt werden und das Logo am Anfang 40 Sekunden.

Abbildung 19.1: Startseite der Diashow

Im Bildbeispiel wurden Bildschirmaufnahmen der Experimente in der Dia-Show verwendet. Beachten Sie, dass die Dia-Show eine definierte Startseite besitzt, die selbst nicht Teil der Dia-Show ist. Erst durch einen Klick startet die Show. Das Ende sollte ebenso bedacht werden. Der Benutzer benötigt eine Abbruchmöglichkeit. Dazu wurde in der Dia-Show-Seite ein Verweis auf die nicht selbst aktive Startseite eingebaut. Wenn der Benutzer die Show abbrechen will, findet er die Möglichkeit dazu durch einen Verweis.

Abbildung 19.2: Abbruchmöglichkeit

19.2 Standardisierungsprobleme

Allgemein lässt sich sagen, dass META Einträge nicht standardisiert sind. Häufig werden die folgenden Einträge interpretiert.

19.2.1 Angaben zu Autor und Editierprogramm (meist automatisch eingefügt)

```
<meta name="Author" content="Walter Herglotz">
<meta name="GENERATOR" content="Mozilla/4.7 [en] (Win95; I) [Netscape]">
```

19.2.2 Informative META-Angaben

Beschreibung und Schlüsselworte

```
<META NAME="description" CONTENT="Eine Beschreibung der HTML 4.0 META Elemente.">
<META NAME="keywords" CONTENT="META, meta element, metadata, metainformation, meta data, meta information, keywords, description, refresh, HyperText Markup Language, HTML, HTML4, HTML 4.0, Beschreibung, Schlüsselworte">
```

Sind diese Einträge vorhanden, kann eine Suchmaschine darauf zurückgreifen. Ansonsten bleibt einer Suchmaschine nur übrig, die ersten paar hundert Byte eines Dokuments zu lesen und auf Schlüsselwörter zu untersuchen. Wenn eines der Schlüsselwörter gefunden wird, kann die Suchmaschine statt der ersten Zeilen des Dokumentes die angegebene Beschreibung anzeigen. Das kann sowohl für die Suchmaschine als auch für den Autor sinnvoll sein.

19.2.3 Steuerung der Suchmaschinen (der Roboter)

Die Suchmaschinen des WWW durchkämmen das Netz, folgen von einem Verweis zu der Ziel-Webseite und von dort immer weiter. Die Hoffnung ist dabei, dass auf diese Art das ganze WWW erfasst und einigermaßen katalogisiert werden kann.

Das ganze Netz zu erfassen, ist längst zur Illusion geworden. Aber die Suchmaschinen haben sich inzwischen auch weiterentwickelt. Zusätzlich zu der automatischen Suchfunktion wurden Redaktionen aufgebaut, die gezielt suchen und Inhalte katalogisieren.

Und außerdem hat fast jede Suchmaschine inzwischen für Anbieter von Webseiten die Möglichkeit geschaffen, sich selbst anzumelden.

Sollten Sie Webseiten auf einem Webserver zur Verfügung stellen, können Sie durch einen Eintrag im `<meta>`-Abschnitt Suchmaschinen, die auch Roboter oder *crawler* (to crawl: robben) genannt werden, steuern.

Beispiel:

```
<meta name="robots" content="index, nofollow">
```

Der Wert *index* steuert dabei die Stichwortsuche in einem Dokument. Man kann die Stichwortsuche abschalten, wenn man als Wert *noindex* verwendet. Der zweite Wert steuert die Verwendung der Verweise in der Seite. Mit dem Wert *nofollow* verbietet der Autor der Suchmaschine, den Verweisen zu folgen und weitere Seiten zu untersuchen. Umgekehrt genehmigt der Wert *follow* der Suchmaschine, dem weltweiten Netz weiter zu folgen.

Mit dem Wert *all* kann man beides erlauben. Damit kann die Suchmaschine das Dokument durchsuchen und den Verweisen folgen. Dies ist auch die Voreinstellung. Umgekehrt verbietet *none* beides.

Im nächsten Kapitel

Die Entwicklung der HTML-Versionen haben nach und nach immer mehr Formatierungsmöglichen geboten.

Und bei den `<meta>`-Steuerungen wurden verschiedene Vorschläge erarbeitet, um Webseiten zu klassifizieren. Einer der Vorschläge ist unter dem Namen *DC-Dublin Core* bekannt geworden.

Der größte Einschnitt in der Entwicklung der Sprache HTML kam mit der Version 4.0. Endlich versuchte man, die Grundidee der Trennung von Inhalt und Formatierung wieder umzusetzen. Eine Vielzahl von direkten Formatierungen für Farben, Schriftarten oder Einzüge werden nun als *unerwünscht* (engl. deprecated) bezeichnet.

Als neues Mittel der Formatierung griff man zu einem parallel erarbeiteten Vorschlag, Formatelemente in einer separaten Datei zusammenzufassen und damit zentral die Formatierung zu steuern. Der Name des Vorschlags ist *CSS-Cascading Style Sheets* (Kaskadierte Stil-Vorlagen). Aber davon gleich mehr.

19.3 Lösungen

M-EXP01: Angabe des Dokumententyps und des verwendeten Zeichensatzes

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>M-EXP01: Zeichensatzangabe</title>
</head>
```

```
<body>
 
<h2>Dia Show </h2>
<br>Hier beginnen die Vorarbeiten zu einer Dia-Show.
<br> 
</body>
</html>
```

M-EXP02: Dia-Show/Start

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>M-EXP02: Dia-Show</title>
</head>
<body>
 
<br><a href="m-exp02-2.html">Dia Show starten</a>
<br> 
<br> 
<br> 
<br> 
</body>
</html>
```

M-EXP02-2: Dia-Show

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <meta http-equiv="refresh" content="20;url=m-exp02-3.html">
   <title>M-EXP02: Dia-Show</title>
</head>
<body>
 
<br>Dia Show
<p>Bild 2
<p><img SRC="../exp-bilder/b-exp01-1.gif" height=407 width=375>
</body>
</html>
```

M-EXP02-3: Dia-Show

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <meta http-equiv="refresh" content="20;url=m-exp02-2.html">
   <title>M-EXP02: Dia-Show</title>
</head>
<body>
 
<br>Dia Show
<p>Bild 3
<p><img SRC="../exp-bilder/b-exp01-1.gif" height=407 width=375>
<p><a href="m-exp02.html">Dia Show verlassen</a>
</body>
</html>
```

20 Stil-Vorlagen

Erinnern Sie sich noch an die Diskussionen am Anfang des Buches? Dort wurde die große Idee besprochen, dass der Inhalt einer HTML-Datei nur logische Auszeichnungen enthalten sollte. Die Darstellung und das Layout sollten außerhalb z.B. durch den Browser wahrgenommen werden.

Mit HTML 4.0 wurden verschiedene Standardisierungsprojekte zusammengefasst. Eines dieser Projekte befasst sich mit den Formatierungsbefehlen. Der Name des Projektes ist *CSS* (Cascading Style Sheets / kaskadierte Stil-Vorlagen).

Mit deren Hilfe kann man nun wieder „reines" HTML schreiben und die Formatierung zentral in einer einzigen Datei speichern. Stellen Sie sich eine große Firma vor, die sich bemüht, nach außen einen sauber gestalteten Auftritt für die Öffentlichkeit aufzubauen. Alle Spielregeln könnten in einer einzigen Datei gesammelt und auf alle Webseiten angewendet werden. Eine einzige Änderung in einer Datei würde automatisch alle Webseiten wie gewünscht verändern.

Eine schöne Vorstellung! Aber kein Traum. Denn genau das kann man mit Hilfe von so gennanten *Style-Sheets* (Stilvorlagen) erreichen.

Noch sind Stil-Vorlagen auf dem Weg der Standardisierung. Es gab zwei verschiedene Zielrichtungen. Die eine befasste sich vornehmlich mit den Attributen, die wir in diesem Kapitel zumindest in Teilen kennen lernen werden. Das Ergebnis ist heute als *CSS-1*-Standard veröffentlicht. Die andere kümmerte sich um die Positionierung von Elementen. Ihr Ergebnis kann man unter dem Namen *CSS-P* finden. Erst *CSS-2* wird beide Teile zusammenbringen.

20.1 Einführung in Stilvorlagen

Stilvorlagen werden sinnvollerweise in eine zentrale Datei geschrieben. Für den Moment würde aber diese Lösung doch einigen Aufwand bedeuten. Zum Lernen ist es meist einfacher, Stile und die restliche HTML-Datei zusammen zu bearbeiten. Beginnen wir mit dem einfachen Fall einer Stilvorlage innerhalb einer HTML-Datei. Der geeignete Platz ist innerhalb der `<head>`-Anweisung. Und natürlich fangen wir wieder mit den grundlegenden Befehlen an.

Ein Beispiel:

```
<style type="text/css">
h1 { color: rot }
em,s { color: green }
body {font-family:arial helvetica sans-serif;color: blue; }
</style>
```

Noch ein Wort zur Syntax der Anweisungen. Beachten Sie, dass der Doppelpunkt zwischen Anweisung und Werten benutzt wird. Dies steht im Gegensatz zu HTML, die ja ein Gleichheitszeichen verwendet. Werden zu einer Anweisung mehrere Werte angegeben, dann werden die einzelnen Werte durch Leerzeichen statt durch Kommas getrennt. Mehrere HTML-Anweisungen kann man in eine Liste vor eine einzelne Stil-Anweisung, die durch Kommas getrennt wird, schreiben. Und noch etwas ist anders: es gibt keine Anführungszeichen für Werte.

Die `<style>`-Anweisung umfasst die Stil-Anweisungen für die Datei. Im einfachsten Fall kann man für jede geeignete HTML-Anweisung geeignete Attribute festlegen. Nicht geeignet sind dabei Anweisungen wie `<hr>` oder `<br>`.

Im Beispiel werden alle `<h1>`-Überschriften rot, Text innerhalb einer `<em>`-Anweisung grün und Texte außerhalb der `<p>`-Anweisung blau. Die letzte Eigenschaft wird durch die Farbangabe für `<body>` erreicht. Da wir eine Liste aus HTML-Anweisungen benutzt haben, gilt die gleiche Einstellung für `<em>` und `<s>`. Die Texte innerhalb der `<body>`-Anweisung sollen mit der ersten gefundenen Schrift aus der angegebenen Schriftfamilie dargestellt werden.

20.2 Verschiedene Stil-Varianten

Im Beispiel haben wir durch die Typangabe (*style*) dem Browser mitgeteilt, dass es sich um einen Text handelt, der gemäß dem *CSS*-Standard aufgebaut ist. Diese Angabe ist Pflicht. Im Navigator von Net-

scape gibt es noch eine weitere Variante. Die Bezeichnung ist dafür: *text/javascript*. Netscape hat hier eine Syntax entwickelt, die sich an die Sprache *Javascript* anlehnt. Da diese Variante aber nicht allgemeingültig ist, wollen wir sie nicht näher betrachten.

20.3 Stil-Angaben und alte Browser

Browser überlesen unbekannte Anweisungen, die ja bekanntlich in spitzen Klammern stehen. Sollte einer der Leser Ihrer Seiten einen alten Browser verwenden, dann würde er durch `<style>` und auch `</style>` nicht gestört. Es bleibt nur das Problem, dass der Browser möglicherweise versucht, die einzelnen Stil-Angaben zu interpretieren. Und das ist nicht möglich, wenn er nicht auch *CSS* kennt.

Man löst das Problem recht elegant. Ein Browser, der *CSS* kennt, ignoriert die HTML-Kommentarzeichen innerhalb der `<style>`-Anweisung. Also kann man als erstes innerhalb der `<style>`-Anweisung einen HTML-Kommentar schreiben, der nur von alten Browsern ausgewertet wird, und den Kommentar unmittelbar vor der Ende-Marke wieder schließen.

Die beste Schreibweise ist also wie folgt:

```
<style type = "text/css">
<!--
h1 {color: red}
-->
</style>
```

Hier im Buch wollen wir die Kommentare weglassen. Die Beispiele lesen sich leichter, wenn nur die wesentlichen Elemente vorgestellt werden. Und leider hat ein Buch auch nur eine begrenzte Zahl von Seiten, die wir besser für weitere Experimente nutzen wollen.

Übersicht über wesentliche Stil-Anweisungen		
background	Hintergrund	Farbname
background-image	Hintergrund-Bild	Oder rgb(r,g,b)
border-color	Rahmen-Farbe	Oder rgb(r%,g%,b%)
border-width	Rahmen-Breite	thin, medium, thick, XX
color	Farbe	Wie oben
font-family	Zeichensätze	Arial, serif, monospace ...
font-size	Größe der Zeichen	large, small.. 14pt
font-style	Attribute	Normal, italic

Übersicht über wesentliche Stil-Anweisungen		
font-weight	Schriftdicke	Bold, 100...900
margin	Abstand	XX
margin-left	Linker Abstand	Längenangabe
margin-right	Rechter Abstand	Längenangabe
margin-top	Kopf Abstand	Längenangabe
margin-bottom	Fuß Abstand	Längenangabe
text-align	Ausrichtung	left | center | right
text-decoration	Verschiedene Textmarkierungen	None | line.|.through | overline | underline
text-indent	Einzug erste Zeile	Längenangabe
text-transform	Automatisches Umsetzen von Zeichen	capitalize | lowercase | uppercase

Tabelle 20.1: Grundlegende Attribute

Farben können wie schon gewohnt, mit einem Namen oder einem RGB-Hex-Wert angegeben werden. Zusätzlich gibt es eine rgb()-Funktion, die drei dezimale oder prozentuale Parameter akzeptiert.

Längenangaben können in verschiedenen Maßeinheiten angegeben werden. Relative Längen werden durch *px* (Pixel), *em* (em) und *ex* (x) angegeben. Absolute Werte werden mit *in* (inches), *cm* (centimeters), *mm* (millimeters), *pi* (picas) und *pt* (points).

20.4 Vorsicht bei einfachen Editoren

Manche Editoren (z.B. der Composer von Netscape) entwickeln ein Eigenleben. Manche Auszeichnungen werden durch andere ersetzt. So entfernt mein Composer `<em>`-Anweisungen und ersetzt sie durch `<i>`-Anweisungen.

Aus diesem Grund sind die Beispiele in diesem Kapitel alle mit einem einfachen Texteditor geschrieben, um die erwähnten Überraschungen zu vermeiden.

S-EXP01: Stil-Vorgaben für eine HTML-Datei

Erstellen oder ändern Sie eine HTML-Datei ab und führen Sie den Stil ein, dass alle vorformatierten `<pre>`-Abschnitte in reiner blauer Farbe dargestellt werden. Nutzen Sie die rgb()-Funktion, um die Farbe einzustellen.

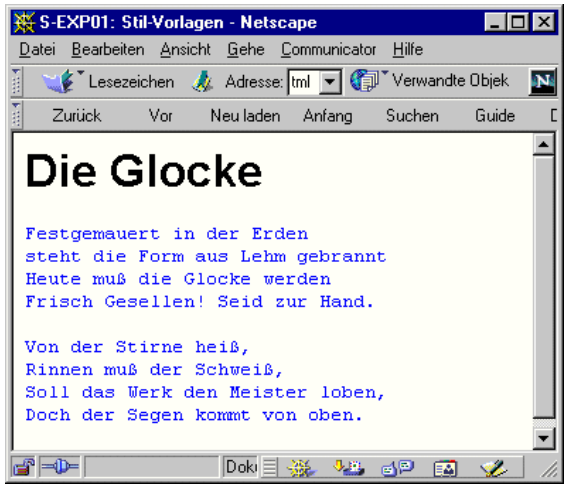

Abbildung 20.1: Datei mit Stilvorgabe

20.5 Wertegruppen für Blöcke

Für Blöcke kann man verschiedene Angaben machen, die die gesamte Umrandung betreffen. Das kann nun ein Rahmen (*border*) oder auch eine Abstandsangabe (*margin*) sein. Einen Rahmen kann man je nach Anzahl der verwendeten Werte mit einem gemeinsamen Wert für alle vier Seiten (oben, unten, rechts und links) oder Kombinationen daraus oder letztlich getrennt für jeden einzelnen der vier Teile bestimmen.

In der Tabelle sind solche Angaben mit „XX“ angezeigt. Ein Wert gilt dann für alle vier Seiten, zwei Werte stehen für oben und unten sowie links und rechts, bei drei Werten werden nacheinander oben, links gemeinsam mit rechts sowie unten angegeben und bei vier Werten beginnen wir oben und geben die Werte im Uhrzeigersinn an.

Machen wir noch weitere Übungen. Als nächstes wollen wir einmal einen breiten Rand auf der linken Seite einrichten. Breite Ränder können wir z.B. im Zusammenspiel mit Hintergrundbildern nutzen. Ein immer wieder verwendetes Bildmotiv ist ein Spiralblock. Hinterlegt man eine Seite mit einem solchen Motiv, dann soll ja kein Text auf der ganz links gezeigten Spirale stehen. Eine Einrückung mit einem Rand würde diese Aufgabe erfüllen.

S-EXP02: Randeinstellungen

Schreiben oder ändern Sie eine HTML-Datei mit enthaltenen *CSS*-Elementen.

Legen Sie mit einem *CSS*-Element fest, dass für den gesamten `<body>`-Bereich ein 3 cm breiter Einzug auf der linken Seite eingerichtet wird.

Sie können zwischen der *margin* oder *margin-left*-Anweisung wählen. Aber Vorsicht: die *margin*-Anweisung benötigt vier Werte.

Abbildung 20.2: Einzug mit StilAnweisungen

Die meisten Stil-Angaben kennen wir schon von HTML-Anweisungen. Auch dort ließ sich die Farbe eines Textes ja durchaus einstellen. In einigen Fällen aber kann man mit *CSS*-Anweisungen deutlich mehr beschreiben. Probieren wir eine Umrandung.

S-EXP03: Umrandung von Texten

Schreiben Sie eine HTML-Datei oder ändern eine vorhandene ab. Eine `<h1>`-Überschrift soll nun eine Umrandung erhalten. Oben und links soll ein dünner Strich angezeigt werden, unten und rechts ein dicker.

Möglicherweise schaut das Ergebnis ein wenig einem 3D-Knopf der graphischen Oberfläche gleich. Die Reihenfolge der Werte beginnt wieder oben und folgt dem Uhrzeigersinn.

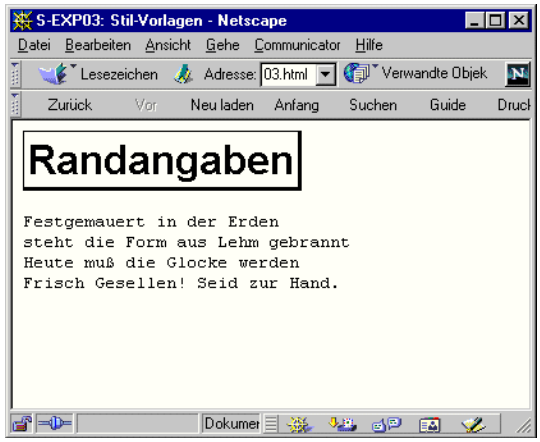

Abbildung 20.3: Blockränder

Kommen wir zu einem anderen Thema, der speziellen Bearbeitung von Textstellen. Der Browser kann einzelne Textstellen individuell bearbeiten. Er kann z.B. sicherstellen, dass alle Wörter einer Überschrift immer mit Großbuchstaben beginnen, oder dass die erste Zeile eines Absatzes eingerückt wird.

S-EXP04: Automatische Formatierungen

Schreiben oder verändern Sie eine HTML-Datei mit *CSS*-Elementen. Legen Sie den Einzug der ersten Zeile jedes Absatzes fest und bestimmen Sie, dass jeder Anfangsbuchstabe in einer `<h1>`-Überschrift automatisch groß geschrieben wird, auch wenn Sie im HTML-Text vorsätzlich kleine Buchstaben verwenden.

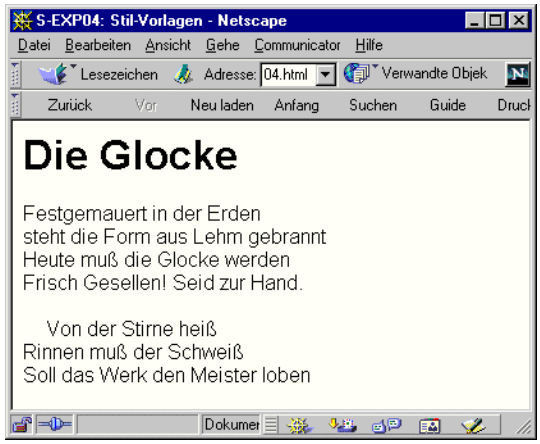

Abbildung 20.4: Automatische Formatierungen

Vielleicht probieren Sie die automatische Großschreibung der ersten Buchstaben auch mit Umlauten aus. Nicht alle Browser können korrekt mit so einfachen Operationen umgehen. Schade eigentlich.

20.6 Variationen für Anweisungen (Klassen)

Ein und dieselbe HTML-Anweisung kann in ganz verschiedenen logischen Situationen vorkommen. Ein Absatz kann in einer Einleitung, als Begleittext eines Bildes oder auch Teil eines Artikels vorkommen. Je nach Anwendungsfall sollte vielleicht ein Absatz ganz anders dargestellt werden oder Attribute erhalten.

Daher kann man in den Stil-Vorgaben sogenannte Klassen bilden. Eine Klasse wird durch einen lokalen Namen definiert, der dann hinter jede für die Klasse beschriebene Anweisung mit einem Punkt gesetzt wird. Wie alle lokalen Namen müssen wir dabei Groß- und Kleinschreibung beachten.

Damit kann eine Stil-Anweisung z.B. so aussehen:

```
p.einleitung {color: blue }
p.artikel {color: black }
h1.einleitung {color: green }
h1.artikel {color: red }
.artikel {font-size: 14pt }
```

Will man nun z.B. einer Überschrift mitgeben, dass sie zum Artikel gehört, fügt man in die `<h1>`-Anweisung den Parameter *class* ein, der als Wert den Namen der gewünschten Klasse erhält. Da es sich um einen HTML-Parameter handelt, benutzen wir hier wieder ein „=“-Zeichen.

Eine Besonderheit zeigt die letzte Zeile des Beispieles. Sie enthält keine HTML-Anweisung, sondern nur den Punkt gefolgt von einem Klassennamen. Damit lassen sich Einstellungen für alle HTML-Anweisungen der Klasse vorgeben. Man kann also, wie hier, allen Absätzen, Überschriften etc. der Klasse *artikel* einen gemeinsamen Stil vorgeben.

S-EXP05: Klassen für Stile

Schreiben oder ändern Sie eine HTML-Datei mit Stil-Elementen.

Es sollen drei Klassen eingerichtet werden. Eine Klasse für die Einleitung, den Artikel und das Literaturverzeichnis.

Ein Absatz in der Einleitung soll kursiv geschrieben werden, im Artikel sollen Artikel mit einer Schrift mit Serifen (*serif*) dargestellt werden und schließlich sollen Absätze im Literaturverzeichnis mit kursiv und mit 10 Punkten Schrift am Bildschirm erscheinen.

Geben Sie außerdem vor, dass Hervorhebungen (*em*) in der Einleitung blau und im Artikel rot gezeigt werden.

Schreiben Sie auch ein wenig Text, um die Fälle auszuprobieren.

Abbildung 20.5: Gruppierung von Stilelementen

20.7 Spezielle Auszeichnung eines bestimmtes Containers

Neben den Möglichkeiten der Angabe von Klassen, die für mehrere Container in einer Webseite gelten können, gibt es noch die Möglichkeit spezielle Auszeichnungen unter einem lokalen Namen festzulegen, der wie alle lokalen Namen, in der Datei eindeutig sein muss. Wie auch in HTML, werden lokale Namen durch ein vorangestelltes „#“-Zeichen definiert. Danach folgt, wie bei anderen Stildefinitionen auch, die Stil-Anweisungen in geschweiften Klammern.

Die spezielle Stil-Anweisung mit einem Namen ist genauso aufgebaut, wie alle anderen Stil-Anweisungen. Allerdings steht an Stelle der HTML-Anweisung nun der frei definierte Name.

In beliebigen HTML-Anweisungen kann dann diese Stilangabe einmal eingefügt werden. Der Parameter heißt dann *id*, dem der Stilname als Wert mitgegeben wird.

Mit solchen Elemente lassen sich also ganz spezielle Abweichungen vom normalen Stil definieren.

S-EXP06: Arbeiten mit Stil-Ausnahmen

Schreiben oder ändern Sie eine HTML-Datei mit Stil-Anweisungen.

Benutzen Sie den lokalen Name *ausnahme*, um eine Stil-Anweisung zu schreiben. Diese Stil-Anweisung soll in einem beliebigen Container mit dem *id*-Parameter benutzt werden.

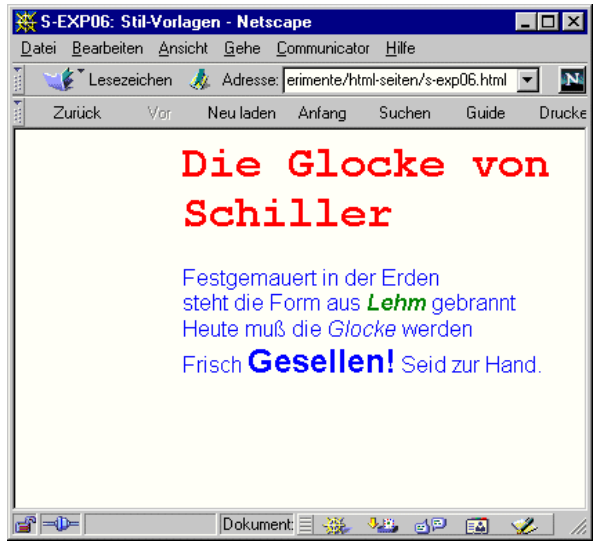

Abbildung 20.6: Stilausnahmen

20.8 Verschachtelung von Stilvorgaben

Der Name der verwendeten Stil-Anweisungen war *cascading style sheets* (kaskadierte Stil Angaben). Was macht eigentlich eine Kaskade aus? Über eine Kaskade läuft Wasser von einer oberen Stufe über verschiedene Zwischenstufen nach unten.

S-EXP07: Kaskadierung von Stilen

In einer HTML-Datei sollen Absätze vorkommen. Mindestens in einem Absatz soll ein Wort betont werden (*em*).

Für die Absätze wird eine Schriftgröße von 16 Punkten (*pt*) festgelegt. Die Einstellung wird an die eingeschlossene Markierung mit `<em>`

weitergegeben. Aus Sicht der Markierung spricht man von erben: die markierte Stelle erbt die umgebende Stileinstellung.

Im Browser sollten alle Buchstaben in der gleichen Schriftgröße dargestellt werden.

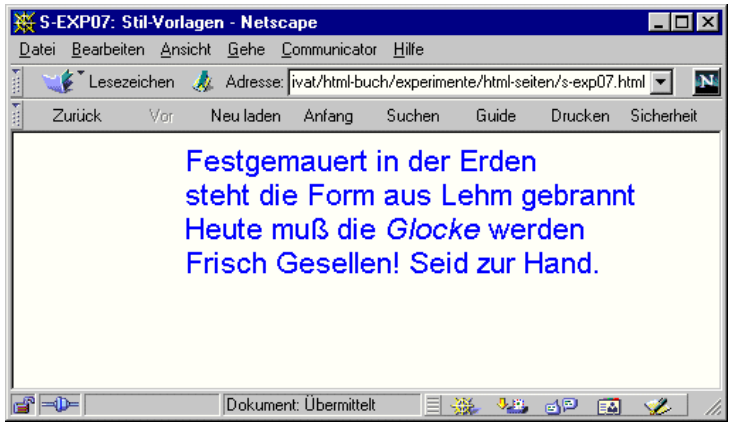

Abbildung 20.7: Kaskadierung

Im einfachen Beispiel floss die Größeninformation aus dem äußeren Absatz-Container in den darin enthaltenen Betonungs-Container. Man kann auch im Stil der *OOP*, der objektorientierten Programmierung sagen, der innere Betonungs-Container hat eine Eigenschaft des umgebenden Containers geerbt.

Diese Spielregel gilt solange, wie nicht eine präzisere Vorgabe erfolgt. Falls Sie für eine *em*-Anweisung jedoch die Schriftgröße getrennt festlegen, gilt diese und nicht die geerbte.

Diese Container-Philosophie lässt sich gut nutzen, um bestimmte Situationen von Container-Verschachtelungen beschreiben.

Man kann die Reihenfolge des Auftretens in Containern festlegen. Dies geschieht durch eine Liste von HTML-Anweisungen vor den geschweiften Klammern. Die Werte der Liste werden hier durch Leerzeichen getrennt.

20.9 Spezielle Container

Gerade für die Stilvorlagen werden zwei Container benützt. Den einen kennen Sie bereits: `<div>`. Mit `<div>` werden Blöcke verschachtelt.

Das passende Gegenstück dazu ist die -Anweisung. Mit kann man beliebige Textstellung für den Zweck der Auszeichnung mit Stilelementen markieren. Ohne Stilelemente hat die -Anweisung keinerlei Auswirkung.

**S-EXP08: Markierung mit **

Schreiben Sie eine HTML-Datei mit Stilvorgaben. Eine -Anweisung soll einige Wörter in einem Satz markieren. Der zugehörige Stil hebt die markierten Wörter hervor.

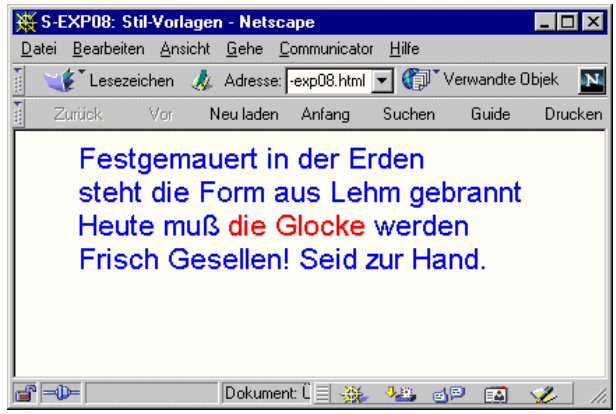

*Abbildung 20.8: Markierung mit *

20.10 Verschachtelte Stil-Anweisungen

Eine der interessantesten Varianten der Stil-Auszeichnungen sind verschachtelte Stile. Betrachtet man Auszeichnungen als Container, dann kann man die Container in einer bestimmten Reihenfolge verschachteln. Die passenden Stil-Auszeichnungen legen dann eine bestimmte Reihenfolge der Auszeichnungen fest. Eine bestimmte Stilauszeichnung wird nur dann wirksam, wenn der beschriebene Stil in der vorgegebenen Reihenfolge durch Container erreicht wird.

Man gibt dazu einfach die Reihenfolge der Auszeichnungen in der Stil-Anweisung an. Reihenfolgen werden durch eine Liste der Auszeichnungen mit Leerzeichen als Trenner angegeben.

Beispiel:

```
p b i {color: green}
```

S-EXP09: Stile für Verschachtelungen

Legen Sie fest, dass alle starken Betonungen, die innerhalb einer <h2>-Überschrift stehen, rot angezeigt werden sollen. Außerdem sollen alle Worte, die innerhalb eines Absatzes zuerst als fett und danach als kursiv ausgezeichnet werden, in grün erscheinen

Nutzen Sie das Container-Modell, um diese Reihenfolge zu beschreiben.

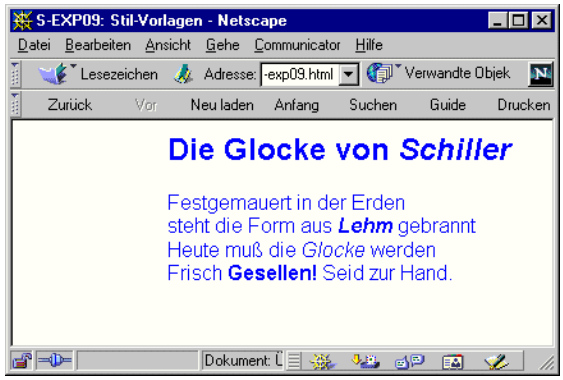

Abbildung 20.9: Verschachtelte Stile

20.11 Stile in HTML-Anweisungen

In fast allen HTML-Anweisungen können Sie einen Stil angeben. Die Stil-Anweisung wird wie ein Parameter mit dem Namen *style* geschrieben. Die Werte stehen dann wie gewohnt nach dem Zuweisungssymbol („=“) in Anführungszeichen.

Beispiel:

```
<h1 style="font-family: monospace; font-size: 14pt">Die Überschrift </h1>
```

Erinnern Sie sich? In der Syntax der Stilanweisungen haben wir erwähnt, dass im Gegensatz zu den HTML-Anweisungen in den Stil-Anweisungen keine Anführungszeichen erlaubt sind. Das Beispiel einer Stilanweisung als Parameter einer HTML-Anweisung zeigt den Grund. Ohne diese Einschränkung wäre es kaum möglich, die verschiedenen Anführungszeichen auseinander zu halten.

Die unmittelbare Stilangabe innerhalb einer HTML-Anweisung ist übrigens die präziseste Angabe. Diese Variante betrachten wir im nächsten Experiment.

S-EXP10: Direkte Stilanweisungen in HTML

Schreiben Sie in einer HTML-Datei, in der HTML-Anweisungen einen Parameter mit Stilanweisungen erhalten.

Beobachten Sie die Auswirkung auf eine geerbte Stilvorgabe.

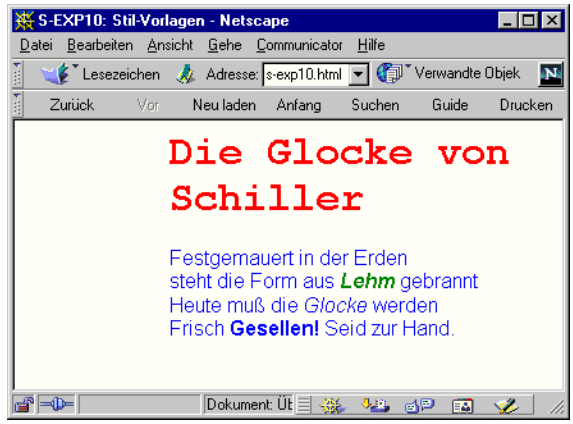

Abbildung 20.10: Direkt Stilanweisungen

Mit den direkten Stilangaben haben Sie die präziseste und damit unbedingt bindende Vorgabe für eine Anweisung gemacht.

20.12 Arbeiten mit Stil-Dateien

In den bisherigen Experimenten haben wir die einzelnen Möglichkeiten der Stilanweisungen kennen gelernt. Aber die große Leistungsfähigkeit des Systems von Stilanweisungen kommt erst zum Tragen, wenn wir alle Stilanweisungen in einer einzigen Datei zusammenfassen.

Das hat zwei Auswirkungen. Einerseits muss man sich vorher Gedanken machen, welche Stile benutzt werden und für was. Es entsteht sozusagen automatisch ein Gesamtkonzept für den Webauftritt einer Firma oder einer Person. Wenn Sie auf Menschen zugehen, die Sie als Ihre Kunden oder Freunde betrachten, dann sollten Stil und Inhalt zusammen einen freundlichen und doch korrekten Eindruck vermitteln.

Jeder, der sich auf andere konzentriert, wird neben dem technischen Aspekt immer auch den gestalterischen Aspekt berücksichtigen. Und als Webautor schreiben Sie für andere.

Also richten wir eine eigene Datei für die Stilvorgaben ein. Dabei kann man alle Anweisungen, die wir bisher in den Kopfteil der HTML-Seite geschrieben haben, in eine zentrale Datei auslagern. In dieser Stildatei fehlt natürlich die HTML-Anweisung `<style>`. Es genügen die *CSS*-Anweisungen.

Innerhalb der HTML-Datei müssen wir nun angeben, welche *CSS*-Datei wir benutzen wollen. Dies geschieht wieder im Kopf der Datei mit der `<link>`-Anweisung.

Beispiel:

```
<head>
...
<link rel="stylesheet" stype="text/css" href="meincss.css">
</head>
```

S-EXP11: CSS-Dateien

Benutzen Sie zwei der vorangegangenen Beispiele und schreiben Sie sie so um, dass nun ohne weitere Änderungen an den Stil-Anweisungen eine *CSS*-Datei benutzt wird. Das optische Ergebnis sollte das gleiche wie bisher sein.

Ändern Sie dann das Erscheinungsbild eines Stilelementes in der zentralen Stil-Datei und beobachten Sie die Auswirkungen auf beide Dateien.

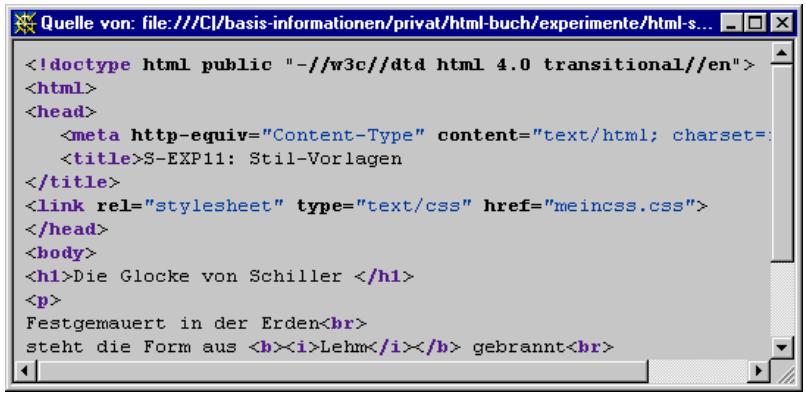

Abbildung 20.11: Zentrale CCS-Datei

Nun können Sie in einer einzigen Datei den Stil (oder wie es auch heißt: die corporate identity) festlegen und alle Ihre Web-Seiten werden diesen Stil berücksichtigen.

So kurz das Experiment geraten ist, so groß kann die Auswirkung auf einen größeren Webauftritt sein. Die Zeit und die Arbeit, die man in die Strukturierung einer Webseite steckt, erhält man als Einsparung im Laufe der Pflegezeit mehrfach wieder zurück.

20.13 Pseudo-Elemente

Am Schluss können wir uns noch ein paar Pseudo-Elemente betrachten. Diese Elemente beschrieben keine HTML-Anweisungen, sondern erlauben die Attribute der Verweise einzustellen.

S-EXP12: Arbeiten mit Pseudo-Elementen

Definieren Sie in einer HTML-Datei Stil-Anweisungen, die die Farben der Verweise in einer HTML-Seite steuern. Pseudoanweisungen werden hinter die HTML-Anweisung mit einem Doppelpunkt gesetzt. Danach folgt wieder die übliche Stil-Anweisung.

Hier im Beispiel geben wir für die `<a>`-Anweisung die Pseudoanweisungen *active*, *link* und *visited* (gedrückt, normaler Verweis und besucht) und setzen jeweils die gewünschte Farbe dahinter.

Da wir uns mit diesen Anweisungen bereits ein Stück im zukünftigen Standard *CSS-2* bewegen, ist es nicht sicher, dass Ihr Browser die Anweisungen korrekt ausführt.

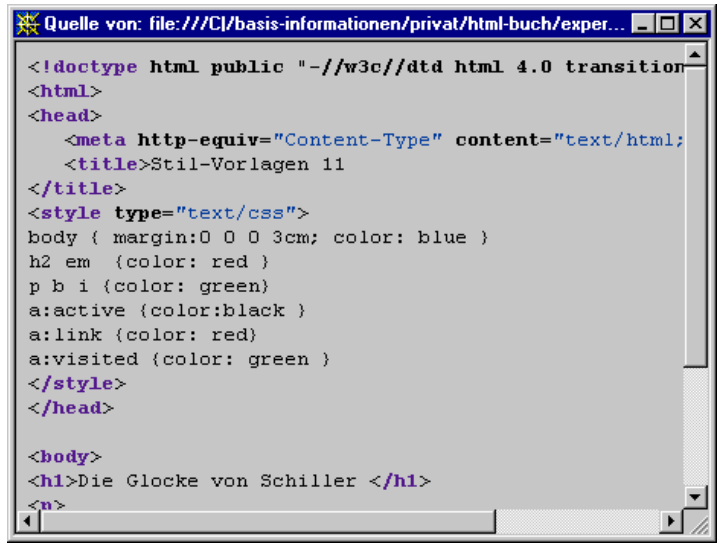

Abbildung 20.12: Setzen von Verweis-Farben

Im nächsten Kapitel

Mit diesem Kapitel über Stile haben wir einen großen Sprung in Richtung eines professionellen Webauftritts gemacht.

Wenn Sie gleichzeitig mit den Experimenten in diesem Buch auch am eigenen Webauftritt gearbeitet haben, dann stehen jetzt HTML-Dateien, Bilder und Stilvorlagen auf Ihrer Festplatte zur Verfügung.

Nun müssen wir sie nur noch auf einen Server stellen, der dauernd mit dem Internet verbunden ist.

20.14 Lösungen

S-EXP01: Stil-Vorgaben für eine HTML-Datei

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>S-EXP01: Stil-Vorlagen</title>
<style type="text/css">
pre {color: rgb(0,0,255) ; }
</style>
</head>
<body>
<h1>
Die Glocke</h1>
<pre>Festgemauert in der Erden
steht die Form aus Lehm gebrannt
Heute mu&szlig; die Glocke werden
Frisch Gesellen! Seid zur Hand.</pre>
<pre>Von der Stirne hei&szlig;,
Rinnen mu&szlig; der Schwei&szlig;,
Soll das Werk den Meister loben,
Doch der Segen kommt von oben.</pre>
</body>
</html>
```

S-EXP02: Randeinstellungen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
```

```
   <meta name="Author" content="Walter Herglotz">
   <title>S-EXP02: Stil-Vorlagen</title>
<style type="text/css">
body { margin:0 0 0 3cm; color: blue }
</style>
</head>
<body>
<pre>Festgemauert in der Erden
steht die Form aus Lehm gebrannt
Heute mu&szlig; die Glocke werden
Frisch Gesellen! Seid zur Hand.</pre>
</body>
</html>
```

S-EXP03: Umrandung von Texten

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>S-EXP03: Stil-Vorlagen</title>
<style type="text/css">
h1 { border-width: 1 3 3 1 }
</style>
</head>
<body>
<h1>
Randangaben</h1>
<pre>Festgemauert in der Erden
steht die Form aus Lehm gebrannt
Heute mu&szlig; die Glocke werden
Frisch Gesellen! Seid zur Hand.</pre>
</body>
</html>
```

S-EXP04: Automatische Formatierungen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>S-EXP04: Stil-Vorlagen</title>
<style type="text/css">
p { text-indent: 2ex }
```

```
h1 {text-transform: capitalize}
</style>
</head>
<body>
<h1>
die glocke</h1>
Festgemauert in der Erden
<br>steht die Form aus Lehm gebrannt
<br>Heute mu&szlig; die Glocke werden
<br>Frisch Gesellen! Seid zur Hand.
<p>Von der Stirne hei&szlig;
<br>Rinnen mu&szlig; der Schwei&szlig;
<br>Soll das Werk den Meister loben<br<
doch der Segen kommt von oben
</p>
</body>
</html>
```

S-EXP05: Klassen für Stile

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>S-EXP05: Stil-Vorlagen</title>
<style type="text/css">
p.einleitung { font-style: italic; }
em.einleitung {color: blue}
p.artikel {font-family: serif}
em.artikel {color: red}
p.literatur {font-style: italic ;font-size:10pt ; font-weight: 600}
</style>
</head>
<body>
<h1>
HTML-Anweisungen</h1>
<h2>
Einleitung</h2>
<p class="einleitung">
Die grundlegende Eigenschaft von HTML ist die Mischung
aus <em class="einleitung">Anweisungen </em>und <em>Steuerbefehlen</em>.</
p>
<h2>
HTML-Lernen</h2>
```

```
<p class="artikel">Mit Hilfe der vielen <em class="artikel">Experimente</
em> erleben Sie
Schritt f&uuml;r die M&ouml;glichkeiten von HTML.</p>
<h2>
Literaturverzeichnis</h2>
<p class="literatur">Die wichtigsten Dokumente sind die Standards von
www.w3c.org.</p>
</body>
</html>
```

S-EXP06: Arbeiten mit Stil-Ausnahmen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <title>S-EXP06: Stil-Vorlagen
</title>
<style type="text/css">
body { margin:0 0 0 3cm; color: blue }
h2 em  {color: red }
p b i {color: green}
#extra {font-size: 18pt }
</style>
</head>
<body>
<h1 style="font-family: monospace; color: red">Die Glocke von Schiller </
h1>
<p>
Festgemauert in der Erden<br>
steht die Form aus <b><i>Lehm</i></b> gebrannt<br>
Heute muß die <em>Glocke</em> werden<br>
Frisch <b id="extra">Gesellen!</b> Seid zur Hand.
</p>
</body>
</html>
```

S-EXP07: Kaskadierung von Stilen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <title>S-EXP07: Stil-Vorlagen
</title>
<style type="text/css">
body { margin:0 0 0 3cm; color: blue }
p { font-size: 16pt }
```

```
</style>
</head>
<body>
<p>
Festgemauert in der Erden<br>
steht die Form aus Lehm gebrannt<br>
Heute muß die <em>Glocke</em> werden<br>
Frisch Gesellen! Seid zur Hand.
</p>
</body>
</html>
```

S-EXP08: Markierung mit

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <title>S-EXP08: Stil-Vorlagen
</title>
<style type="text/css">
body { margin:0 0 0 1cm; color: blue }
p { font-size: 16pt }
span {color:red}
</style>
</head>
<body>
<p>
Festgemauert in der Erden<br>
steht die Form aus Lehm gebrannt<br>
Heute muß <span>die Glocke</span> werden<br>
Frisch Gesellen! Seid zur Hand.
</p>
</body>
</html>
```

S-EXP09: Stile für Verschachtelungen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>S-EXP09: Stil-Vorlagen</title>
<style type="text/css">
body { margin:0 0 0 1cm; color: blue }
h2 strong  {color: red }
```

```
p b i {color: green}
</style>
</head>
<body>
<h2>
Die Glocke von <strong>Schiller</strong></h2>
<p>Festgemauert in der Erden
<br>steht die Form aus <b><i>Lehm</i></b> gebrannt
<br>Heute mu&szlig; die <i>Glocke</i> werden
<br>Frisch <b>Gesellen!</b> Seid zur Hand.
</p>
</body>
</html>
```

S-EXP10: Direkte Stilanweisungen in HTML

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <title>S-EXP10: Stil-Vorlagen
</title>
<style type="text/css">
body { margin:0 0 0 3cm; color: blue }
h2 em  {color: red }
p b i {color: green}
</style>
</head>
<body>
<h1 style="font-family: monospace; color: red">Die Glocke von Schiller </
h1>
<p>
Festgemauert in der Erden<br>
steht die Form aus <b><i>Lehm</i></b> gebrannt<br>
Heute muß die <em>Glocke</em> werden<br>
Frisch <b>Gesellen!</b> Seid zur Hand.
</p>
</body>
</html>
```

S-EXP11: CSS-Dateien

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <title>S-EXP11: Stil-Vorlagen
</title>
```

```
<link rel="stylesheet" type="text/css" href="meincss.css">
</head>
<body>
<h1>Die Glocke von Schiller </h1>
<p>
Festgemauert in der Erden<br>
steht die Form aus <b><i>Lehm</i></b> gebrannt<br>
Heute muß die <em>Glocke</em> werden<br>
Frisch <b>Gesellen!</b> Seid zur Hand.
</p>
</body>
</html>
```

S-EXP12: Arbeiten mit Pseudo-Elementen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <title>Stil-Vorlagen 11
</title>
<style type="text/css">
body { margin:0 0 0 3cm; color: blue }
h2 em  {color: red }
p b i {color: green}
a:active {color:black }
a:link {color: red}
a:visited {color: green }
</style>
</head>
<body>
<h1>Die Glocke von Schiller </h1>
<p>
Festgemauert in der Erden<br>
steht die Form aus <b><i>Lehm</i></b> gebrannt<br>
Heute muß die <em>Glocke</em> werden<br>
Frisch <b>Gesellen!</b> Seid zur Hand.
</p>
<a href="s-exp05.html"> Ende des Gedichtes</a>
</body>
</html>
```

Pearson Education

ADDISON-WESLEY

21 Publizieren im Netz

Bisher haben wir unsere Dateien lokal auf der eigenen Festplatte abgelegt. Für alle, die keinen dauerhaften Zugang zum Webserver besitzen, ist das sicher eine elegante Lösung.

Da es aber inzwischen recht preiswert ist, haben vielleicht auch schon viele eine eigene Homepage im Netz. Nun bleibt nur noch der Transport der Dateien auf einen Webserver.

Für kleine Webauftritte wird meist das FTP-Programm benutzt, um Dateien über das Netz zu kopieren. FTP-Programme gibt es in zwei verschiedenen Varianten: einer Kommandozeilen-Variante, die meist bereits zusammen mit dem Betriebssystem mitgeliefert wird. Der Programmaufruf ist meist einfach *ftp*. Dies gilt übrigens auch unter Windows.

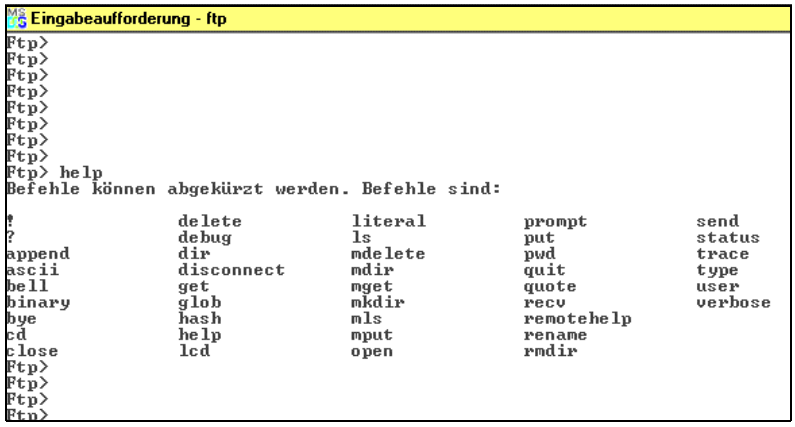

Abbildung 21.1: Befehle der Kommandozeilenversion

Man kann sich das Leben etwas leichter machen und ein Programm mit graphischer Oberfläche benutzen. Unter Windows gibt es z.B. *WS_Ftp*.

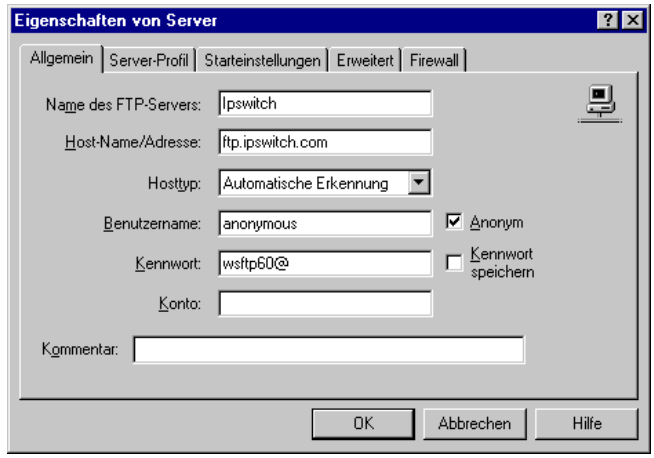

Abbildung 21.2: Verbindungseinstellung im FTP-Programm

Eine Verbindung konfiguriert man im *Eigenschaften*-Fenster. Für die Übertragung öffentlich zugänglicher Daten genügt eine anonyme Anmeldung. Der Benutzername ist dabei *anonymous*. Als Passwort wird erwartet, dass man seine eigene E-Mail-Adresse angibt. Dies muss im Beispielfenster noch geschehen.

Die Bedienung der Programme ist meist sehr einfach. Letztlich müssen ja nur Dateien 1:1 kopiert werden. Für *WS_Ftp* gibt es sogar eine Oberfläche, die wie ein Teil der Dateiverwaltung aussieht.

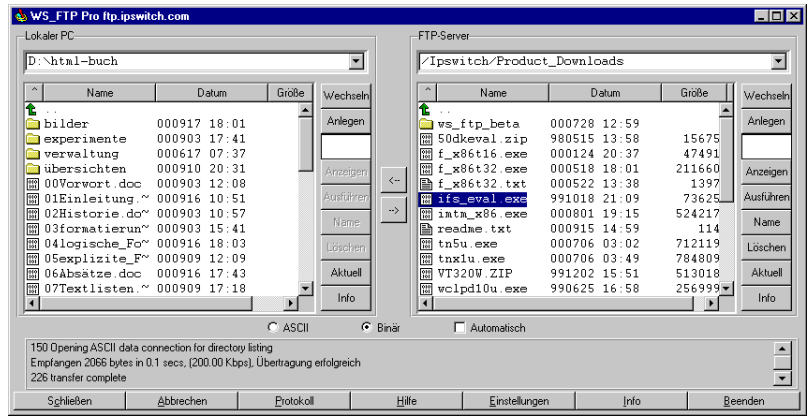

Abbildung 21.3: Übertragungssteuerung mit FTP

Man wählt zur Übertragung jeweils auf dem eigenen und dem entfernten Rechner das gewünschte Inhaltsverzeichnis aus. Dann kann man Dateien markieren und mit den mittleren Pfeilen kopieren.

Das Kopieren ist also nicht das Problem – bis auf eine Einstellung. Verwenden Sie immer die binäre Übertragung, schließlich sind die 7-Bit-ASCII-Zeiten vorbei. Werfen Sie auch einen Blick in die allgemeinen Einstellungen des Programms. Oft gibt es hier eine Zuordnungstabelle, die abhängig von der Dateiendung das Übertragungsverfahren festlegt. Manche Programme schalten sogar automatisch auf 7-Bit-Übertragung, wenn die Dateiendung *.HTM oder *.HTML ist.

Auch hier sollte keinesfalls die 7-Bit-Übertragung für ASCII eingestellt werden.

21.0.1 Fallstricke beim Publizieren

Der häufigste Problemfall sind die Dateinamen. Die Server im Netz laufen häufig unter Linux/Unix. Dieses Betriebssystem beachtet Groß- und Kleinschreibung. Unter anderen Betriebssystemen spielt die Schreibweise der Dateinamen keine Rolle. Man sollte dann als Autor sich auf eine Schreibweise festlegen. Meist wird man alle Dateinamen klein schreiben.

Was aber für Autoren wirklich sehr unangenehm werden kann, ist die Marotte von manchen Betriebssystemen eigenmächtig den Namen zu verändern. Manchmal werden automatisch Großbuchstaben an den Anfang des Dateinamens gestellt. Rechner sind Maschinen – und sollten sich so verhalten.

Durch solche unangebrachten Dienstleistungen handelt man sich immer wieder Fehler bei den Verweisen ein.

21.1 Pfadnamen

Pfadnamen sind innerhalb einer Gruppe von Webseiten, die untereinander mit Verweisen verbunden sind, am besten immer relativ.

Funktionieren die Verweise zwischen einer Gruppe von Webseiten lokal, dann funktionieren sie auch auf dem Server, wenn man von den beschriebenen Fallstricken absieht.

Es gibt aber auch gute Gründe, Verweise mit dem absoluten Pfadnamen der Seite aufzunehmen. Manche Autoren möchten sicherstellen, dass immer die aktuelle Seite benutzt wird, auch wenn jemand sich die Seite lokal auf seinen Rechner kopiert hat. Dann zeigt der Verweis immer auf den Webserver.

21.2 Registrieren in Suchmaschinen

Suchmaschinen durchkämmen das WWW an Hand der gefundenen Verweise. Wenn nun aber niemand einen Verweis auf Ihre Webseiten gesetzt hat, dann sind die Chancen gering, dass eine Suchmaschine Ihre Seiten findet.

Die meisten Suchmaschinen haben daher die Möglichkeit geschaffen, Webseiten selbst zu registrieren. Mit einer Registrierung ist heute auch durchaus eine redaktionelle Bearbeitung verbunden, da Suchmaschinen ungern auf nicht akzeptable Inhalte verweisen möchten.

Nach einigen Tagen sollten Sie dann Ihre Seiten auch in der Suchmaschine wiederfinden.

Wie im Kapitel über die `<meta>`-Anweisungen besprochen, haben sich Angaben über Suchbegriffe und eine Beschreibung der Seite bewährt.

Im nächsten Kapitel

An dieser Stelle stehen nun unsere Seiten auf dem Webserver und damit allen Interessierten zur Verfügung. Noch sind es aber passive Seiten. Um Seiten dynamisch zu gestalten, benötigen wir Programme. Eine Variante der Programme kennen wir bereits im Zusammenspiel mit den sensitiven Bildern. Programme können aber auch bereits in den Webseiten enthalten sein. Damit lässt sich die Schnittstelle zwischen dem Menschen und der Maschine deutlich ausbauen.

Wenden wir uns der dafür zumeist verwendeten Sprache *Javascript* zu.

22 Javascript

22.1 Programmieren für den Browser

Nun haben wir eine schöne und gut gestaltete Webseite, mit all den Möglichkeiten der Formatierung und der Stile. Aber die Seite bleibt ruhig, nichts rührt sich, mit Ausnahme der animierten Graphiken.

Dabei gäbe es noch weit mehr zu tun: Eingaben in Formulare prüfen oder Bilder austauschen, wenn wir mit der Maus darüber fahren.

Dazu müssen wir Programme auf dem Browser ausführen können, denn alle Reaktionen auf unsere Eingaben oder sonstige Tätigkeiten müssen durch Programme ausgewertet werden.

Mit HTML 4.0 (und einigen Browser Versionen davor) kam Leben in die Webseiten. In vielen Anweisungen können wir nun Ereignisse beschreiben und angeben, welche Reaktion denn auf ein bestimmtes Ereignis erfolgen soll. In HTML gibt es also ein Reiz-Reaktions-Modell der Programmierung.

22.2 Programmiersprachen für HTML-Programmierung

Die Überschrift ist vielleicht nicht ganz richtig. Denn da wird der Plural benutzt. Natürlich gibt es viele Sprachen, aber eine Sprache im WWW macht nur Sinn, wenn sie unabhängig von irgendeinem Hersteller ist. Also bleibt doch nur eine einzige globale Sprache, die nebenbei noch eine interessante Geschichte hat: *Javascript*.

Der Name sagt es: es ist eine Script-Sprache, die irgendwie der Sprache Java ähnelt. Scriptsprachen bestehen schlicht aus Text, der erst bei der Ausführung von einem so genannten Interpreter gelesen und eben interpretiert werden muss. Auf der weiten Skala der Ausführungs-

geschwindigkeiten sind wir damit in der Unterliga. Aber das macht nichts. Denn unsere Sprache muss ja keine großen Leistungen vollbringen. Ab und zu ein Bild austauschen, ein paar Tests machen und auch mal auf Mausklicks reagieren.

Die zweite Sprache für das WWW und auch für alle anderen Zwecke, ist Java. Dieser Sprache und den notwendigen Steueranweisungen, um von HTML aus mit Java-Programmen umgehen zu können, ist ein eigenes Kapitel gewidmet.

Und außerdem verändert sich die Hardware so schnell, dass ein Programm, das uns heute langsam vorkommt, in zwei Jahren schon schnell läuft.

22.3 Definition von Ereignissen

Bleiben wir einen Moment bei HTML. In HTML werden die Teile einer Seite beschrieben, die Ereignisse auslösen können. Die einfachsten sind die Sende- oder Lösch-Knöpfe der Formulare. Aber auch Bilder stehen als Lieferanten eines Ereignisses zur Verfügung. Hier können Mausbewegungen über der Oberfläche oder ein Klick als Ereignis gewertet werden.

Ein Ereignis wird durch einen Parameter beschrieben, der die passende Reaktion als Wert erhält. In der Anweisung, die nun eine dynamische Reaktion zeigen soll, gibt man als Parameter den gewünschten Reiz an. Beispiele sind: *onclick* (wenn man darauf klickt), *onmouseover* (wenn die Maus darüber geführt wird) oder *onsubmit* (wenn ein Formular abgeschickt wird).

Wieder einmal kommen wir zum Thema der Schreibweise. Es hat sich in den meisten englischsprachigen Unterlagen eingebürgert, dass die ersten Buchstaben der Worte innerhalb des Reiznamens groß geschrieben werden (also als Beispiel *onClick*). Sie werden dadurch leichter lesbar. Genauso wie die Groß- und Kleinschreibung der deutschen Sprache das Erfassen von Texten erleichtert. Solange wir uns im bisherigen HTML-Bereich bewegen, können Sie den Namen eines Parameters nach Belieben schreiben. Aber eben nur im alten HTML. Für die Programmierung und für die kommenden Standards gilt eindeutig: nur Kleinschrift erlaubt.

22.4 Programmieren mit Javascript

Nun müssen wir nur noch die Reaktion beschreiben. Und damit – so fürchte ich – verlassen wir den Themenbereich des Buches. Programmieren lernen ist genauso sinnvoll wie Autofahren lernen. Es ist sicherlich eine absolut sinnvolle und notwendige Basisausbildung für jeden, der in der IT-Welt zu Hause ist. Wer hier nicht wenigstens ein wenig mit einer Programmiersprache umgehen kann, dem fehlt ein wichtiger Schlüssel zum Verständnis der Datentechnik.

Der Platz wird aber nicht reichen, um in diesem Buch eine solide Darstellung einer Programmiersprache oder eine Grundausbildung zum Programmierer zu schaffen.

Aber ein paar Eindrücke von den Möglichkeiten der Programmierung im Browser wollen wir uns doch verschaffen.

22.5 Funktionen

Wirth, der Erfinder der akademischen Programmiersprache Pascal, hat einmal ein Buch geschrieben mit dem schönen Titel „Daten + Algorithmen = Programme“. Algorithmen (Rechenvorschriften) bestehen aus vielen einzelnen Schritten. Und ein einzelner Schritt heißt Befehl. Befehle sind die kleinsten Aktionen in einer Computersprache.

Zu den Daten, den vielen Elementen einer HTML-Seite, kommen wir noch.

Um es nicht zu schwer zu machen, begnügen wird uns mit einigen Auszügen aus der Welt von Javascript. Damit wir es möglichst einfach halten, schreiben wir die Javascript-Anweisung direkt als Wert des Reiz-Parameters. Das kann sinnvollerweise nur ein einzelner Javascript-Befehl machen.

Dazu müssen wir dem Javascript-Befehl den Namen der Sprache gefolgt von einem Doppelpunkt voranstellen. Der Befehl wird mit einem Strichpunkt abgeschlossen. Werfen Sie auch einen Blick auf die Verschachtelung der beiden Anführungszeichen.

Beispiel:

```
onmouseover="alert('Fenster ist nun offen');"
```

J-EXP01: Reiz – Reaktion

Schreiben Sie eine HTML-Datei und verwenden Sie darin einen Verweis auf eine andere HTML-Seite. Wenn Sie mit der Maus ohne zu klicken über den Verweis fahren (*onmouseover*), soll ein Fenster aufgehen und einen Text ausgeben ("alert('Angezeigter Text');").

Beachten Sie, dass der Parametertext der *alert()*-Funktion in einfachen Anführungszeichen gesetzt wurde. Das einfache Anführungszeichen findet sich auf deutschen Tastaturen auf der „#“-Taste.

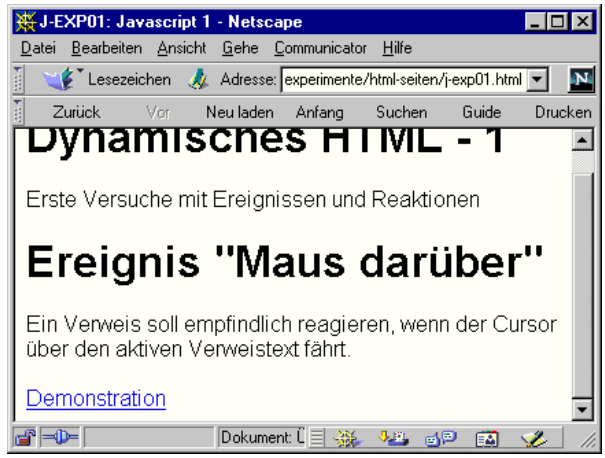

Abbildung 22.1: Reaktion auf Mausberührung

Wenn das kleine Programm korrekt abläuft, dann sehen Sie ein Meldungsfenster aufgehen, das den gewünschten Text anzeigt. Damit Sie nicht in die Irre geführt werden und vielleicht denken, dass dieses Fenster eine Meldung des Betriebsystems ist, steht in der Kopfzeile der Hinweis, dass es ein Javascript-Fenster ist.

Das Beispiel war sicher gut, um einen ersten, schnellen Erfolg zu bekommen. Aber eine richtige Programmierung war es noch nicht. Schreiben wir daher das Beispiel um. Anstelle den Befehl direkt einzugeben, soll nun eine Funktion aufgerufen werden.

22.5.1 Funktionen in Javascript

Funktionen sind Programmstücke, die einen eigenen Namen besitzen. Der Name bestimmt den Startpunkt der Befehle in der Funktion. Beim Aufruf schreibt man den Namen der Funktion und danach einen Satz von runden Klammern. In den Klammern können Werte

stehen, die an die Funktion übergeben werden. Die übergebenen Werte nennt man *aktuelle Parameter*.

Nach dem Start der Befehle einer Funktion werden diese abgearbeitet und am Ende kehrt die Funktion an die Aufrufstelle zurück. Bei der Bearbeitung der Reaktion auf einen Reiz ist damit der Ablauf beendet. Es ist wie eine eingefügte Sprechblase in einem Asterix-Heft.

Eine der einfachsten Funktionen in Javascript heißt *alert()*. Man kann ihr einen Text mitgeben. Damit der Interpreter erkennt, wo Anfang und Ende des Textes ist, rahmt man den Text entweder in einfache oder doppelte Anführungszeichen ein. Vielleicht benutzen Sie, wenn es möglich ist, bei Javascript eher die einfachen Anführungszeichen, um die doppelten lieber HTML zu überlassen.

In Büchern werden meist proportionale Schriften verwendet. Beim Programmieren sollten Sie auf die in Büchern übliche Unterscheidung zwischen Anfangs- und Ende-Anführungszeichen verzichten. Hier gibt es nur die normalen Anführungszeichen, die immer oberhalb des Textes stehen.

22.5.2 Funktionen in HTML

Nun müssen wir einen Platz suchen, um die Funktion zu schreiben. Der beste Platz ist an einer Stelle, die der Browser sicher gelesen hat, bevor er mit der Anzeige des Inhaltes beginnt und vor dem Inhalt, der in der `<body>`-Anweisung, steht, kommt die `<head>`-Anweisung.

Also schreiben wir die Funktion in der `<head>`-Anweisung der HTML-Seite. Programme schreibt man innerhalb der `<script>`-Anweisung. Wie erwähnt, können Skripten in verschiedenen Sprachen geschrieben werden. Allerdings bleibt immer die Frage, ob alle Browser die benutzte Sprache verstehen. Und eben deshalb ist Javascript die Sprache der Wahl, da sie von den allermeisten Browsern verstanden wird.

Der Kern der Sprache ist auch in einem internationalen Standard der *ECMA* (European Computer Manufacturers Association / Vereinigung europäischer Computerhersteller) genormt. Gelegentlich werden Sie daher auch den Begriff *ECMA-Script* lesen können.

Der `<script>`-Anweisung muss man jedoch mitteilen, welche Sprache gewünscht wird. Dies geschieht derzeit mit zwei verschiedenen Parametern. Der eine heißt *language* und ist optional. Der zweite heißt *type* und muss angegeben werden. In älteren Browsern wird nur der *language*-Parameter erkannt, ab HTML 4.0 ist *type* verpflichtend. Für *language* genügt die Angabe des Sprachnamens (hier: javascript), für *type* wird ein -*MIME*-Typ angegeben (hier: text/javascript).

Der *language*-Parameter akzeptiert auch Versionsunterscheidungen. Man kann gezielt *javascript1.1* oder *javascript1.2* anfordern. Den Browsern unbekannte Angaben werden wie immer überlesen. Der Nachteil dieser Angaben ist es, dass es keine Gruppe gibt, die die Angaben standardisiert. Für die Angaben des *type*-Parameters werden im Gegensatz dazu *MIME*-Typen benutzt, die einer Standardisierung unterliegen.

Mit mehreren `<script>`-Anweisungen und unterschiedlichen Versionsangaben, lassen sich Programmteile schreiben, die unterschiedliche Versionen benutzen. Wenn möglich, sollte man darauf verzichten. Wir werden außerdem noch eine allgemeingültige Methode kennen lernen, wie man die vorhandenen Möglichkeiten testen kann.

Als Inhalt der `<script>`-Anweisung können wir nun unser Programm schreiben. In den meisten Fällen werden wir Funktionen schreiben, die dann als Reaktion auf einen Reiz aufgerufen werden können.

Geben wir der Funktion einen beliebigen Namen – sagen wir *herbert* zu ihr. Außerdem sollte sie einen Wert erhalten. Die Werte für die Funktion nennen wir beim Schreiben des Unterprogramms *formale Parameter*.

Die Begriffe sind für den Moment gar nicht wichtig, wenn nur spätestens in den Beispielen die Verwendung verständlich wird. Im Zweifel werden wir aber einfach die richtigen Fachbegriffe verwenden.

Und damit sich der Interpreter auskennt, setzen wir den Befehl *function* vor den Namen unseres Unterprogrammes *herbert* und lassen in runden Klammern Namen für die übergebenen Werte folgen. Die Namen können wir wieder frei festlegen. Die Befehle, die zum Unterprogramm gehören, werden durch ein geschweiftes Klammerpaar zusammen gehalten.

Für einen Programmierer, der sehr viel von *C/C++* gelernt hat, ist es ziemlich vergnüglich zu sehen, dass fast alle Sprachen, die danach entwickelt wurden, sich möglichst immer an die praktischen Vorgaben von *C* halten. Dazu gehören auch die geschweiften Klammern.

Als einzigen Befehl rufen wir wieder das *alert()*-Unterprogramm auf. Nur geben wir ihm diesmal nicht einen konstanten sondern einen variablen Text mit. Das ist derjenige, den wir beim Aufruf übergeben. *x-wert* ist also eine Variable, die unterschiedliche Inhalte bekommen kann.

Beispiel:

```
<script language="javascript">
function herbert(x-wert)
{
alert(x-wert);
}
</script>
```

Um die Funktion an einer Stelle in einem Programm oder als Reaktion auf einen Reiz aufzurufen, kann man einfach den Namen schreiben, der von einem runden Klammerpaar gefolgt wird. In die runden Klammern schreiben wir, was die Funktion erhalten soll. Hier nehmen wir wieder einen Text.

Den Schluss markieren wir mit einem Strichpunkt (auch Semikolon genannt).

```
herbert('Hallo Herbert');
```

Der hier angegebene Text (‚Hallo Herbert') wandert beim Aufruf in die Variable *x-wert*. Und dort steht sie zur Verfügung; wenn wir sie brauchen. Gebraucht wird sie nur zur Übergabe an die schon bekannte *alert()*-Funktion.

J-EXP02: Aufruf einer Javascript-Funktion

Schreiben Sie eine HTML-Datei, die in der `<head>`-Anweisung eine `<script>`-Anweisung mit einer Funktion enthält.

Die Funktion soll einen Parameter erhalten. In der Funktion rufen wir *alert()* auf und geben den erhaltenen Parameter weiter.

Weiter soll die HTML-Datei einen Verweis enthalten. Wenn der Mauszeiger über dem Verweis steht, soll ein Fenster mit dem Text „Hallo Herbert" aufgehen.

Abbildung 22.2: Reiz und Reaktion mit Funktion

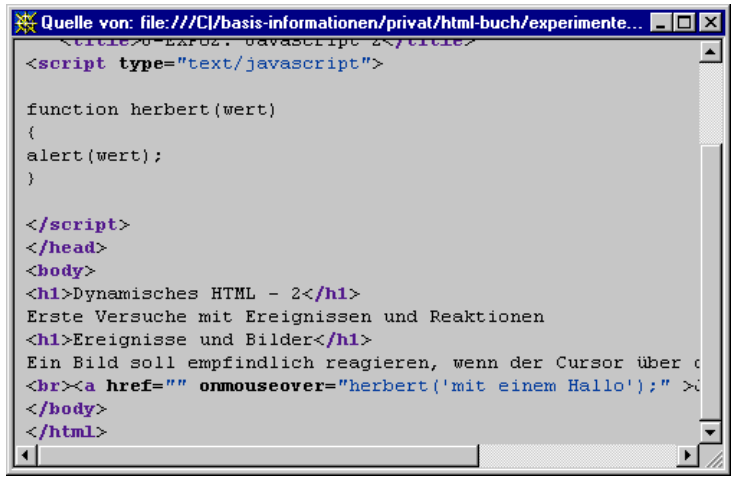

Abbildung 22.3: Quellcode

22.6 DOM – Document Object Model (Modell für Objekte des Dokumentes)

Wenn Sie das erste und zweite Beispiel erfolgreich geschrieben haben, dann ist ein Fenster aufgegangen, wenn Sie mit dem Cursor über den Verweis gefahren sind. Wir haben hier auf eine möglichst einfache und nachvollziehbare Art den ersten Schritt zur Verwendung von Javascript getan.

Das kleine Programm bestand hier nur aus einem einzigen Aufruf einer Funktion, die einen Text erhält. Und die Daten bestanden letztlich nur aus einem konstanten Text, den die Funktion anzeigen sollte.

Aber unser Browser kann auf alle Elemente einer Webseite zugreifen und sie u.U. verändern.

Beim Lesen einer Webseite, die aus der Sicht der Programmiersprache Javascript als *document* bezeichnet wird, baut der Browser einen Baum der einzelnen Teile der Seite auf. Es ist ähnlich wie bei einem Mobile. Es hängt an einem Punkt und von dort aus kann man über die verschiedenen Queräste und kleinen Fäden jedes der einzelnen Teile erreichen.

Die Strukturierung einer Webseite nach einem hierarchischen Muster nennt man das *DOM (document object model / Dokumentenaufbau mit Objekten)*.

Ausgehend von der obersten Ebene, dem *window*, kann man auf die darin enthaltenen Elemente, wie z.B. das angezeigte Dokument, und die darin enthaltenen Sub-Elemente zugreifen. Die einzelnen Komponenten nennt man dazu beim Namen und verbindet sie mit einem Punkt.

Beispiel:

```
window.document.images[„bild2“].src = bildfeld[2].src;
```

Dieses Modell erlaubt es, eine Seite zu laden und nachträglich wieder zu ändern. Das bekannteste Beispiel ist vielleicht der Austausch von Bildern, während der Cursor über einem Bild steht. So kann man durch den Austausch dem Benutzer anzeigen, welcher Auswahlpunkt gerade aktiv ist.

Leider trügt der schöne Schein hier wieder. Zwar gibt es Standardisierungsbemühungen, aber leider wird denen nicht immer gefolgt. Der eine Browser-Hersteller bemüht sich immer um eigene Verbesserungen, der andere wartet ab, bis der ganz neue eigene Browser herauskommt.

Und außerdem gibt es Sprachelemente, die gar nicht im Standard vorkommen. Und trotzdem häufig benutzt werden.

22.7 Dynamisches HTML

Unter dem Schlagwort DHTML werden alle möglichen Standards und Vorschläge zu Marketing-Zwecken zusammengefasst. Was genau DHTML ist, kann man kaum prägnant beschreiben. Vermutlich ist es HTML 4.0, Javascript und Java, Server und dynamische Generierung von Web-Seiten. Also alles das, was wir in diesem Buch zumindest in wesentlichen Punkten beschreiben.

Das erste Javascript-Beispiel soll zeigen, wie eine Web-Seite auf dem Browser dynamisch angereichert wird. Neben vorhandenen Texten, soll die Web-Seite Informationen über den Browser des Benutzers anzeigen. Die Informationen über den Browser des Benutzers stehen logischerweise erst während des Ladens der Seite zur Verfügung. Die Browser-Informationen müssen also dynamisch ergänzt werden.

Man kann dazu an einer beliebigen Stelle im `<body>`-Teil der HTML-Seite *Javscript*-Programme einbinden, die z.B. HTML-Code generieren. Der fertige HTML-Code und der dynamisch zugemischte ergeben dann zusammen die Seite, die der Benutzer sieht.

Die folgenden Zeilen könnten irgendwo im `<body>`-Bereich stehen:

```
<script language="javascript" type="text/javascript">
document.write("- die Browser Version: <b>" + navigator.appVersion +"</
b><br>");
document.write("- den Browser-Hersteller: <b>" + navigator.appName + "</
b><br>");
document.write("- den Spitznamen: <b>" + navigator.appCodeName + "</
b><br>");
document.write("- die Plattform: <b>" + navigator.platform + "</b>.");
</script>
```

Mit den Zeilen, die in den normalen HTML-Code eingestreut werden können, wird festgelegt, dass zwischen der Start- und der Ende-Marke der <script>-Anweisung Befehle einer bestimmten Scriptsprache stehen.

Mit der Methode *write()* aus dem *document* der Web-Seite kann man dynamisch HTML-Code beim Laden einfügen. Texte in Anführungszeichen werden ausgegeben, ein „+“ Zeichen verbindet Texte und bestimmte Informationen. Informationen über den Browser stehen unter *navigator*.

Die einzelnen Elemente in dem Objekt, das man unter *navigator* erreichen kann, sind von Browser zu Browser leider wieder unterschiedlich. Im WWW finden sich daher ganze Web-Seiten, die sich abmühen, eine allgemeingültige Programmierung zu finden, mit der man exakt den verwendeten Browser und seine jeweiligen Marotten erkennen kann.

J-EXP03: Browser-Informationen

Schreiben Sie eine Webseite, die einen einführenden Text und die oben angegebenen Zeilen im `<body>`-Bereich enthält.

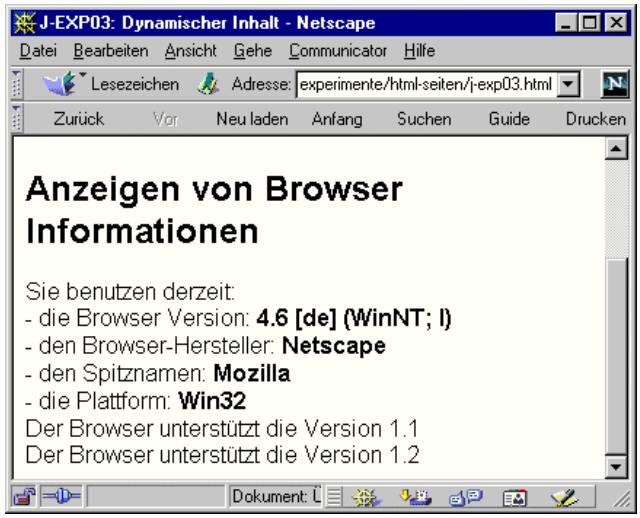

Abbildung 22.4: Infos zum Browser

22.8 Formulare und Javascript

Formulare realisieren in HTML die Benutzerschnittstelle. Im Zusammenhang mit Javascript ergeben sich zwei grundlegende Fragestellungen: wie arbeiten die beiden zusammen und welche zusätzlichen Vorteile können wir als Autoren oder Benutzer einer Sprache im Browser beim Anwenden der Formulare erwarten.

22.8.1 Benutzerschnittstellen mit Formularen und Javascript

Beim Besuch von Web-Auftritten (z.B. von Buchhändlern) finden wir häufig auf der Startseite ein kleines Eingabefeld, in dem wir einen Suchbegriff eingeben können. Manchmal müssen wir dann einen Knopf anklicken, um die Frage abzusenden, manchmal genügt aber auch einfach die Eingabetaste.

Besonders wichtig ist es innerhalb von Web-Seiten mit Formularen, die eingegebenen Werte zu prüfen. Dazu kann man das Ereignis *onblur* (Verlust des Fokus) nutzen. Wenn Sie in einem Eingabefeld einen Wert eingegeben haben und danach zum nächsten Feld gehen, dann verliert das vorhergehende Eingabefeld den Fokus. Damit wird ein Reiz ausgelöst, der genutzt werden kann, um eine Testroutine für den eingegebenen Wert aufzurufen.

Letztlich werden am Schluss nur noch geprüfte Werte übertragen. Die Frage ist oft nur, hat der Benutzer zumindest formal richtige Angaben gemacht? Manche Angaben kann man überprüfen. Eine Internet E-Mail-Adresse muss einen bestimmten Aufbau haben. Es muss ein @-Symbol geben. Davor können ein oder zwei Wörter stehen, die bei Bedarf durch einen Punkt getrennt werden. Und danach müssen mindestens zwei Wörter stehen, die ebenfalls durch einen Punkt getrennt sind. Weiter dürfen in E-Mail-Adressen keine Zeichen außerhalb des amerikanischen ASCII-Zeichensatzes vorkommen.

Das Beispiel mit den E-Mail-Adressen kann man sicher auch auf andere Eingaben ausweiten. So kann es Pflicht- oder Kann-Eingaben geben. Die Frage ist nur, wer macht wann die Überprüfung? Ein Benutzer, der ein großes Formular eingeben hat und es zum Server schickt, wird kaum erfreut sein, wenn als Antwort zurückkommt, dass er einen Eingabefehler gemacht hat und nun die ganze Eingabe wiederholen muss.

Natürlich kann man auch benutzerfreundliche Antworten geben. Wer auch immer das Formular auswertet, sollte dem Benutzer wenigstens die bisher gemachten Eingaben wieder zur Verfügung stellen, damit er nur einen kleinen Teil verändern muss.

Aber zurück zur Frage: wer macht die Überprüfung? Mit Javascript können wir Programme auf dem Browser ausführen. Meist sind die Rechner der Anwender weit weniger belastet, als der zentrale Server. Es wäre daher sinnvoll, wenn ein Javascript-Programm sozusagen „vor Ort“ die Überprüfung übernimmt.

J-EXP04: Testen von Werten aus Formularen

Schreiben Sie eine HTML-Datei, die ein Formular enthält. Eines der Eingabefelder erhält die Angabe eines Reiz-Reaktionspaares. Wenn der Fokus aus dem Feld genommen wird, dann soll ein kleines Hinweisfenster aufgehen.

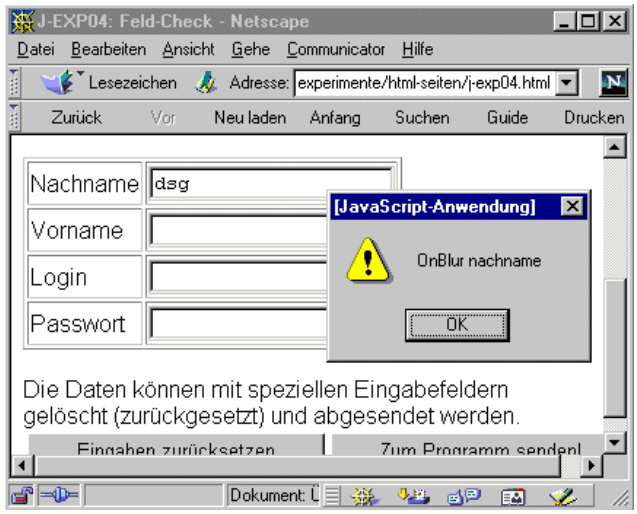

Abbildung 22.5: Reaktion auf einzelne Feldeingaben

22.9 Nutzung spezieller Eingabeelemente

Bei der Diskussion verschiedener Eingabetypen innerhalb von Formularen mussten wir einen Eingabetyp aussparen: den Typ *button*. Das Problem dabei war, dass dem Typ kein Eingabefeld zugeordnet werden konnte. Im Gegensatz dazu hat beispielsweise eine Texteingabe ein Eingabefeld, das dem Benutzer die Eingabe ermöglicht.

Im Gegensatz dazu können Sie auch ein Eingabeelement definieren, dass als Typ *button* hat und die Reaktion mit Hilfe des *onclick*-Reizes und mit Hilfe einer Javascript-Funktion bearbeitet.

J-EXP05: Benutzung des Eingabetyps button mit Javascript

Schreiben Sie eine HTML-Seite, die ein Formular mit einer Eingabeanweisung mit dem Typ *button* enthält. Die Reaktion auf das Anklicken definieren Sie mit dem *onclick*-Reiz und einer Javascript-Funktion. Natürlich genügt für Lernzwecke der Aufruf der *alert()*-Funktion.

Zum Gestalten des Knopfes stehen Höhen- und Breitenangaben (*height* und *width*) sowie die Angabe der Aufschrift mit *value* zur Verfügung.

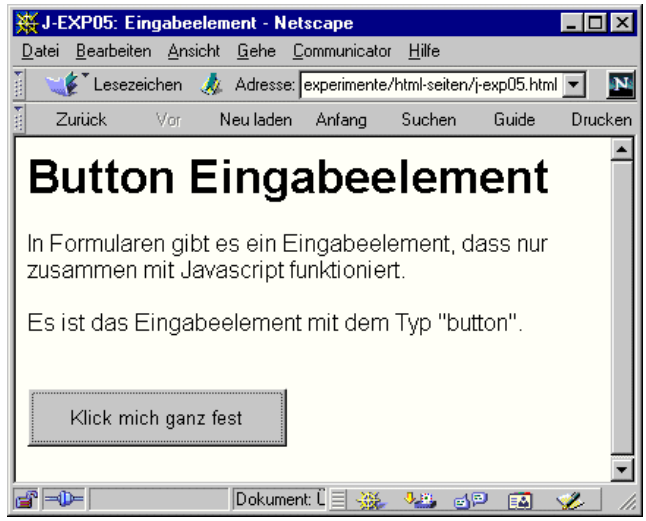

Abbildung 22.6: Arbeiten mit dem button-Eingabeelement

22.10 Bilder austauschen

Eine sehr häufig genutzte Möglichkeit von Javascript erlaubt es, Bilder auszutauschen, wenn die Seite bereits geladen ist. Oft werden damit Benutzerführungen realisiert. In dem Moment, wo der Benutzer mit der Maus über einen Menüpunkt fährt, wechselt der Browser das Bild, um dem Benutzer zu zeigen, was er gerade anklicken könnte.

Dazu brauchen wir einige Vorarbeiten. Zuerst müssen `<head>`-Anweisungen geschrieben werden, um alle Bilder zu laden, die dynamisch ausgewechselt werden sollen. Die Bilder sollten die jeweils zueinander passende Größe haben. Im Beispiel wurde für die vier beteiligten Bilder eine einheitliche Größe von 60x40 Pixel gewählt.

```
<script language="javascript" type="text/javascript">
function auswaehlen(index)
{
if (document.images)
        document.images["bild" + index].src = gewaehlt[index].src;
return true;
}
function normalisieren(index)
{
if (document.images)
        document.images["bild" + index].src = normal[index].src;
return true;
}
```

```
var gewaehlt = new Array();  // Feld anlegen
gewaehlt[0] = new Image();  // Bildbeschreibung anlegen
gewaehlt[0].src = "../../bilder/grün.gif";  // Bild laden
gewaehlt[1] = new Image();
gewaehlt[1].src = "../../bilder/blau.gif";

var normal = new Array();
normal[0] = new Image();
normal[0].src = "../../bilder/rot.gif";

normal[1] = new Image();
normal[1].src = "../../bilder/gelb.gif";
</script>
```

Der Javascript-Code definiert zwei Funktionen, die zum Ändern und Zurücksetzen der Bilder benötigt werden. Mit der Abfrage wird getestet, ob der Browser diese Handhabung der Bilder überhaupt unterstützt. Wenn ja, erfolgt der Austausch der Bilder. Der boole'sche Rückgabewert steuert die Auswertung der in der `<a>`-Anweisung angegebenen Verweise auf Dateien. In unserem Fall, der nur Bilder austauscht, sollen natürlich alle anderen Parameter ihre Gültigkeit bewahren. Daher geben wir *true* zurück.

Das Programmstück danach wird beim Laden der Seiten ausgeführt. Da es in der `<head>`-Anweisung steht, wird es vor dem Laden und Anzeigen der eigentlichen Seite ausgeführt.

Hier werden die notwendigen Variablen definiert. Da Javascript sich an *OOP*-Denkweisen (Objektorientierte Programmierung) anlehnt, werden neue Objekt-Variablen mit „new" angelegt. Einfache Variablen, wie *gewaehlt* legt man mit dem Schlüsselwort „var" an.

Mit *gewaehlt* erzeugen wir eine Referenz, legen mit *new* ein neues Feld an und speichern die Referenz darauf in *gewaehlt*. Die einzelnen Einträge des Feldes verweisen auf Verwaltungsobjekte für Bilder, die wir ebenfalls mit *new* anlegen.

Nun brauchen wir nur noch einer bestimmten Eigenschaft in den Bildverwaltungen, nämlich der Quellenangabe *src,* den Pfad der gewünschten Datei mitgeben. Der Javascript-Interpreter wird dies als Aufforderung verstehen, die Dateien in seinen internen Puffer zu laden.

Am Ende des Programmstückes sollten zwei Felder vorhanden sein, die jeweils zwei Bilder verwalten. Und auf die Felder kann man mit zwei Referenzvariablen zugreifen. Damit haben wir sozusagen das Material für aktive Schaltflächen bereitgestellt.

Jetzt müssen wir nur noch den notwendigen Reiz beschreiben und die richtige Reaktion auswählen.

Beispiel:

```
<a href="a-exp01.html"
onmouseover="auswaehlen(0);"
onmouseout="normalisieren(0);">
<img SRC="rot.gif" NAME="bild0" border="0" height="40" width="60">
</a>
```

In der HTML-Datei schreiben wir nun einen Verweis. In der `<a>`-Anweisung können wir Reaktionen angeben. Hier soll eine Reaktion erfolgen, wenn die Maus über das Bild (oder den Verweistext) fährt (*onmouseover*). Eine andere Reaktion erfolgt, wenn die Maus das Bild (oder den Verweistext) wieder verlässt (*onmouseout*). Bei einem Ereignis (Reiz) rufen wir eine passende Funktion auf und geben ihr einen Index mit. Die Namen der Bilder sind in der `<img>`-Anweisung so aufgebaut, dass sie mit dem Wort „bild“ beginnen und direkt die Indexnummer anfügen.

Mit dieser Namensgebung kann die Funktion dann aus dem Basiswort und dem Index den richtigen Bildnamen erzeugen und darauf zugreifen.

J-EXP06: Austausch von Bildern

Schreiben Sie eine HTML-Datei, die die Möglichkeiten von Javascript nutzt, um dynamisch Bilder währen der Anzeige der Seite auszutauschen.

Benutzen Sie dabei die besprochenen Code-Fragmente.

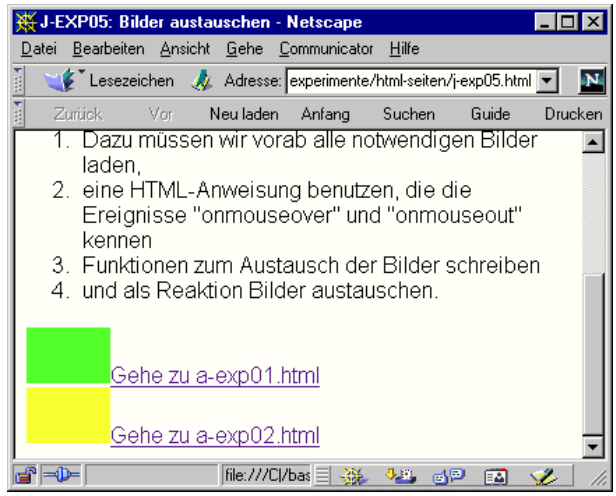

Abbildung 22.7: Bilderwechsel

22.11 Gleichzeitiges Laden von zwei Rahmen

Im Kapitel über Frames/Rahmen haben wir gesehen, dass die Benutzerführung nicht ganz einfach ist, da in manchen Situationen alte Informationen im Hauptfenster stehen bleiben, wenn wir auf der Hauptnavigationsebene das Thema wechseln. Hier wäre es schön, wenn wir bei einem Hauptthemenwechsel gleichzeitig die neue Detailnavigationsleiste einblenden könnten und auch eine Übersichtsseite zum neuen Thema.

Dieses Problem lösen wir noch schnell mit Javscript.

Allerdings müssen wir dabei den schmalen Pfad des kleinsten gemeinsamen Nenners zwischen den Browsern entdecken.

Besprechen wir dazu zuerst noch das Thema der Rückgaben einer Reaktion auf Reize. Wir haben bisher gesehen, dass eine Funktion den boole'schen Wert *true* zurückgeliefert hat. Im Falle eines *false*-Wertes sollte eigentlich keine weitere Auswertung der Parameter erfolgen. Aber grau ist alle Theorie. Nicht alle Browser halten sich daran.

Aus diesem Grund nutzen wir eine andere Möglichkeit. Wir können ja einen Pseudo-URL angeben, in dem das Schlüsselwort „javascript" gefolgt von einem Doppelpunkt einen Javabefehl anstelle des URLs angegeben wird. Damit umgehen wir das Problem, das Rückgabewerte von Reaktionen auf Reize nicht immer korrekt abgearbeitet werden.

Wie schaut aber eine Funktion in Javascript aus, die bestimmten Fenstern einen URL-Inhalt zuweist? Erinnern wir uns an den Vergleich mit dem Mobile, das als Muster für den internen Aufbau der Elemente einer Web-Seite diente. Wenn wir auf einem Querast sitzen und zu einem andern wechseln wollen, dann müssen wir sozusagen eine Ebene nach oben gehen, zum neuen Bereich wechseln, und dann wieder abwärts steigen.

Der Vergleich entspricht übrigens genau dem Navigieren in einem Dateibaum. „Nach oben" wird in einer WWW-kompatiblen Notation mit einem „.." beschrieben und bedeutet, „gehe zum übergeordneten Verzeichnis. Im DOM steht dafür das Wort „parent", das wir schon bei den Rahmen kennen gelernt haben.

Im übergeordneten Fenster, dem Gesamtrahmen, gibt es nun die einzelnen Rahmen, die einen eigenen Namen haben. Und darin gibt es wieder ein Dokument und auch eine Angabe, woher der Inhalt kommt. Diese Ortsangabe heißt *location*. Und darin existiert als Element (oder Eigenschaft) die Angaben des URLs in *href*. Ändern wir in dieser Eigenschaft den Wert, passiert das gleiche, das wir auch bei den Bildern gesehen haben. Der Browser lädt die entsprechende Datei nach.

Beispiel:

```
<script language="javascript" type="text/javascript">
function beides()
{
parent.main.location.href="j-exp07-leer.html";
parent.detailnavigation.location.href="j-exp07a-dnav.html";
return false;
}
</script>
```

Als Reiz nehmen wir nun besser nur den normalen Verweis und nicht die sonst übliche Beschreibung mit *onclick* etc. Schließlich kennen alle Browser den normalen Verweis.

Beispiel:

```
<a href="javascript: void beides();">Logische Auszeichnungen</a>
```

Ein letztes Problem bleibt noch zu lösen. Was machen wir mit dem Rückgabewert von „beides()“, wenn er existiert? Besser wird sein, wir ignorieren ihn in der momentan gewählten Anordnung. Der Befehl dazu heißt *void* (ungültig). Damit sollte es möglich sein, bei allen Standardbrowsern zwei Rahmen auf einmal zu füllen.

J-EXP07: Rahmenseite mit Mehrfachänderung von Detailrahmen

Ändern Sie das Rahmenbeispiel aus dem Rahmenkapitel so ab, dass bei einem Klick auf eine Sub-Menü-Auswahl, das Detailnavigationsfenster gesetzt, und gleichzeitig das Hauptfenster mit einer leeren Seite vorbesetzt wird. Benutzen Sie die angegebenen Code-Beispiele.

Abbildung 22.8: Situation vor dem Klick

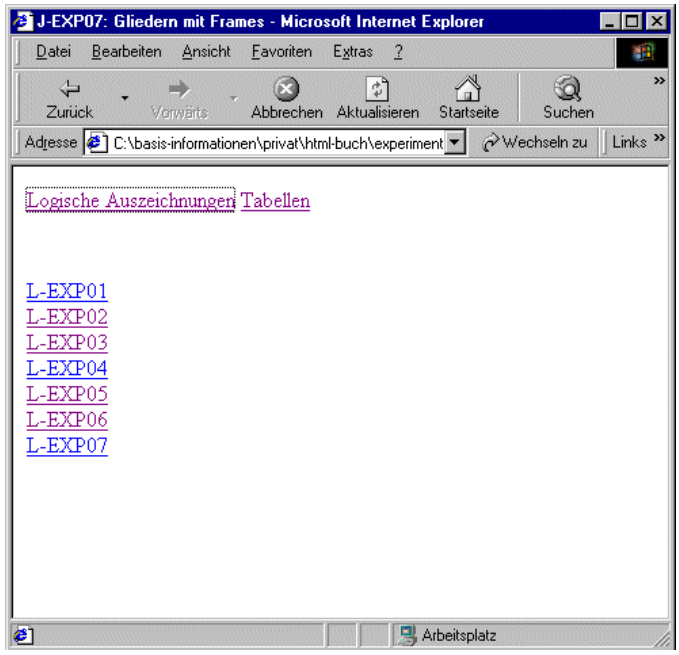

Abbildung 22.9: Situation nach dem Klick auf „Logische Auszeichnungen“

Mit diesen Beispielen sollten Sie einen Einblick in die Welt von Javascript und mögliche Anwendungen bekommen haben. Zum eigenen Programmieren sollten Sie trotzdem noch ein Buch Ihrer Wahl zu Rate ziehen.

22.12 Hinweise zu Javascript

Javascript-Funktionen und Programmteile müssen nicht immer innerhalb der HTML-Datei definiert werden. Sie können den *Javascript*-Code in einer externen Datei zusammenfassen und dann im Kopf die Datei mit Hilfe der folgenden Anweisung einbinden:

```
<script language="JavaScript" src="javascriptcode.js" type="text/javas-
cript">
</script>
```

Damit kann der gleiche Code in mehreren HTML-Dateien genutzt werden.

Im nächsten Kapitel

Mit den Programmiermöglichkeiten, die Javascript bietet, können wir einfache Änderung an HTML-Texten und Elementen einer Seite vornehmen. Eine zweite Aufgabe ist es, die Korrektheit von Eingaben zu überwachen. Man könnte noch weit mehr mit *Javascript* programmieren. Es gibt Beispiele für Taschenrechner, Steuerungen von Präsentationen und anderes.

Doch ausgefeilte Benutzerprogramme mit *Javascript* findet man im WWW eher selten.

Ein Grund ist sicher die Schwierigkeit, die die großen Browserhersteller allen Programmierern verursachen. Die Zugriffsmöglichkeiten auf die einzelnen Elemente einer Seite werden unterschiedlich realisiert. Nun müssten Programmierer mehrere Varianten einer Seite schreiben, um vielen Benutzern gerecht zu werden. Und dann bleiben immer noch einige Benutzer ausgesperrt, deren Browser nicht von den dominierenden Herstellern kommen.

Erst mit der nächsten Version des W3C Standards „DOM2“ (document object model / Dokumenten Aufbau Modell 2) wird es hoffentlich besser. Und mit der Version 6 des Netscape-Browsers ist auch bereits der erste Browser angekündigt, der sich vollständig an den W3C Standard halten wird und auch die Testseiten von W3C bisher als einziger korrekt anzeigt. Hier ist also ein wenig Hoffnung durchaus berechtigt.

22.13 Lösungen

J-EXP01: Reiz – Reaktion

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <title>J-EXP01: Javascript 1</title>
</head>
<body>
<h1>
Dynamisches HTML - 1</h1>
Erste Versuche mit Ereignissen und Reaktionen
<h1>
Ereignis "Maus dar&uuml;ber"</h1>
Ein Verweis soll empfindlich reagieren, wenn der Cursor &uuml;ber den akti-
ven
Verweistext f&auml;hrt.
<p><a href="j-exp02.html" onmouseover="javascript:alert('Hallo');">Demon-
stration</a>
</body>
</html>
```

J-EXP02: Aufruf einer Javascript-Funktion

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <title>J-EXP02: Javascript 2</title>
<script type="text/javascript">
function herbert(wert)
{
alert(wert);
}
</script>
</head>
<body>
<h1>Dynamisches HTML - 2</h1>
Erste Versuche mit Ereignissen und Reaktionen
<h1>Ereignisse und Bilder</h1>
Ein Bild soll empfindlich reagieren, wenn der Cursor über den Verweis gezo-
gen wird.
<br><a href="" onmouseover="herbert('mit einem Hallo');" >Javascript und
Funktionen</a>
```

```
</body>
</html>
```

J-EXP03: Browser-Informationen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="Author" content="Walter Herglotz">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <title>J-EXP03: Dynamischer Inhalt</title>
</head>
<body>
<h1>
Dynamischer Inhalt mit Javascript</h1>
 
<h2>
Anzeigen von Browser Informationen</h2>
Sie benutzen derzeit:
<br><script language="javascript" type="text/javascript">
document.write("- die Browser Version: <b>" + navigator.appVersion +"</
b><br>");
document.write("- den Browser-Hersteller: <b>" + navigator.appName + "</
b><br>");
document.write("- den Spitznamen: <b>" + navigator.appCodeName + "</
b><br>");
document.write("- die Plattform: <b>" + navigator.platform + "</b><br>");
</script>
<script language="javascript1.1" type="text/javascript">
document.write("Der Browser unterstützt die Version 1.1<br>");
</script>
<script language="javascript1.2" type="text/javascript">
document.write("Der Browser unterstützt die Version 1.2<br>");
</script>
</body>
</html>
```

J-EXP04: Testen von Werten aus Formularen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>J-EXP04: Feld-Check</title>
</head>
```

```
<body>
<h1>
Formular mit Feld-Check</h1>
Beispiel:
<br>&lt;input type="text" name="eingabe1" value="Vorschlag">
<br>Die Wertvorgabe wird im Eingabefensterangezeigt und benutzt, falls
sie vom Benutzer nicht gel&ouml;scht wird.
<h2>
Texteingaben</h2>
<form method="get" action="http://walter02/php3/echo_var.php3">
<table BORDER >
<tr>
<td>Nachname</td>
<td><input type="text" name="eingabe1" onblur="javascript:alert('OnBlur
nachname');"></td>
</tr>
<tr>
<td>Vorname</td>
<td><input type="text" name="eingabe2" ></td>
</tr>
<tr>
<td>Login</td>
<td><input type="text" name="eingabe3" ></td>
</tr>
<tr>
<td>Passwort</td>
<td><input type="password" name="eingabe4" ></td>
</tr>
</table>
<p>Die Daten k&ouml;nnen mit speziellen Eingabefeldern gel&ouml;scht
(zur&uuml;ckgesetzt)
und abgesendet werden.
<table BORDER=0 >
<tr>
<td><input type="reset" value="Eingaben zurücksetzen"></td>
<td><input type="submit" value="Zum Programm senden!"></td>
</tr>
</table>
</form>
</body>
</html>
```

J-EXP05: Benutzung des Eingabetyps button mit Javascript

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
```

```
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>J-EXP05: Eingabeelement</title>
</head>
<body>
<h1>
Button Eingabeelement</h1>
In Formularen gibt es ein Eingabeelement, dass nur zusammen mit Javascript
funktioniert.
<p>Es ist das Eingabeelement mit dem Typ "button".
<p><form methode="post" action="http:www.walter-digital.de/programme/
echo_var.php3">
<br><input type="button" onclick="javascript:alert('Knopf gedrückt');"
height="40" width="80" value="Klick mich ganz fest">
<br></form>
</body>
</html>
```

J-EXP06: Austausch von Bildern

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>J-EXP06: Bilder austauschen</title>
<script language="javascript" type="text/javascript">
function auswaehlen(index)
{
if (document.images)
document.images["bild" + index].src = gewaehlt[index].src;
return true;
}
function normalisieren(index)
{
if (document.images)
document.images["bild" + index].src = normal[index].src;
return true;
}
var gewaehlt = new Array();  // Feld anlegen
gewaehlt[0] = new Image();  // Bildbeschreibung anlegen
gewaehlt[0].src = "grün.gif";  // Bild laden
gewaehlt[1] = new Image();
gewaehlt[1].src = "blau.gif";
var normal = new Array();
```

```
normal[0] = new Image();
normal[0].src = "rot.gif";
normal[1] = new Image();
normal[1].src = "gelb.gif";
</script>
</head>
<body>
<h1>
Bilder autauschen</h1>
Sie kennen es sicher von Web-Seiten im Netz. Man f&auml;hrt mit der Maus
&uuml;ber ein graphisches Element, das einen bestimmte Auswahlm&ouml;glich-
keit
bietet, und kaum ereicht die Maus die Graphik, &auml;ndert sie sich, um
den Men&uuml;punkt deutlich hervorzuheben.
<ol>
<li>
Dazu m&uuml;ssen wir vorab alle notwendigen Bilder laden,</li>
<li>
eine HTML-Anweisung benutzen, die die Ereignisse "onmouseover" und
"onmouseout"
kennen</li>
<li>
Funktionen zum Austausch der Bilder schreiben</li>
<li>
und als Reaktion Bilder austauschen.</li>
</ol>
<a href="a-exp01.html" onmouseover="auswaehlen(0);" onmouseout="normali-
sieren(0);"><img SRC="rot.gif" NAME="bild0" BORDER=0 height=40 width=60></
a><a href="a-exp01.html" onmouseover="auswaehlen(0);" onmouseout="normali-
sieren(0);">Gehe
zu a-exp01.html</a>
<br><a href="a-exp02.html" onmouseover="auswaehlen(1);" onmouseout="norma-
lisieren(1);"><img SRC="gelb.gif" NAME="bild1" BORDER=0 height=40
width=60></a><a href="a-exp02.html" onmouseover="auswaehlen(1);" onmouse-
out="normalisieren(1);">Gehe
zu a-exp02.html</a>
</body>
</html>
```

J-EXP07: Rahmenseite mit Mehrfachänderung von Detailrahmen

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
```

```
   <title>J-EXP07: Gliedern mit Frames</title>
</head>
<frameset rows="65,*" >
<frame src="j-exp07-mnav.html" name="masternavigation" scrolling="no" fra-
meborder="0" noresize>
<frameset cols = "30%,70%" >
<frame src= "j-exp07-leer.html" name="detailnavigation" frameborder="0">
<frame src= "j-exp07-leer.html" name="main" frameborder="0">
</frameset>
<noframes>
<body>
Kann leider die Frames nicht anzeigen.<br>
Haben sie einen aktuellen Browser?<br>
</body>
</noframes>
</frameset>
</html>
```

J-EXP07-MNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP03 Master Navigation</title>
<script language="javascript" type="text/javascript">
function beides()
{
parent.main.location.href="j-exp07-leer.html";
parent.detailnavigation.location.href="j-exp07a-dnav.html";
return false;
}
</script>
</head>
<body>
<a href="javascript: void beides();">Logische Auszeichnungen</a>
<a href="j-exp07b-dnav.html"  target="detailnavigation">Tabellen</a>
</body>
</html>
```

J-EXP07A-DNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

```
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>R-EXP01header: Gliedern mit Frames</title>
</head>
<body>
<a href="l-exp01.html"  target="main">L-EXP01</a> <br>
<a href="l-exp02.html"  target="main">L-EXP02</a><br>
<a href="l-exp03.html"  target="main">L-EXP03</a><br>
<a href="l-exp04.html"  target="main">L-EXP04</a><br>
<a href="l-exp05.html"  target="main">L-EXP05</a><br>
<a href="l-exp06.html"  target="main">L-EXP06</a><br>
<a href="l-exp07.html"  target="main">L-EXP07</a><br>
</body>
</html>
```

J-EXP07B-DNAV:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>J-EXP073-dnav</title>
  <base target="main">
</head>
<body>
<h3>Tabellen</h3>
<a href="k-exp01.html" >K-EXP01</a> <br>
<a href="k-exp02.html">K-EXP02</a> <br>
<a href="k-exp03.html">K-EXP03</a> <br>
<a href="k-exp04.html">K-EXP04</a> <br>
<a href="k-exp05.html">K-EXP05</a> <br>
<a href="k-exp06.html">K-EXP06</a> <br>
<a href="k-exp07.html">K-EXP07</a> <br>
<a href="k-exp08.html">K-EXP08</a> <br>
<a href="k-exp09.html">K-EXP09</a> <br>
<a href="k-exp10.html">K-EXP10</a> <br>
<a href="k-exp11.html">K-EXP11</a> <br>
</body>
</html>
```

J-EXP07-LEER:

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

```
   <meta name="GENERATOR" content="Mozilla/4.51 [de] (Win98; I) [Netscape]">
   <meta name="Author" content="Walter Herglotz">
   <title>Fülldatei</title>
</head>
<body>
</body>
</html>
```

23 Java

23.1 Programmiersprachen für das Internet

Im Web werden innerhalb der Seiten zur Ausführung vor Ort, also bei Ihnen auf dem Rechner, zwei Sprachen vorrangig benutzt. Die erste ist Javascript. Javascript ist eine interpretierende Sprache, die somit in der Seite im Quelltext vorliegt und vom einem Interpreter im Browser ausgeführt wird. Verschiedene Anwendungen haben wir dazu bereits kennen gelernt.

Die zweite Sprache ist Java von Sun Microsystems. Sie ist eine Compilersprache. Der Javaquelltext wird übersetzt und das resultierende Programm kann dann in Dateien und Dateiarchiven auf dem Server zur Verfügung gestellt werden. Eine Webseite setzt sich typischerweise aus verschiedenen Dateiinhalten zusammen. Die HTML-Datei beschreibt dabei das Zusammenspiel zwischen Texten, Bildern und anderen Dateien, Eine der Dateiarten, die eingebunden werden können, sind die Java-Programmdateien.

23.2 Java-Besonderheit: Virtuelle Maschinen

Die Java-Programme werden in einen speziellen Code übersetzt, der für eine virtuelle CPU gedacht ist. Diesen Pseudo-Code nennt man Byte-Code. Da es diese CPU nicht gibt, schreibt man für jede Plattform ein Programm, das dort diese virtuelle CPU nachbildet. Damit können Programme einmal geschrieben und ohne weitere Veränderungen auf den verschiedensten Umgebungen ausgeführt werden.

Der entscheidende Unterschied zu anderen Sprachen und Programmiersystemen ist die Möglichkeit, fertig übersetzten Code auf völlig unterschiedlichen Maschinen ablaufen zu lassen. Ein Programm, das ein Webserver zur Verfügung stellt, kann beim Benutzer ohne Änderung auf Linux wie auf Mac oder Windows (oder Großrechnern) ablaufen.

Dies machen jeweils *Virtuelle Maschinen* möglich, die auf den Zielrechnern zur Verfügung stehen und eine spezielle CPU, eine Ablaufumgebung und auch ein graphisches System für die graphische Benutzerschnittstelle von Java-Programmen zur Verfügung stellen. Ein anderer Begriff dafür ist auch *JRE* (java runtime environment / Java Ablaufumgebung).

23.3 Laden von Java-Programmen

Verwendet man in einer HTML-Seite nun einen passenden Verweis auf ein Java-Programm, wird es über das Netz geladen und innerhalb eines sogenannten Sandkastens des Browsers ausgeführt. Der Sandkasten blockiert Zugriffe auf Festplatten u.ä. Fremder Code kann nur mit dem Browser auf dem Rechner des Nutzers und dem Server, von dem das Programm kommt, agieren.

Das Wunderbare an diesem Verfahren ist die einfache Möglichkeit der Programmpflege. Denn es genügt bei einer neuen Version die einzige Programmdatei auf dem Server auszutauschen, um alle Anwender unmittelbar in den Genuß der neuen Version zu bringen.

Es gibt noch weitere Sprachen, die aber nur innerhalb einer Herstellerwelt eine lokale Bedeutung haben und vor allem nicht in offiziellen Gremien standardisiert wurden.

23.4 Einbinden in HTML-Seiten

Um ein Java-Programm in einer HTML-Seite einbinden zu können, muss das Programm als so genanntes *Applet* („kleine Anwendung“) geschrieben worden sein. Das Applet nutzt die Umgebung des Browsers. Das Gegenstück zu einem Applet, das nur innerhalb eines Browsers ablaufen kann, ist eine Java-Applikation, die eigenständig lauffähig ist.

Eine spezielle HTML-Anweisung wurde traditionell zum Einbinden des Java-Applets benutzt. Es ist die `<applet>`-Anweisung. Der HTML-Standard empfiehlt inzwischen die allgemeinere `<object>`-Anweisung. Leider funktionieren zur Zeit noch nicht einmal die einfachsten Beispiele, die im Standard erwähnt werden, in den gängigsten Browsern. Bleiben wir daher bei der `<applet>`-Anweisung.

Beispiel:

```
<applet codebase="classes" code="JavaClock.class" width="150"
height="150">
<param name="delay" value="100">
...
</applet>
```

Im Beispiel, das Sie unter *http://www.walter-herglotz.de/programme/ javauhr.html* finden, reservieren die Breiten- und Höhenangaben in der `<applet>`-Anweisung ein Rechteck auf der Browserseite. Dies ist nun der Ausgabebereich des Applets. Ein Applet hat kein eigenes Fenster, sondern arbeitet nur innerhalb des Browserfensters.

Mit *codebase* beschreibt man das Verzeichnis entweder relativ wie hier oder auch mit einem absoluten URL, um auf die Originalquelle zu verweisen. Für produktive Anwendungen empfiehlt sich die zweite Variante, da man so sicherstellen kann, dass alle Seiten, die ein Java-Applet benutzen, in den Genuss einer Aktualisierung des Programmes kommen.

Zwischen der Start- und der Ende-Marke der `<applet>`-Anweisung können Parameterwerte mit Namen und Wert eingefügt werden. Diese Angaben werden vom Programmierer des Programms vorgesehen und erlaube Ihnen, das Applet in Ihrer Seite anzupassen. Meist werden fehlende Angaben mit einem Standardwert ersetzt. Zusätzliche Parameter, die ein HTML-Autor erfindet, haben natürlich keinen Einfluss auf das Programm.

W-EXP01: Java-Applets in Web-Seiten einbinden

Surfen Sie ein wenig im Netz. Besuchen Sie *www.javasoft.com* und suchen Sie nach freien Applets, die man in eigene Webseiten einbauen darf.

Auf der Buch-Web-Seite steht auch eine Download-Möglichkeit einiger Beispiele von SUN zur Verfügung. (in der genannten Web-Seite unter *www.walter-herglotz.de/programme/javauhr.html*).

Erstellen Sie eine Webseite, die ein Applet einbindet.

Abbildung 23.1: Java-Uhr in einer Web-Seite

23.5 Anwendungsbeispiel

Applets haben ein gutes Stück dazu beigetragen, die Programmiersprache Java und Anwendungen damit sehr populär zu machen. Heute werden eher eigenständige Java-Programme genutzt, die über das Netz lokal installiert aber bei einem Aufruf des Programms zentral gepflegt werden. Das spart die Übertragung des Programms bei jedem Start.

Aber ein Beispiel aus der Welt der Applets möchte ich noch erwähnen. In einer Ecke meines Kellers steht eine Linux-Maschine, die einen Webserver und viel Speicherplatz für meinen PC bereitstellt. Es ist nur ein Linux-PC ohne Monitor. Um aber bei Bedarf auch diese Maschine bedienen zu können, steht auf dem Server eine Webseite mit einem Telnet-Applet. Telnet ist eine Fernwartungsprogramm, das ein Terminal (Eingabeaufforderung) über das Netz zur Verfügung stellt.

Nun kann ich von meinem Windows-PC über das Netz die entsprechende HTML-Seite vom Linux-PC laden, das eingebaute Telnet-Applet bedienen und damit den Linux-PC fernsteuern. Den Monitor habe ich nur zur Erstinstallation benötigt.

23.6 Kombination von Java und Javascript

Java und Javascript können auch zusammen arbeiten. Funktionen eines Java-Applets können mit Hilfe von Javascript aufgerufen werden und umgekehrt.

Das erlaubt eine sehr flexible und dynamische Gestaltung von Webseiten.

W-EXP02: Javascript ruft Java

Laden Sie die Webseite W-EXP02.HTML und fahren Sie mit der Maus über die Verweise. Die Änderungen sollten Sie im Java-Applet sehen, das eine Ampel nachbildet.

Betrachten Sie auch den Quellcode, um die Javascript-Anweisungen zu sehen, die die Java-Funktion *setlampe()* aufrufen.

Damit dies einfach möglich ist, hat in diesem Beispiel das Java-Applet einen HTML-Namen erhalten (*A1*).

Abbildung 23.2: Kombination aus Java und Javscript

Im nächsten Kapitel

Leider gibt es nun kein nächstes Kapitel mehr. Wir sind am Ende unserer Reise durch die Möglichkeiten der Webseiten-Gestaltung angekommen.

Hoffentlich haben Sie erleben können, welche Chancen das WWW jedem zur Publikation bietet. Kein Wunder, dass insbesondere Marketing- und Werbestrategen das Web für Ihre Zwecke entdeckt haben. Aber es kann auch an tausend anderen Stellen kleine Wunder bewirken.

Man kann sehr einfach beginnen, die eigenen Seiten mit der Zeit weiter zu entwickeln und schließlich sogar Programme einzusetzen.

Publizieren Sie ihr ernsthaftes Hobby, stellen eigene Erfahrungen anderen zur Verfügung und gewinnen Sie durch die Diskussion mit Gleichgesinnten.

Oder sammeln Sie einfach Unterlagen, die Sie sowieso erstellen, nun im Webformat. Lehrer, Professoren, Universitäten oder Volkshochschulen könnten ihren *Kunden* weit mehr Nutzen bieten, wenn Sie sich zusammen mit ihren Schülern, Studenten oder Teilnehmern als Arbeitsgruppen begreifen würden, deren Arbeitsergebnisse aufeinander aufbauen. Aber das ist natürlich eine Frage der Kultur des Einzelnen und der Organisationen.

Erlauben Sie mir einen abschließenden Tipp: suchen Sie sich eine Anwendung, nutzen Sie das Gelernte für Ihre eigenen Zwecke. Gehen Sie ein eigenes Projekt an.

Viel Erfolg und noch mehr Spaß dabei wünscht Ihnen

Walter Herglotz
wjh@walter-herglotz.de

»Streng dich an.
Versuche soviel Ausbildung wie möglich zu bekommen,
und dann,
um Himmels Willen,
tu etwas.«
- Lee Iacocca

23.7 Lösungen

W-EXP01: Java-Applets in Web-Seiten einbinden

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.73 [de]C-CCK-MCD DT  (WinNT; U)
   [Netscape]">
   <title>Analoge Uhr mit Java (von SUN)</title>
</head>
<body>
<h1>Java - Uhr</h1>
<div align="center">
<applet codebase="classes" code="JavaClock.class" width="150"
height="150">
<param name="delay"   value="100">
<param name="link"    value="http://java.sun.com/">
<param name="border"  value="5">
<param name="nradius" value="80">
<param name="cfont"   value="TimesRoman|BOLD|18">
<param name="bgcolor" value="ffffff">
<param name="shcolor" value="ff0000">
<param name="mhcolor" value="00ff00">
<param name="hhcolor" value="0000ff">
<param name="ccolor"  value="dddddd">
<param name="ncolor"  value="000000">
</applet>
</div>
<p>Beispiel von: www.javasoft.com</p>
<p><a href="demo.zip">Download des lokalen Archivs</a>
</p>
</body>
</html>
```

W-EXP02: Javascript ruft Java

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
   <meta name="GENERATOR" content="Mozilla/4.6 [de] (WinNT; I) [Netscape]">
   <title>Ampel mit Java und Javascript</title>
</head>
<body>
 
<center>
```

```
<h1>
Ampel mit Applet und Javascript</h1></center>
<table BORDER COLS=2 WIDTH="100%" >
<tr>
<td><applet
code = ampel.class
width = 50
height = 150
name = A1></applet></td>
<td>
<br><a href="ampel.java" onMouseOver="javascript:document.A1.set-
lampe(1)">Rot</a>
<p><a href="ampel.java" onMouseOver="javascript:document.A1.set-
lampe(2)">Gelb</a>
<p><a href="ampel.java" onMouseOver="javascript:document.A1.set-
lampe(3)">Gr&uuml;n</a>
<p><a href="ampel.java" onMouseOver="javascript:document.A1.set-
lampe(0)">Reset</a>
<br> </td>
</tr>
</table>
<a href="ampel.java">Quellext (Java) </a>
<hr>
<br> 
</body>
</html>
```

A Anhang

A.1 Darstellung von Farben mit einer hexadezimalen Dreier-Gruppe

Zur Farbdarstellung verwendet HTML eine RGB-Angabe. Jede Farbe wird dabei aus Werten für Rot, Grün und Blau angegeben. Eine einzelne Farbe kann dabei Werte von 0 bis 255 annehmen. Stellt man diese Werte binär dar, passen Sie genau in ein Byte. Insgesamt werden also 24 Bits (3 Bytes) zur Farbangabe benutzt. Damit lassen sich 16,7 Millionen Farbwerte darstellen. Die heute üblichen Graphikkarten können diese Farbwerte anzeigen.

Man kann beliebige, dezimale Zahlen in die Hexadezimal-Darstellung durch eine Division mit 16 umrechnen. Die Werte für 10 bis 15 werden dabei durch die Buchstaben ‚A' bis ‚F' ersetzt.

Beispiel:

Setzen wir Rot auf 200 und die beiden anderen Farben auf 0.

200 : 16 = 12D oder CH (12 in dezimaler und ‚C' in hexadezimaler Darstellung)

Rest 8

Also erhalten wir: C8H

Und als Farbwert ergibt sich: „#c80000"

In vielen Betriebssystemen kann man einen Farbrechner starten, der die Werte der verschiedenen Farben in RGB angibt. Die Umrechnung macht dann ein entsprechender Taschenrechner. Es ist auch nicht schlimm, wenn man die Umrechnung durch Division macht.

A.2 Tabelle der Farbnamen

Farb-Namen	Rot	Grün	Blau
aliceblue	F0	F8	FF
antiquewhite	FA	EB	D7
aqua	00	FF	FF
aquamarine	7F	FF	D4
azure	F0	FF	FF
beige	F5	F5	DC
bisque	FF	E4	C4
black	00	00	00
blanchedalmond	FF	EB	CD
blue	00	00	FF
blueviolet	8A	2B	E2
brown	A5	2A	2A
burlywood	DE	B8	87
cadetblue	5F	9E	A0
chartreuse	7F	FF	00
chocolate	D2	69	1E
coral	FF	7F	50
cornflowerblue	64	95	ED
cornsilk	FF	F8	DC
crimson	DC	14	3C
cyan	00	FF	FF
darkblue	00	00	8B
darkcyan	00	8B	8B
darkgoldenrod	B8	86	0B
darkgray	A9	A9	A9
darkgreen	00	64	00
darkkhaki	BD	B7	6B
darkmagenta	8B	00	8B
darkolivegreen	55	6B	2F
darkorange	FF	8C	00
darkorchid	99	32	CC
darkred	8B	00	00
darksalmon	E9	96	7A
darkseagreen	8F	BC	8F
darkslateblue	48	3D	8B
darkslategray	2F	4F	4F

Farb-Namen	Rot	Grün	Blau
darkturquoise	00	CE	D1
darkviolet	94	00	D3
deeppink	FF	14	93
deepskyblue	00	BF	FF
dimgray	69	69	69
dodgerblue	1E	90	FF
firebrick	B2	22	22
floralwhite	FF	FA	F0
forestgreen	22	8B	22
fuchsia	FF	00	FF
gainsboro	DC	DC	DC
ghostwhite	F8	F8	FF
gold	FF	D7	00
goldenrod	DA	A5	20
gray	80	80	80
green	00	80	00
greenyellow	AD	FF	2F
honeydew	F0	FF	F0
hotpink	FF	69	B4
indianred	CD	5C	5C
indigo	4B	00	82
ivory	FF	FF	F0
khaki	F0	E6	8C
lavender	E6	E6	FA
lavenderblush	FF	F0	F5
lawngreen	7C	FC	00
lemonchiffon	FF	FA	CD
lightblue	AD	D8	E6
lightcoral	F0	80	80
lightcyan	E0	FF	FF
lightgoldenrodyellow	FA	FA	D2
lightgreen	90	EE	90
lightgrey	D3	D3	D3
lightpink	FF	B6	C1
lightsalmon	FF	A0	7A
lightseagreen	20	B2	AA
lightskyblue	87	CE	FA
lightslategray	77	88	99

Farb-Namen	Rot	Grün	Blau
lightsteelblue	B0	C4	DE
lightyellow	FF	FF	E0
lime	00	FF	00
limegreen	32	CD	32
linen	FA	F0	E6
magenta	FF	00	FF
maroon	80	00	00
mediumaquamarine	66	CD	AA
mediumblue	00	00	CD
mediumorchid	BA	55	D3
mediumpurple	93	70	DB
mediumseagreen	3C	B3	71
mediumslateblue	7B	68	EE
mediumspringgreen	00	FA	9A
mediumturquoise	48	D1	CC
mediumvioletred	C7	15	85
midnightblue	19	19	70
mintcream	F5	FF	FA
mistyrose	FF	E4	E1
moccasin	FF	E4	B5
navajowhite	FF	DE	AD
navy	00	00	80
oldlace	FD	F5	E6
olive	80	80	00
olivedrab	6B	8E	23
orange	FF	A5	00
orangered	FF	45	00
orchid	DA	70	D6
palegoldenrod	EE	E8	AA
palegreen	98	FB	98
paleturquoise	AF	EE	EE
palevioletred	DB	70	93
papayawhip	FF	EF	D5
peachpuff	FF	DA	B9
peru	CD	85	3F
pink	FF	C0	CB
plum	DD	A0	DD
powderblue	B0	E0	E6

Farb-Namen	Rot	Grün	Blau
purple	80	00	80
red	FF	00	00
rosybrown	BC	8F	8F
royalblue	41	69	E1
saddlebrown	8B	45	13
salmon	FA	80	72
sandybrown	F4	A4	60
seagreen	2E	8B	57
seashell	FF	F5	EE
sienna	A0	52	2D
silver	C0	C0	C0
skyblue	87	CE	EB
slateblue	6A	5A	CD
slategray	70	80	90
snow	FF	FA	FA
springgreen	00	FF	7F
steelblue	46	82	B4
tan	D2	B4	8C
teal	00	80	80
thistle	D8	BF	D8
tomato	FF	63	47
turquoise	40	E0	D0
violet	EE	82	EE
wheat	F5	DE	B3
white	FF	FF	FF
whitesmoke	F5	F5	F5
yellow	FF	FF	00
yellowgreen	9A	CD	32

Quelle: *http://home.netscape.com/misc/contact_info.html*

Pearson Education

ADDISON-WESLEY

G Glossar

ASCII

American standard code of information interchange, ein Zeichen verwendet nur 7 Bits, wurde in ISO-8859-1 und UniCode integriert.

Block

In HTML ein Auszeichnungselement, das auf einer eigenen Zeile beginnt (z.B. eine Überschrift).

Browser

Anzeige- und Navigationsprogramm auf Ihrem Rechner (to browse - durchsuchen). Versteht HTML und das Übertragungsprotokoll HTTP.

CGI

Common gateway interface – übliche Verbindungsschnittstelle zwischen Web-Server und Anwendungsprogrammen, die aus dem Web heraus aufgerufen werden können.

Client

Bedeutet Kunde oder Nutzer, meist der Browser im WWW.

Container

In HTML eine Auszeichnungselement, mit einem Inhalt. Container existieren als Blockcontainer (z.B. <body>, Tabellenzellen) oder als in-line Container (z.B. , aber auch).

Domain

Verwaltungsbezirk im Internet, verwaltet meist eigene Domänen und Rechnernamen, wir unterteilt in die erste Stufe (first level) mit Länderkennungen (de, fr, it, cc..) oder Funktionen in den USA (com, gov, edu..).

Host/Server

Rechner, der Dienste im Internet anbietet, z.B. Web-Server, der WWW-Dateien senden kann.

HTML

Hypertext Markup Language – Seitenbeschreibungssprache.

HTTP

Hypertext Transfer Protokoll, Übertragungsprotokoll zwischen Browser und Web-Server.

HTTPD

Kurzbezeichnung für einen Web-Server; steht für HTTP-Dämon. Dämonen sind unter Unix/Linux die Server-Programme im Hintergrund.

Hypertext

Lesbarer Text, der mit anklickbaren Sprungelementen durchsetzt ist. Beispiele sind Web-Seiten oder Hilfedateien.

Internet

Verbindungseinrichtungen zwischen Netzen (Inter-Net), die zusammen mit den Teilnetzen ein weltumspannendes Kommunikationssystem bilden, basiert auf dem IP-Protokoll.

ISO-8859-1

International genormter Zeichensatz für Mitteleuropa, enthält den ASCII-Code, auch Latin-1 oder Western genannt.

Java

Plattform unabhängige Programmiersprache, verbreitet sich stark in Verbindung mit Web-Anwendungen und Kommunikationsanwendungen, wird übersetzt.

Javascript

Standardisierte Script-Sprache für Web-Seiten, wird im Browser ausgeführt.

Link

Verweis innerhalb einer Web-Seite auf eine Stelle in einer Web-Seite oder auf eine andere Datei oder Dienstleistung.

MarkUp

Auszeichnen. Funktioniert wie ein "Marker" aus dem Schreibwarenladen. Damit kann man bestimmte Stellen zu einem Thema logisch markieren (z.B. grün = merken, gelb = ausschneiden, rot = an einen Freund schicken) . In HTML dienen Anweisungen (Steuerelemente) dazu, verschiedene logische Markierungen anzubringen (z.B. mittlere Überschrift, Text hervorheben,...).

Netzwerk

Allgemeiner Name für Rechnerverbindungen, kann grob in verbindende (routed) und lokale (non routed) Netzwerke unterteilt werden. Für das Internet benötigen wir verbindende Netzwerke.

Protokoll

Ein Protokoll regelt den Datenaustausch zwischen den Komponenten des Netzes. Beispiele: IP-das Basisprotokoll, NNTP-das Zeitprotokoll.

Provider

Anbieter, der Anrufern meist über Telefon einen Zugang zum Internet ermöglicht.

Publizieren

Schreiben von Web-Seiten und deren Veröffentlichung auf einem Web-Server.

Rechnernamen

Rechner im Netz benötigen einen weltweit eindeutigen Rechnernamen. Technisch verbergen sich hinter Rechnernamen IP-Adressen.

Server

Dienstgeber, kann als Software ein Protokoll bedienen (z.B. WWW-Server) oder als Kombination von Rechner und Software gemeint sein.

Surfen

Umgangssprachlicher Ausdruck für das Klicken auf Verweise, um von Seite zu Seite zu gelangen.

TCP/IP

Name der zwei wichtigsten Übertragungsprotokolle im Internet: IP -Internet Protocol (verbindungslos) und TCP - Transmission Control Protocol (verbindungsorientiert). Das Besondere an IP ist, das jedes Paket Sender- und Empfänger-Adressen mit sich führt. Jede Adresse besteht sowohl aus einer Netz- wie einer Rechnerkennung.

UniCode

Weltstandard für alle Zeichen, enthält ASCII und ISO-8859-1 Zeichen.

URL

Uniform resource locator – einheitliche Adressierung von Ressourcen im WWW.

Verweis

Siehe Link

Webseite

WWW-Anbieter, von der kleinen Homepage bis hin zu einem globalen Netz von Rechner-Systemen, die unter einem einzigen Namen angesprochen werden können.

WWW-World Wide Web

Name für das Netz, das man mit Verweisen (Links) in den Dokumenten aufbaut, basiert auf dem HTTP-Protokoll des Internets, entstanden am CERN/ Genf/ Schweiz.

WWW.W3C.ORG

Standardisierungs-Organisation für das WWW, getragen von INRIA (Frankreich), MIT (USA) und Keio Universität (Japan).

S Stichwortverzeichnis

A

B

C

D

E

R

S

T

U

V

W

Z