Vorwort

Das Internet ist eine Technologie, die nicht alles, aber doch vieles einfacher macht. Es gibt zwar auch einige Unannehmlichkeiten wie zum Beispiel die Virenverteilung per E-Mail, die guten Seiten des Internets überwiegen jedoch: Man ist mit dem Laptop quasi immer und überall erreichbar, auch unterwegs. E-Mails sparen Portogebühren und kommen schneller ans Ziel. Im WWW sind nahezu alle Informationen kostenlos abrufbar. Kurz, das Internet bringt Kommunikation satt.

Die Sprache, die das WWW für die Benutzer und die Entwickler so einfach machte, ist HTML. HTML steht für **H**yper**T**ext **M**arkup **L**anguage, was soviel heißt wie Textauszeichnungssprache. Man kann HTML nicht als Programmiersprache bezeichnen, da keine Variablen benutzt oder Schleifen eingesetzt werden können. Für die HTML-Technologie ist übrigens ein eigenes Kapitel im Buch vorgesehen, denn ohne die grundlegenden Regeln der Sprache zu kennen, macht es keinen Sinn, sich mit Cascading Stylesheets auseinanderzusetzen.

HTML hat jedoch zwei Schwächen: Zum einen bietet es nur begrenzte Möglichkeiten, Texte oder Daten auszuzeichnen. Deswegen ist die Sprache XML dabei, HTML abzulösen. Das Kürzel steht hierbei für E**x**tensible **M**arkup **L**anguage, also Erweiterbare Auszeichnungssprache. XML kann nicht nur für Text, sondern allgemein für die Beschreibung strukturierter Daten eingesetzt und – wie der Name schon sagt – erweitert werden. Für XML ist ein kleinerer Abschnitt im Kapitel »HTML-Schnellkurs« integriert, da tiefer gehende XML-Kenntnisse hier nicht erforderlich sind. Dieses Buch soll schwerpunktmäßig den Einsatz von CSS mit HTML behandeln.

Der zweite Schwachpunkt von HTML betrifft das Aussehen der Site – und hier kommen Cascading Stylesheets ins Spiel: Wenn Sie auf einer Website grafische Elemente und Textformatierungen einsetzen wollen, stoßen Sie mit HTML schnell an die Grenzen – denn dafür wurde diese Sprache nicht entwickelt. CSS dagegen ist eine im Web häufig eingesetzte Sprache speziell zur Formatierung von verschiedenen HTML-Elementen. Zur Darstellung von XML ist CSS auch geeignet. Es ist zwar nur eine von mehreren Sprachen, wohl aber die bedeutendste, vor allem für die Aufbereitung von XML-Daten als Webseiten. CSS wurde, genauso wie andere Standards des WWW, zum Beispiel auch HTML und XML, vom W3C entwickelt.

Soviel vorweg zu Cascading Stylesheets und warum eine Beschäftigung mit dieser Technologie jetzt schon, aber mehr noch für die Zukunft wichtig ist. Im Laufe der Lektüre dieses Buches werden Sie alles Wichtige über diese mächtige Formatierungssprache erfahren.

Mit Danksagungen will ich Sie zwar nicht nerven, aber da gibt es eine Person, die mir wirklich entscheidend ausgeholfen hat. Ich danke Robert Jost dafür, dass er mir sein Notebook drei Monate zur Verfügung stellte, als mein Rechner funktionsuntüchtig war. Danke auch an Florian Kunkel, der ebenso wie Jochen Rill, Timo Elbert und Jan Stanzel den Lerneffekt des vorliegenden Buches ausgetestet hat.

Noch ein Wort zum Buch: Ich empfehle Ihnen, die Einleitung sorgfältig zu lesen, denn sie enthält Informationen, die für die weitere Arbeit sehr wichtig sind. Auch verrät sie Ihnen, was benötigt wird, um die Beispiele sofort nachvollziehen zu können. Zugleich erfahren Sie, wie das Buch aufgebaut ist.

Und last but not least: Ich habe mich bemüht, dieses Buch leicht verständlich zu schreiben. Sollten Sie trotzdem mal etwas nicht verstehen, so schicken Sie mir eine E-Mail an *daniel.jokat@mut.de*. Ich freue mich natürlich auch über Anregungen sowie jegliche Art von Kritik. Bei einer 2. Auflage werde ich versuchen, alles weitestgehend zu berücksichtigen.

Das war's fürs Erste. Ich wünsche Ihnen viele interessante Lesestunden und rasche Fortschritte.

Daniel Jokat

Einleitung

Sicher sind Sie schon wild darauf, endlich mit CSS zu beginnen. Wunderbar, denn das ist die erste Voraussetzung für einen raschen Erfolg. Mit diesem Buch werden Sie als Webprogrammierer bald CSS einsetzen können, um Ihre Site zu optimieren oder die Erstellung zu beschleunigen. Doch sollten Sie sich noch etwas Zeit für die Einleitung nehmen, um sich einen Überblick über den Stoff zu verschaffen und den roten Faden im Buch zu entdecken.

Wie das Buch aufgebaut ist

Der Schwerpunkt eines Buches liegt – natürlich – immer auf dem Hauptthema – in diesem Falle also CSS. Doch wie manche Themen ist auch dieses komplexer (sonst würde man ja auch keine Bücher darüber schreiben) und erfordert ein paar Vorkenntnisse.

Das vorliegende Buch beginnt mit allgemeinen Anmerkungen zur Sprache CSS: Wer entwickelte sie, was kann sie, wofür setzt man sie ein und so weiter.

Der Button auf dem Cover verspricht einen Start ohne Vorwissen. Da CSS ohne Grundkenntnisse im Umgang mit HTML nicht sinnvoll angewendet werden kann, folgt in Kapitel 2 ein Grundkurs HTML. Sie können fragen: »Warum ein ganzes Kapitel über HTML und nur ein kurzer Abschnitt über XML?« Das Buch befasst sich größtenteils mit dem Einsatz von CSS mit HTML, dem bisher hauptsächlichen Einsatzgebiet von CSS. CSS im Zusammenhang mit XML verwenden wir in diesem Buch nur, um aus einer XML-Datei eine Webseite zu machen. Dafür benötigen Sie keine umfangreichen XML-Kenntnisse.[1]

Des Weiteren lernen Sie, wie man in CSS »sauber« programmiert, das heißt, wie man eine gute CSS-Codestruktur entwickelt. Dabei erfahren Sie unter anderem auch, wie CSS-Dateien aufgebaut werden.

Im Anschluss werden Sie Ihr erstes Stylesheet mit CSS schreiben, um einen einfachen Text zu formatieren. Eine zeilenweise Besprechung des Codes (auch des HTML-Codes) hilft, das Beispiel zu verstehen.

Nachdem Sie ein – mehr oder weniger – praxisnahes Beispiel nachvollzogen haben, werden weitere Stileigenschaften und Werte (so genannte *values)* behandelt. Sie lernen, mit CSS zu gestalten und zu formatieren und die sehr nützlichen Klassen in CSS zu definieren und in HTML einzusetzen.

»Regeln für die Typografie« heißt es im weiteren Verlauf. Es gibt für das Internet verschiedene grundsätzliche Gestaltungsregeln, von denen ich einige darstellen möchte.

In diesem Buch gilt der Grundsatz »Do it on your own« - sprich, wir schreiben den ganzen Code selbst. Doch im späteren (Berufs)Leben werden Sie zweifelsohne mit CSS-Editoren arbeiten, die viel Zeit sparen können (was ist zum Beispiel wenn Ihnen die erforderliche CSS-Stileigenschaft nicht mehr auf Anhieb einfällt?). Zwei Editoren, die Sie auch auf der Buch-CD finden, stelle ich Ihnen vor.

XML ist dabei, HTML als Web-Standard abzulösen (bis es aber so weit ist, vergeht wohl noch ein Weilchen). Mit CSS sind Sie in der Lage, eine XML-Datei zur Webseite zu machen. Wie, erfahren Sie auch in diesem Buch.

Wenn Ihnen das im Buch gebotene Themenspektrum zu CSS noch nicht gereicht hat, können Sie sich im Internet ausführlicher informieren. Vier Referenzen werden vorgestellt und einige gute und schlechte Beispiele für Webseiten, die CSS verwenden, analysiert.

Was Sie lesen sollten

Betrachten Sie das Buch als eine Art Anleitung zum Selbststudium. Sie bringen sich durch das Lesen des Buches, das Ausprobieren der Beispiele und durch die Übungsaufgaben die Sprache CSS selbst bei. Ein Kapitel des Buchs entspräche somit einer Lektion eines CSS-Kurses. Wollen Sie wirklich das bestmögliche Resultat mit dem Buch erzielen, so sollten Sie das Buch von vorne nach hinten durcharbeiten, da die einzelnen Kapitel aufeinander aufbauen. Für diejenigen Leser, die schon Vorkenntnisse mitbringen oder gezielt nach bestimmten Informationen suchen, hier eine Kurzübersicht:

1. Wenn Sie sich näher mit XML befassen wollen, können Sie dies anhand des Buches »Jetzt lerne ich XML« von Günter Born, erschienen im Markt+Technik Verlag tun.

Soweit Sie nicht wissen, für was CSS gut ist, sollten Sie Kapitel 1 lesen. Hier erfahren Sie, wofür man CSS einsetzt, welche Browser CSS unterstützen, für welchen Zweck man CSS entwickelte, etc..

Wenn Sie keine HTML-Grundkenntnisse besitzen, sollten Sie unbedingt Kapitel 2 lesen. Es ist für das Verständnis des restlichen Buches unentbehrlich.

In Kapitel 3 werden Sie den Grundaufbau einer CSS-Datei kennen lernen und erfahren, was man unter einer guten Struktur versteht und wie man gut strukturierten Code aufsetzt. Außerdem werden Sie Ihr erstes Stylesheet schreiben. Ich empfehle, das Kapitel zu lesen, da hier der Code zeilenweise behandelt wird und Sie erste, detaillierte Einblicke in die CSS-Programmierung bekommen.

Kapitel 4 befasst sich dann intensiv mit CSS, mit den einzelnen Formatierungsmöglichkeiten, Variationen und so weiter. Dieses Kapitel ist ein "Muss", wenn Sie CSS erlernen wollen.

Mit Bildern lässt sich in CSS einiges anstellen. Kapitel 5 widmet sich den Grafiken.

Kapitel 6 zeigt fortgeschrittenere Möglichkeiten auf, die Ihnen CSS eröffnet. So können Sie zum Beispiel festlegen, wie das Dokument beim Ausdruck aussehen soll, oder Schriftarten aus einer Schriftartendatei verwenden. Dieses Kapitel kann am ehesten übersprungen werden, denn:

1. die heutigen Browser unterstützen den Befehlssatz, den Sie hier kennen lernen, (noch) nicht und
2. die sehr speziellen Funktionen sind mehr für den fortgeschrittenen Web-Designer von Bedeutung.

Typografie im Internet ist Thema des 7. Kapitels. Es soll Ihnen ein paar Regeln zur Gestaltung nahe bringen. Typografie ist ein häufig unterschätzter Faktor für den Erfolg und die Akzeptanz einer Webseite.

Kapitel 8 ist interessant für Leser, die keine Lust haben, den ganzen Code von Hand einzutippen. Hier geht es nämlich um CSS-Editoren. Dabei stelle ich Ihnen die zwei wichtigsten vor und unterziehe sie einem Test. Eine anschließende Gegenüberstellung zeigt auf, wo die Editoren ihre Schwächen und Stärken haben.

Zukunftsweisend ist auch XML. Deshalb empfehle ich, das 9. Kapitel zu lesen, damit Sie mit CSS eine XML-Datei grafisch anzeigen können und auch für die Zeit »nach HTML« am Ball bleiben.

Wenn Sie wissen wollen, wo Sie noch mehr über CSS erfahren können, ist Kapitel 10 hilfreich. Hier werden URLs genannt und getestet, unter denen Sie zusätzlich Informationen finden.

Mit diesem Überblick können Sie nun selbst entscheiden, ob Sie das Buch als ganzes lesen oder einzelne, für Sie wichtige Kapitel herausgreifen.

Was Ihnen die Icons im Buch mitteilen

Sie werden im Buch immer wieder auf Icons stoßen, die auf besonders wichtige oder wertvolle Informationen aufmerksam machen sollen. Sie erfahren hier, wo Sie aufpassen müssen, was Sie tun sollten und was nicht.

Sehen Sie dieses Icon, so heißt es aufgepasst. Etwas könnte hier nicht funktionieren oder wird oft falsch gemacht.

Dieses Icon gibt Ihnen einen Hinweis.

Bei diesem Icon finden Sie einen Tipp. Sie können sich das Leben damit einfacher machen und Ihr Wissen vertiefen.

Hier folgt ein Beispiel zum Gelernten.

Welche Systemvoraussetzungen Sie brauchen

Ich habe alle Beispiele unter *Windows 98 SE* verfasst. Ist also vom »Betriebssystem« die Rede, so meine ich *Windows 98 SE.* Alle Beispiele habe ich mit dem Browser *Microsoft Internet Explorer 6.0* getestet. Sollte dieser Browser einmal »versagt« haben, wurde zum *Netscape 6* gegriffen. Wenn auch der etwas nicht wie gewünscht anzeigen konnte, kam der *Opera-Browser* zum Zug.

Alles in allem benötigen Sie von der technischen Seite gesehen:

- ✘ einen Computer (klar...)
- ✘ *Windows 95/98/2000/ME/NT/XP*
- ✘ einen Browser (es sollte ein *Internet Explorer* sein; optimal wäre *IE 6)*
- ✘ evtl. ein CD-ROM-Laufwerk (wenn Sie sich das lästige Abtippen von Code sparen wollen, da alle Beispiele auch auf der Buch-CD vorhanden sind)

- ✘ einen einfachen ASCII-Texteditor, wie zum Beispiel *Notepad,* der bei einem Windows-Betriebssystem schon vorinstalliert ist

Am Besten ist es meiner Erfahrung nach, wenn Sie vor dem eingeschalteten Rechner arbeiten, um die Beispiele gleich ausprobieren und gestellte Aufgaben lösen zu können.

Welches technische Verständnis notwendig ist

Dieses Buch verspricht einen Start ohne Vorwissen. Allerdings sollten gewisse Grundkenntnisse im Umgang mit PCs vorhanden sein. Wenn Sie also nicht wissen, wie Sie Ihr Betriebssystem zu bedienen haben, sollten Sie dieses Defizit vor dem Weiterlesen erst einmal ausgleichen. Voraussetzung ist, dass Sie:

- ✘ mit *Windows* umgehen können (im Buch wird mit *Windows* gearbeitet),
- ✘ das Dateisystem von *Windows* kennen,
- ✘ wissen, wie man Programme aufruft,
- ✘ Dateien speichern können.
- ✘ Weitergehende HTML-Kenntnisse, als sie im Schnellkurs in Kapitel 2 dargestellt werden, sind nicht notwendig, schaden aber auch nicht. Falls Sie sich tiefer in HTML einarbeiten wollen, empfehle ich das »HTML Kompendium« von Günter Born, ISBN 3-8272-5830-8.

Was die CD zum Buch enthält

Die CD bietet vieles, was Ihnen die Arbeit mit diesem Buch erleichtert.

In erster Linie enthält sie alle vorkommenden Beispiele. Nach jedem Beispiel folgt im Text des Buches der Hinweis, den Code unter einem bestimmten Namen zu speichern. Unter diesem Namen finden Sie das Beispiel dann auch auf der CD in den Unterverzeichnissen zu den einzelnen Kapiteln, die Sie wiederum im Ordner *buchdaten* öffnen können. Darüber hinaus enthält das Verzeichnis *Tools* zwei CSS-Editoren. Diese werden getestet, damit Sie sich ein genaues Bild von diesen Programmen machen und ihren Einsatz erwägen können.

Last but not least, sind auf der Buch-CD die Browser *Mirosoft Internet Explorer 6, Netscape Navigator 6* und *Opera 5* enthalten. Im Buch wird der Internet Explorer verwendet. Die Browser finden Sie im Verzeichnis *Browser.*

KAPITEL 1

Cascading Stylesheets – Was ist das?

Wenn Sie beim Begriff Stylesheet bisher nicht wussten, wovon die Rede war, sind Sie hier richtig, bei der Einführung zum Thema Cascading Stylesheets.

Sie lernen in diesem Kapitel:

- ✘ was »Stylesheet« bedeutet,
- ✘ für welchen Zweck man CSS entwickelte,
- ✘ wozu CSS heute eingesetzt wird,
- ✘ welche Browser CSS unterstützen.

1.1 Das Stylesheet – eine allgemeine Definition

Stylesheet bedeutet, aus dem Englischen übersetzt, soviel wie Stilvorlage, Formatvorlage. Aus dieser Übersetzung lässt sich, gerade im Zusammenhang mit HTML, schon eine Menge ableiten.

Wenn Sie eine Homepage erstellen und deren Erscheinungsbild festlegen wollen, so können Sie einen eigenen »Stil« kreieren, den Sie als Stilvorlage oder Stylesheet programmieren. Für die Definition des Stylesheets verwenden Sie die Sprache CSS.

Bei XML ist das sehr ähnlich: Auch hier beschreiben Sie mit Hilfe von Stylesheets die Formatierung von Daten (nur dass Sie sich in XML die Struktur der Daten selbst definieren und nicht an die Vorgaben des HTML-Standards gebunden sind).

Ganz allgemein wird in einem Stylesheet also das Aussehen bestimmter Inhalte auf dem Bildschirm oder auf Papier festgelegt – in den meisten Fällen dürfte es sich um Text und Zahlen handeln. In diesem Buch verwenden wir für die Erstellung dieser Stylesheets die Sprache CSS.

CSS wurde vom W3C entwickelt. Dieses Kürzel steht für ein Konsortium, das für fast alle Web-Standards verantwortlich ist (CSS, HTML, SVG, ...). Wenn Sie an den aktuellen Standards und Informationen über neueste Entwicklungen interessiert sind, besuchen Sie einfach die Webseite des W3C unter *www.w3.org*.

1.2 CSS – Ursprünglicher Gedanke und heutiger Einsatz

HTML-Programmierer werden sicher wissen, wie schwierig es sein kann, das Aussehen einer Webseite auf mehrere Browser abzustimmen. Was im *Internet Explorer* fabelhaft aussieht, wirkt im *Netscape Navigator* furchtbar, und umgekehrt. Auch wenn sich beide Konkurrenten heute wieder mehr an die Standards halten, ist es immer noch schwierig genug, beiden Browsern gerecht zu werden.

CSS sollte diese Probleme aus der Welt schaffen, dem Web-Designer ermöglichen, optimale Resultate mit seiner Programmierung zu erzielen, Einstellungen im Browser des Betrachters zu umgehen und nicht zuletzt die eingeschränkten Formatierungsmöglichkeiten von HTML erweitern. Selbst eine gewisse Plattformunabhängigkeit sollte geschaffen werden.

Eine in sämtlichen Belangen befriedigende Lösung hat sich durch die Einführung der CSS-Stylesheets jedoch leider nicht ergeben. Beispielsweise können immer nur noch die Schriftarten angezeigt werden, die das System des Betrachters für den benutzten Browser bereitstellt. Es gibt zwar die Möglichkeit, downloadbare Schriften zu verwenden, dies erfordert aber zusätzlichen Programmieraufwand (siehe Kapitel 6). Ansonsten ist die Umsetzung des CSS-Codes mit der des HTML-Codes vergleichbar: Nicht immer halten sich die Browser wirklich an die Standards.

Nichtsdestoweniger ist es mit CSS möglich, das Design einer Webseite im Wesentlichen zu fixieren. Was Sie in HTML mühsam und vielleicht sogar mehrmals definieren, definieren Sie in CSS ein einziges Mal für das gesamte

Dokument oder auch nur für einen bestimmten Abschnitt. Weitere Vorteile sind:

- Sie können mit CSS eine Webseite schnell und nach Belieben formatieren. In HTML ist die Formatierung doch eher zeitraubend und umständlich. In CSS geht's schneller und einfacher.
- Sie haben die Möglichkeit, eigene Klassen zu definieren. So ist es beispielweise möglich, jeder Klasse eine andere Formatierung zuzuweisen und diese dann auf das Dokument zu übertragen. Haben Sie auf Ihrer Homepage zum Beispiel eine Kinder- und Jugendecke eingerichtet, so sollten Sie eine fetzigere Schriftart wie »Comic Sans MS« und eine große Schrift verwenden. Für den »ernsten« Teil Ihrer Homepage verwenden Sie dann etwa die Schriftart »Verdana« und eine kleinere Schrift. Realisieren ließe sich dies, indem Sie eine Klasse »Jugend« und eine Klasse »Standard« definieren.
- Was in HTML strikt vorgegeben ist, kann mit CSS variiert werden – zum Beispiel kann der Webprogrammierer die Größe einer Überschrift selbst bestimmen.
- CSS kann auf vierfache Art und Weise verfasst werden:
  1. in einer externen Stylesheet-Datei
  2. intern in der HTML-Datei
  3. in den HTML-Tags über das Attribut `style` (siehe Kapitel 3)
  4. über den Befehl `@import` (siehe Kapitel 4)
- Außerdem können Sie Stylesheets von jedem beliebigen Server aus mit Ihrer Webseite verbinden, sprich Sie könnten ein Stylesheet von der Homepage der Firma *Microsoft* verwenden (sofern Sie die Genehmigung dafür haben).

1.3 Die Kompatibilität von Internet Explorer und Netscape Navigator mit CSS

Ähnlich wie HTML gibt es CSS in verschiedenen Sprachversionen: 1.0, 2.0 und 3.0, wobei CSS 3.0 zur Zeit noch ein Working Draft (Arbeitsentwurf) ist.

Die Sprachversion 1.0 wurde im Jahre 1996 verabschiedet, zwei Jahre später, 1998, die Version 2.0. Version 3.0 ist erst Mitte 2001 als Working Draft erschienen. Da CSS 3.0 sehr jung ist, wird es wohl wieder 1-2 Jahre dauern,

bis es zum Standard wird und die Browser es großteils umsetzen können. Aus diesem Grund wird hier auf die Darstellung von CSS 3.0 verzichtet.

Der *Netscape Navigator 4.x* interpretiert CSS 1.0 fast vollständig, sowie einen kleinen Teil der Sprachversion 2.0. Mit dem *Navigator 6.0* verspricht *Netscape* 100% Kompatibilität mit dem W3C, d.h. also mit HTML und anderen Standards. Tatsächlich setzt der *Navigator 6.0* HTML zu 100% und CSS zu 98% korrekt um. Der Netscape Navigator ist Bestandteil der meisten Linux-Distributionen und ist zum Beispiel in SuSE-Linux der Standard-Browser unter KDE.

Der *Internet Explorer 3.0* interpretierte fast den vollen Umfang der Version 1.0. Im *Internet Explorer 4.0* kamen ein Teil der Version 2.0 sowie einige von *Microsoft* selbst entwickelte CSS-Angaben hinzu. Im *Internet Explorer 5.5* werden nun die CSS-Version 1.0 komplett und die CSS-Version 2.0 teilweise unterstützt. Im *Internet Explorer 6* wird CSS 2.0 stärker, CSS 3.0 noch nicht unterstützt.

Ein Browser, den ich aufgrund seiner W3C-Standard-Interpretation sehr schätze, ist *Opera 5*. Mittlerweile ist er Freeware, allerdings erscheint dann eine kleine Opera-Werbeleiste, die für 30$ ausgeblendet werden kann. Soweit mir bekannt ist, wurde der *Opera* aus Unzufriedenheit mit der unterschiedlichen Interpretierung des W3C-Standards durch *IE* und *NN* entwickelt, ist aber selbst auch noch nicht 100% kompatibel zum W3C-Standard. Auch Opera ist für Linux erhältlich.

Es gibt weitere grafische Browser, z.B. den KDE-Browser Konqueror, und rein textbasierte Webbrowser wie Lynx. Zum Erlernen von CSS können diese hier aber unberücksichtigt bleiben.

Wenn Sie als Web-Designer bei einer Firma arbeiten, sollten Sie versuchen, in den beiden marktführenden Browsern, *Internet Explorer* und *Netscape Navigator*, ein einheitliches Bild zu erzielen. Sehen Sie sich die gleiche HTML-Datei in beiden Browsern an und halten Sie die Unterschiede fest. Versuchen Sie, Fehler für den betreffenden Browser zu korrigieren und sehen Sie sich das Resultat abermals in beiden Browsern an. Das ist zwar tatsächlich so umständlich, wie es hier wirkt, aber immer noch der schnellste Weg zum Ziel.

Das unterschiedliche Verhalten von *Internet Explorer* und *Netscape Navigator* lässt nicht zu, in einem Buch zwischen beiden Browsern hin und her zu springen. Man muss sich für einen der beiden entscheiden. Ich bevorzuge den *Internet Explorer*, da er, laut einer Umfrage, von etwa 80% aller Surfer eingesetzt wird (und meines Erachtens leichter zu handhaben ist).

1.4 Zusammenfassung

Über CSS können Sie nunmehr schon Folgendes berichten:

- ✘ Mit CSS spezifiziert man Stylesheets. Über diese Stylesheets kann man den Stil einer Webseite (insbesondere Formatierungen und Layout) definieren).
- ✘ CSS hat noch nicht ganz das erreicht, was geplant war.
- ✘ CSS-Code kann auf vierfache Art geschrieben werden.
- ✘ Es gibt drei Sprachversionen von CSS.
- ✘ Die Browser Internet Explorer und Netscape Navigator interpretieren CSS teilweise verschieden.
- ✘ Alte Browser wie Lynx und Mosaic können CSS nicht interpretieren.
- ✘ Für die Entwicklung der Sprache CSS ist das W3C zuständig.

1.5 Fragen und Antworten

1.5.1 Fragen

1. Was ist ein Stylesheet?
2. Was wollte man mit CSS ursprünglich erreichen?
3. Welche vier Arten der Einbindung von CSS in HTML unterscheidet man?

1.5.2 Antworten

1. Ein Stylesheet ist eine Formatvorlage, die das Aussehen bestimmter Daten beschreibt. Stylesheets für HTML-basierte Webseiten werden gemäß der CSS-Spezifikation aufgesetzt. Für XML ist CSS insofern von Bedeutung, da durch CSS eine formatierte Ansicht der XML-Tags möglich wird.
2. Angesichts der unterschiedlichen Interpretierung der HTML-Sprache durch den Internet Explorer und den Netscape Navigator wollte man eine einheitliche Lösung bereitstellen, die nicht nur dieses Problem aus der Welt schaffen, sondern auch die Möglichkeiten von HTML erweitern sollte.
3. Man kann CSS aus einer externen Datei einbinden, CSS intern in der HTML-Datei definieren oder die Werte im jeweiligen Tag durch das Attribut `style` vornehmen. Außerdem kann der CSS-Befehl `@import` verwendet werden.

KAPITEL 2

HTML-Schnellkurs

Bevor Sie mit CSS arbeiten können, benötigen Sie Grundkenntnisse in den ML-Technologien[1] HTML, der Sprache des WWW, und XML. Wenn Sie HTML schon kennen, können Sie dieses Kapitel bis zum Abschnitt über XML getrost überspringen. Wenn Sie XML auch schon können: Klasse. Allen anderen empfehle ich, dieses Kapitel zu lesen, denn es ist für das Verständnis von CSS unentbehrlich.

Sie lernen in diesem Kapitel:

- ✘ wie HTML-Code aufgebaut ist,
- ✘ wie man Text formatiert,
- ✘ wie man Hyperlinks einsetzt,
- ✘ wie Multimedia-Elemente (z.B. Bilder, Videos, etc.) eingebunden werden,
- ✘ wie besondere Marken eingesetzt werden (z.B. Kommentare),
- ✘ wie man Listen aufsetzt,
- ✘ was bei Sonderzeichen zu beachten ist,
- ✘ wie man ein Tabellenlayout erstellt,
- ✘ was XML ist,
- ✘ wie XML eingesetzt wird.

1. ML steht für Markup Language, also Auszeichnungssprache. Dazu zählt man HTML, XML, SGML etc.

2.1 Der Aufbau eines HTML-Dokuments

In HTML gibt es für die Formatierung von Text, das Einfügen von Bildern etc. so genannte Marken, auch Tags genannt. Diese Tags liefern dem Browser die Informationen, die er für die Darstellung des Dokuments benötigt. Wir unterscheiden zwischen Starttag und Endtag. Das Endtag wird nicht immer gebraucht.

Mit dem Starttag definieren wir den Anfang eines bestimmten Formatierungsbereichs. `<FONT face="Arial">` legt beispielsweise fest, dass der nachfolgende Text die Schriftart Arial hat, so lange bis das Endtag `</FONT>` erscheint.

Ist Ihnen an dem Endtag etwas aufgefallen? Es setzt sich immer aus dem Starttag plus Slash (»/«) zusammen, z.B. `</FONT>`. Es ist das Pendant zum Starttag und markiert das Ende eines bestimmten Formatierungsbereichs.

2.1.1 Ein HTML-Dokument

Als HTML-Code wird der Code bezeichnet, der sich zwischen den Tags `<HTML>` und `</HTML>` befindet. Ein HTML-Dokument beginnt also immer mit `<HTML>` und endet mit `</HTML>`. Was hier Start- und Endtag ist, dürfte klar sein.

Wenn Sie ein HTML-Dokument schreiben, dann verwenden Sie dazu, zumindest in diesem Buch, den »Notepad«-Editor von Microsoft. Den Code speichern Sie dann mit der Endung ».htm« oder ».html« ab. Diese beiden Dateierweiterungen repräsentieren immer ein HTML-Dokument.

Zur Veranschaulichung des bisher Gesagten gehen wir gleich medias in res: Öffnen Sie einen einfachen ASCII-Editor, z.B. Notepad (im Startmenü den Befehl AUSFÜHREN aufrufen, `notepad` eingeben und abschicken).

Geben Sie im Notepad-Editor den folgenden Code ein:


```
<HTML>
</HTML>
```

Speichern Sie die Datei unter *HTML.htm* ab.

HTML-Code wird mit der Dateiendung .htm bzw. .html abgespeichert. Nur diese beiden Formate (die sich lediglich im Namen unterscheiden) kann der Browser als Webseite interpretieren.

Starten Sie nun Ihren Browser und öffnen Sie die Datei.

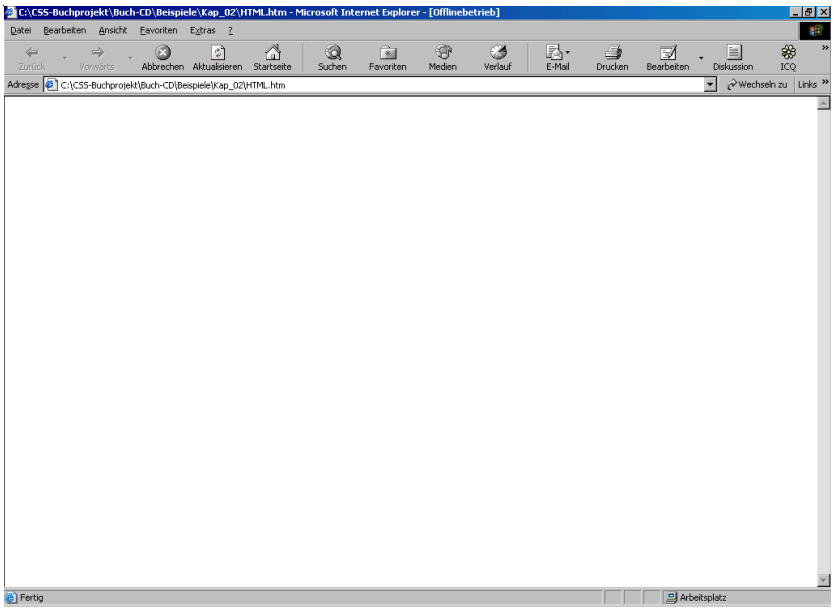

Abb. 2.1: Das fertige HTML-Dokument.

Nein, Sie haben nichts falsch gemacht. Der Browser zeigt wirklich nichts an. Denn wir haben ihm ja auch noch nicht »gesagt«, was er anzeigen soll.

2.1.2 Der Bereich <HEAD>

Sie wissen nun, dass der HTML-Code im HTML-Bereich, das heißt, zwischen `<HTML>` und `</HTML>`, steht.

Innerhalb des HTML-Bereichs befindet sich ein weiterer Bereich: Der HEAD-Bereich. Im HEAD-Bereich stehen alle Informationen, die nicht direkt im Browser angezeigt werden, sondern eine andere Funktion haben, zum Beispiel das Tag `<TITLE>`. Mit diesem Tag können Sie einen Titel für das HTML-Dokument festlegen. Laden Sie das Beispiel aus dem vorherigen Abschnitt in den *Notepad*-Editor und erweitern Sie es folgendermaßen:


```
<HTML>
   <HEAD>
      <TITLE>Mein erstes HTML-Dokument mit Titel</TITLE>
   </HEAD>
</HTML>
```

Sie können zum schnelleren Bearbeiten des HTML-Codes mit Notepad einen Eintrag »Mit Notepad öffnen« in das Kontextmenü der HTML-Datei integrieren. Anhang B beschreibt, wie das geht.

Speichern Sie den Code unter *HEAD.htm*. Am HTML-Dokument hat sich nichts geändert, jedoch an der Titelleiste des Browsers (siehe Abbildung 2.2).

Abb. 2.2: Der Text der Titelleiste wurde verändert.

2.1.3 Der Bereich <BODY>

Es folgt der Teil des HTML-Dokuments, in den der Text eingefügt wird. Nachdem Sie mit `<HEAD>` einen Kopf haben, brauchen Sie noch einen Körper: Den BODY-Bereich.

Erweitern wir das Beispiel noch ein wenig:


```
<HTML>
   <HEAD>
      <TITLE>Mein erstes HTML-Dokument mit Titel und Text</TITLE>
   </HEAD>

   <BODY>
      Nun kommt endlich ein Text dazu.
   </BODY>
</HTML>
```

Achten Sie unbedingt darauf, dass Sie `<TITLE>` im `<HEAD>`-Bereich einfügen. Sollte `<TITLE>` mal in den `<BODY>`-Bereich rutschen, wird der vermeintliche Titel als Text im Dokument angezeigt.

Speichern Sie den Code nun unter *BODY.htm* und betrachten Sie das Ergebnis im Browser.

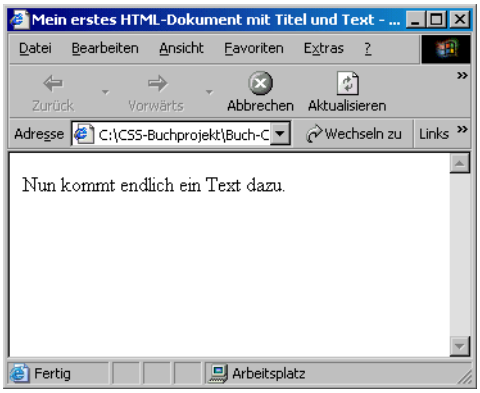

Abb. 2.3: Da kommt Freude auf! Endlich sehen wir Text.

Wenn Sie in HTML programmieren, sollten Sie darauf achten, in den Code-Beispielen mit der ⇥-Taste Einrückungen vorzunehmen. Vielleicht können Sie den Beispielen schon entnehmen, wann ein Einzug zu setzen ist. Dies dient der Übersichtlichkeit des Codes und kann für Sie und andere bei der Bearbeitung vor allem umfangreicherer Seiten eine immense Hilfe sein.

Damit sind die allergrundsätzlichsten Dinge zu HTML schon gesagt und es kommt die...

2.1.4 Zusammenfassung

In diesem Abschnitt haben Sie die erste Stufe der HTML-Grundlagen erklommen. Sie haben gelernt:

- ✘ Im HTML-Bereich steht der HTML-Code.
- ✘ Im HEAD-Bereich stehen Informationen, die nicht im Browser angezeigt werden, sondern eine andere Funktion besitzen.
- ✘ Im BODY-Bereich steht der Text, der im Browser angezeigt werden soll.
- ✘ Es ist sinnvoll, auf eine saubere Struktur des Codes zu achten.

2.2 Eintönige Texte auflockern

Theoretisch könnte man auf die soeben besprochene, sehr einfache Art und Weise beliebig lange Texte darstellen. Doch wirken umfangreiche, gleichförmige Textpassagen schwarz auf weiß auf den Betrachter häufig langweilig. Er will Abwechslung, insbesondere im Web, wo Ihre Seite mit Tausenden anderen, teilweise extrem aufwändig gestalteten konkurrieren muss.

Im vorherigen Abschnitt habe ich auf Einrückungen im Code hingewiesen.

Sie sollten weiterhin den HTML-Code, wenn es irgend möglich ist, genauso abschreiben, wie er im Buch abgedruckt ist. Das gilt später auch für CSS-Code. So gewöhnen Sie sich gleich eine übersichtliche Struktur an.

2.2.1 Überschriften in HTML

HTML sieht für Überschriften sechs Standardgrößen vor. Diese verbergen sich hinter den Tags <H1> bis <H6>. H steht hierbei für Header. <H1> ist die größte Überschrift, <H6> die kleinste. Mit einem kleinen Beispiel zur Illustration ist dann auch schon alles zum Thema Überschriften gesagt:


```
<HTML>
   <HEAD>
      <TITLE>Die 6 verschiedenen Überschriften</TITLE>
   </HEAD>

   <BODY>
      <H1>Überschrift 1</H1>
      <H2>Überschrift 2</H2>
      <H3>Überschrift 3</H3>
      <H4>Überschrift 4</H4>
      <H5>Überschrift 5</H5>
      <H6>Überschrift 6</H6>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *headers.htm* ab. Wenn alles richtig war, sollten Sie diese Überschriftsformatierungen im Browser angezeigt sehen.

Abb. 2.4: Die sechs verschiedenen Überschriften-Größen.

2.2.2 Times New Roman – wie geht's anders?

Times New Roman ist eine Standardschriftart von Windows und wird auch als Standard für HTML-Dokumente vorgegeben (dies kann man jedoch im *Internet Explorer* unter EXTRAS | INTERNETOPTIONEN | SCHRIFTARTEN ändern). Das bedeutet in letzter Konsequenz, dass im HTML-Dokument nur ein und dieselbe Schrift verwendet wird – es sei denn, Sie definieren im HTML-Code eine andere beziehungsweise weitere Schriftarten.

Sie können die Schriftart wechseln, eine Schrift als Standard deklarieren, oder auch beides. Wenn Sie sie als Standardschrift deklarieren, wird sie zum Standard für das HTML-Dokument und ist für das gesamte Dokument gültig.

Befassen wir uns zunächst mit dem Wechseln der Schriftart. Dazu benötigen Sie ein neues Tag: `<FONT>`. Font ist Englisch und bedeutet Schrift. Wir haben also ein eigenes Tag nur für das Schriftbild.

Es gibt einige Tags, die nach dem HTML-4.0-Standard vermieden werden sollten. Als Ersatz soll CSS angewandt werden. Das `<FONT>`-Tag ist eines dieser Tags, wird aber noch recht häufig verwendet. In Anhang D finden Sie eine Liste der Tags, die nicht mehr benutzt werden sollen. Im weiteren Verlauf des Kapitels werde ich keinen besonderen Hinweis für veraltete Tags geben.

In diesem Zusammenhang kommt zum ersten mal ein Attribut ins Spiel. Attribute ergänzen ein Tag, indem sie diesem einen Wert zuordnen. In diesem Fall ist es für die Schriftart zuständig und heißt `face`. Kombiniert man das Tag mit dem Attribut, sieht das folgendermaßen aus:

```
<FONT face>
```

Zu Recht werden Sie feststellen, dass der Browser dadurch noch keine andere Schriftart anzeigen kann, da ja noch nirgends angegeben ist, *welche* Schriftart es denn nun sein soll. Ergänzen wir den Code also um einen Wert für unser Attribut:

```
<FONT face="Arial">
```

Das bedeutet: Der Text, der auf diese Zeile folgt, wird in der Schriftart Arial dargestellt, bis wir diese Anweisung mit `</FONT>` wieder aufheben. Man kann also sagen, dass `<FONT>` einen Bereich einleitet, den Bereich für die Schriftformatierung.

Folgendes Beispiel verdeutlicht die Einsatzmöglichkeiten von `<FONT>`.


```
<HTML>
   <HEAD>
      <TITLE>Meine erste Schriftformatierung</TITLE>
   </HEAD>

   <BODY>
      Ein Text <FONT face="Arial">kann auch</FONT> <FONT face="Comic Sans
MS">formatiert werden.</FONT>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *FONT.htm* und laden Sie die Datei in Ihren Browser.

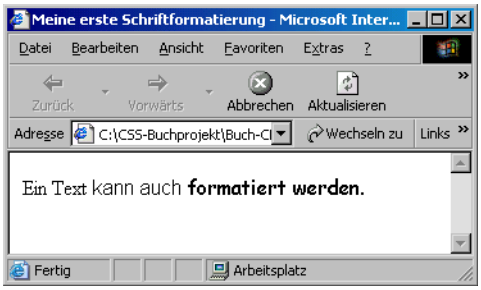

Abb. 2.5: Ein Text in drei Schriften formatiert.

Wenn Sie HTML vornehmlich erlernen, um eine Homepage zu erstellen, sollten Sie sich fragen, was passiert, wenn die verwendete Schriftart nicht auf dem Rechner des Betrachters installiert ist. Der Browser greift in einem solchen Fall auf die Standardschriftart zurück (unter Windows »Times New Roman«).

Wollen Sie aber genau das verhindern, so können Sie eine zweite, dritte oder vierte Schrift angeben Das bleibt ganz Ihnen überlassen. Sie sollten aber das Layout des HTML-Dokuments mit allen angegebenen Schriftarten testen.

Die Schriftarten werden durch Kommata voneinander getrennt. Zunächst wird die zuerst angegebene Schriftart verwendet. Sollte diese nicht installiert sein, wird die zweite verwendet, und so weiter.

```
<FONT face="Arial, Comic Sans MS, Verdana">
```

Und gleich wieder ein Beispiel: Um zu demonstrieren, was passiert, wenn die zuerst angegebene Schriftart nicht installiert ist, geben wir einfach eine an, die gar nicht existiert.


```
<HTML>
   <HEAD>
      <TITLE>Nicht installierte Schriftart</TITLE>
   </HEAD>

   <BODY>
      Dieses Dokument <FONT face="Eine_Schriftart, Arial">zeigt das Verhalten
bei einer</FONT> nicht installierten Schriftart.
   </BODY>
</HTML>
```

Speichern Sie die Datei unter *not_installed_font.htm*. Wenn Sie die Datei in Ihren Browser laden, wird dieser die alternative Schriftart Arial verwenden (siehe Abbildung 2.6).

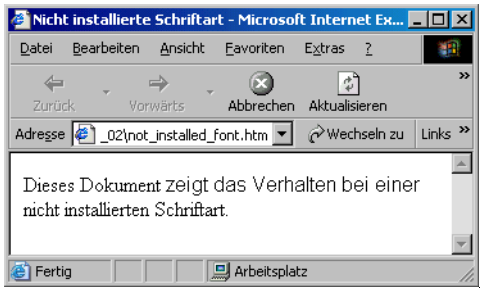

Abb. 2.6: So verhält sich der Browser, wenn eine Schriftart nicht installiert ist.

Ein weiteres sinnvolles Attribut zu <FONT> ist size, mit dem Sie die Schriftgröße ändern. Möglich sind sieben verschiedene Größen (1-7). Die Größen, die diesen Ziffern in Punkt zugewiesen sind, listet diese Tabelle auf:

Tabelle 2.1 Schriftgrößen

HTML-Größe	Größe in Punkt
1	8
2	10
3	12
4	14
5	18
6	24
7	36

2.2.3 Eine Schriftart zum Standard machen

Die von *Microsoft* vorgeschlagene Standardschriftart ist Times New Roman. Sie können aber auch selbst mit dem Tag <BASEFONT> allgemeine Einstellungen für das Schriftbild vornehmen.

<BASEFONT>, das die gleichen Attribute wie <FONT> besitzt, setzt man ein, um für das Schriftbild einen Standard zu definieren. Mit <BASEFONT> kann man deutlich machen, wie der Standard für das Schriftbild lautet. Wenn andere HTML-Programmierer Ihren Code lesen und dieser ziemlich lang ist, werden sie Ihnen dankbar sein, wenn Sie die Standardschrift mit <BASEFONT> definiert haben. Später, wenn wir CSS anwenden, wird das allerdings nicht mehr von großer Bedeutung sein.

Da <BASEFONT> gegenüber <FONT> keine großen Neuigkeiten mit sich bringt, noch ein kleines Beispiel ohne Screenshot des Resultats.


```
<HTML>
   <HEAD>
      <TITLE>Der Gebrauch von BASEFONT</TITLE>
   </HEAD>

   <BODY>
      <BASEFONT face="Comic Sans MS">
     Die Standardschriftart kann man <FONT face="Arial">auch beim Einsatz von
BASEFONT</FONT> ändern.
   </BODY>
</HTML>
```

Ein kleiner Hinweis zu dem Beispiel: Es wäre schön, wenn im Dokumenttext der Text `BASEFONT von` spitzen Klammern umgeben wäre. Dies ist jedoch nicht ohne Weiteres möglich, denn wenn Sie `»auch beim Einsatz von <BASEFONT>«` eintippen, wird `<BASEFONT>` als Tag und nicht als darzustellender Text interpretiert. Aus diesem Grund verzichte ich hier auf die Darstellung der spitzen Klammern. Später in diesem Kapitel kommt dieses Problem noch mal zur Sprache. Sie finden das Beispiel auf der CD-ROM unter dem Namen *BASEFONT.htm*.

2.2.4 Ein bisschen Farbe muss sein

Schwarz auf Weiß – wirkt eher trist, oder? Gerade wenn man eine private Homepage bastelt, sollte man auf keinen Fall schwarze Farbe auf weißem Grund verwenden. Oder doch nicht ausschließlich.

In diesem Unterabschnitt werden wir zum ersten Mal Farbe verwenden. Dazu sind aber Kenntnisse über das RGB-Farbsystem nötig.

Das RGB-Farbsystem

Zwar geht es mit den moderneren Browsern etwas leichter, doch sind auch noch einige alte Browser im Einsatz, die ohne das RGB-Farbsystem keine Farben darstellen können.

RGB steht für die Grundfarben **R**ot, **G**rün, **B**lau. Aus diesen drei Grundfarben können Sie alle beliebigen Farben mischen.

Ein Farbcode im RGB-Farbsystem sieht beispielsweise so aus:

```
#FF0000
```

Dieser Code steht für die Farbe Rot. Die Buchstaben FF geben den Farbwert für Rot an, die ersten zwei Nullen den Farbwert für Grün, die letzten zwei Nullen den Farbwert für Blau.

Der Farbwert für die drei Grundfarben ist der Anteil der jeweiligen Grundfarbe an der Gesamtfarbe. Der Farbwert wird im Hexadezimalsystem angegeben, kann höchstens FF (dezimal 255, maximaler Anteil) und muss mindestens 00 (dezimal 0, kein Anteil) betragen.

Das Hexadezimalsystem

Das Hexadezimalsystem ist ein Zahlensystem – wie das Dezimalsystem, mit dem wir rechnen. Das Dezimalsystem basiert dabei auf der Zahl 10, während das Hexadezimalsystem auf der Zahl 16 aufbaut. Das heißt, es gibt 16 Ziffern. Diese sind: 0-9, A (=10), B (=11), C (=12), D (=13), E (=14), F (=15).

Im Dezimalsystem ist die letzte Stelle die Einerstelle, die Vorletzte die Zehnerstelle, die Vorvorletzte die Hunderterstelle die Viertletzte die Tausenderstelle… Fällt Ihnen etwas auf? Richtig. Die Stellen werden nämlich jeweils mit 10 multipliziert, um auf die nächste Stelle zu kommen (Hunderterstelle (100) · 10 = Tausenderstelle (1.000)).

Und genauso geht's auch mit dem Hexadezimalsystem, eben mit der Zahl 16 als Basis: Einerstelle, Sechzehnerstelle, 256er-Stelle, 4.096er-Stelle… Hier wird die jeweilige Stelle mit 16 multipliziert, um auf die nächste Stelle zu kommen (Sechzehnerstelle (16) · 16 = 256er-Stelle (256)).

So entsteht eine Zahl im Dezimalsystem:

Zerlegen wir die Zahl 123 mal in ihre einzelnen Stellen:

1 = Hunderterstelle, 2 = Zehnerstelle, 3 = Einerstelle. Nun multiplizieren wir die 1 mit 100 = 100, die 2 mit 10 = 20 und die 3 mit 1 = 3. Zum Schluss wird addiert, 100 + 20 + 3 = 123, womit wieder die Ausgangszahl hergestellt wäre.

So entsteht eine Zahl im Hexadezimalsystem:

Zerlegen wir nun die Zahl 7B aus dem Hexadezimalsystem, um eine uns verständliche Zahl, sprich eine Dezimalzahl, zu erhalten:

7 = Sechzehnerstelle, B = Einerstelle. Wir multiplizieren nun wieder die 7 mit 16 = 112, die B (=11) mit 1 = 11 und addieren zum Schluss, 112 + 11 = 123.

Sie sehen also, dass das Hexadezimalsystem logisch aufgebaut ist, verwirrend ist nur die Zahl 16 als Grundzahl.

Wenn Sie einer Farbe den Anteil 255 geben, so ist diese Farbe voll in der Gesamtfarbe enthalten. Wenn Sie allen Farben den Anteil 255 geben, kommt als Gesamtfarbe Weiß heraus. Stellen Sie sich das so vor: Sie haben einen roten, einen blauen und einen grünen Strahler und richten diese auf einen Punkt auf dem Fußboden. Lassen Sie die Lampen aus, bleibt der Fußboden dunkel (schwarz). Schalten Sie alle ein, addieren sich die Farben auf dem Fußboden zu Weiß, da es mit jedem Strahler heller wird.

Mischen wir nun mal die Farbe Gelb. Sie besteht aus den Farben Rot und Grün. Wenn ein natürliches Gelb herauskommen soll, müssen wir diesen Farben den vollen Anteil geben und den RGB-Farbcode folgendermaßen angeben: 255 für Rot, 255 für Grün, 0 für Blau. Diese Zahlen – in das Hexadezimalsystem gesetzt – ergeben den Wert `#FFFF00`. Beachten Sie, dass immer zwei Stellen für einen Farbanteil angegeben werden müssen (im Beispiel `00` für Blau).

Den Farbanteil können Sie natürlich variieren. Wenn Sie den Farbanteil für Rot und Grün in obigem Beispiel verringern, kommt eine dunklere Farbe heraus.

Dies ist alles zum RGB-Farbsystem, das übrigens »nur« für HTML wichtig ist. In CSS kann man zwar auch damit arbeiten, für CSS verwendet man aber die englischen Namen der einzelnen Farben, da die Browser, die CSS unterstützen, auch diese englischen Farbnamen kennen.

Dem Text eine andere Farbe zuweisen

Diese RGB-Farbsystem will natürlich auch eingesetzt werden. Zur Einstellung der Schriftfarbe ziehen Sie wieder das Tag `<FONT>` heran, das ja für das gesamte Schriftbild zuständig ist. Das Attribut, das Sie für die Farbe benötigen, heißt `color`. Überlegen Sie, wie die Farbe Lila im RGB-Farbsystem auszusehen hat. Die Antwort finden Sie im folgenden Beispiel.

Überlegen Sie zuerst, aus welchen Grundfarben Lila besteht. Legen Sie dann die Farbanteile fest und setzen Sie sie der Reihenfolge nach in das Hexadezimalsystem. Vergessen Sie nicht das # vor den Farbanteilen!


```
<HTML>
   <HEAD>
      <TITLE>Eine veränderte Schriftfarbe</TITLE>
   </HEAD>

   <BODY>
      <FONT color=#FF00FF>Eine lila Schrift</FONT>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *color.htm*. Das Ergebnis sehen Sie in Abbildung 2.7.

Leider wird das Buch durchgehend schwarzweiß gedruckt. Deshalb sollten Sie das Beispiel selbst auf Ihrem Rechner nachvollziehen, damit Sie die Schriftfarbe erkennen können, da sie hier grau erscheint.

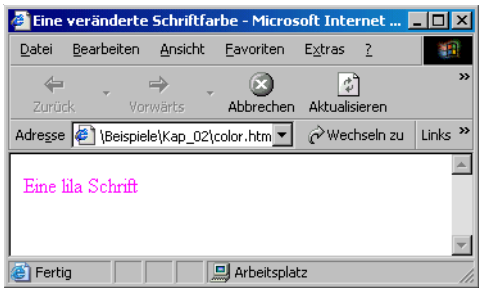

Abb. 2.7: Endlich können wir die Schriftfarbe ändern (im Bild wegen des Schwarz-Weiß-Drucks nicht erkennbar).

2.2.5 Die Standardfarbe mit <BASEFONT> oder <BODY> festlegen

Die Schriftfarbe kann auch als Standard definiert werden. Sie können dazu die schon bekannten Tags `<BASEFONT>` und `<BODY>` benutzen. In beiden Fällen wird die Farbe im RGB-Farbsystem angegeben.

In folgender Tabelle finden Sie eine Übersicht über die Attribute. Ein komplettes Beispiel ist wohl nicht notwendig.

Tabelle 2.2 Standardschriftfarbe setzen

Tag	Attribut	Anwendungsbeispiel
`<BODY>`	`Text`	`<BODY text="#FF0000">`
`<BASEFONT>`	`Color`	`<BASEFONT color="#00FF00">`

2.2.6 Texte ausrichten

Gewiss kennen Sie aus der Textverarbeitung die Begriffe Zeilenumbruch und Absatz. In diesem Unterabschnitt werden Sie lernen, wie Sie in HTML Zeilenumbrüche und Absätze realisieren. Des Weiteren werden Sie Texte linksbündig, rechtsbündig und zentriert ausrichten. Auch Blocksatz ist möglich.

Die Textausrichtung ist ein Aspekt der Typografie. Eine gute Typografie wird dem Betrachter der Webseite ansprechen, ihm die Texte übersichtlich darbieten und die Aufnahme der Information erleichtern – und ihn veranlassen, die Webseite wieder zu besuchen. Kapitel 7 beschäftigt sich mit den Regeln der Typografie, die Ihnen bei der Webgestaltung helfen sollen.

Befassen wir uns zunächst mit dem Zeilenumbruch. Er wird durch ein Tag bewirkt, welches kein Endtag besitzt: `<BR>`. Es definiert einen Zeilenumbruch, sprich der folgende Text wird in die nächste Zeile geschrieben (der Zeilenabstand bleibt dabei gleich).

Für den Absatz wird das Tag <P> verwendet, das nach dem W3C ein Endtag besitzen sollte. Moderne Browser ergänzen das Endtag aber automatisch, sodass man es auch weglassen kann. Ein Endtag können Sie dann einsetzen, wenn Sie auf eine besonders genaue Struktur Wert legen. Im Gegensatz zu <BR> erzeugt <P> einen größeren Zeilenabstand vor der ersten und nach der letzten Zeile des Absatzes. Mit <BR> können Sie auch innerhalb eines Absatzes einen Zeilenumbruch erzeugen.

Ein Beispiel dazu:


```
<HTML>
   <HEAD>
      <TITLE>Zeilenumbuch und Absatz</TITLE>
   </HEAD>

   <BODY>
      Dies ist ein ganz normaler Text.
      <P>Dies ist ein Absatz, der mit dem P-Tag geschaffen wurde. Er ist vom
vorangehenden Text durch Zeilenumbruch und größeren Zeilenabstand getrennt.
</P>
      <P>Dies ist ein weiterer Absatz. Dies ist ein weiterer Absatz. Dies ist
ein weiterer Absatz. Dies ist ein weiterer Absatz. Dies ist ein weiterer
Absatz. Dies ist ein weiterer Absatz. Dies ist ein weiterer Absatz. </P>
     <P>Dies ist noch ein weiterer Absatz.<BR>Das war ein Zeilenumbruch inner-
halb eines Absatzes.</P>

      Das ist wieder normaler Text.
   </BODY>
</HTML>
```

Speichern Sie den Code unter *BRP.htm*.

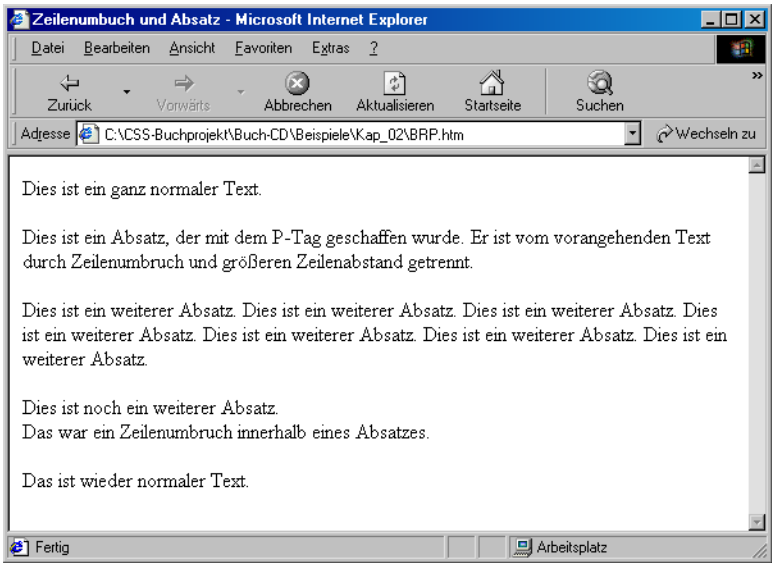

Abb. 2.9: Ein Text mit Zeilenumbruch und Absätzen.

Für die Ausrichtung des Textes an der linken bzw. rechten Seite ist kein eigenes Tag, sondern lediglich ein neues Attribut notwendig: `align`. Die möglichen Werte sind:

Tabelle 2.3 Textausrichtung

Ausrichtung	Wert für align
links	`left`
rechts	`right`
zentriert	`center`
Blocksatz	`justify`

Soll ein Text zentriert ausgerichtet werden, so kann auch ein eigenes Tag, das `<CENTER>`-Tag, angewandt werden. Auch hierzu noch ein Beispiel:


```
<HTML>
   <HEAD>
      <TITLE>Ausrichtung von Text</TITLE>
   </HEAD>

   <BODY>
      Die Ausrichtung von Text in HTML ist einfach.<P align="right">Man kann
den Text rechts ausrichten.</P>
   <P align="left">Man kann den Text aber auch links ausrichten.</P>
   <P align="justify">Um den Blocksatz zu testen, benötigen wir einen
laaaaaaaaangen Text. Um den Blocksatz zu testen, benötigen wir einen
```

```
laaaaaaaaangen Text. Um den Blocksatz zu testen, benötigen wir einen
laaaaaaaaangen Text. Um den Blocksatz zu testen, benötigen wir einen
laaaaaaaaangen Text. Um den Blocksatz zu testen, benötigen wir einen
laaaaaaaaangen Text. Um den Blocksatz zu testen, benötigen wir einen
laaaaaaaaangen Text. Jetzt reicht's.</P>
     <CENTER>Für die zentrierte Ausrichtung ist ein eigenes Tag vorhanden. Man
kann aber auch über das P-Tag eine zentrierte Ausrichtung vornehmen.</CENTER>
   </BODY>
</HTML>
```

Wenn Sie den Code unter *align.htm* gespeichert haben und die Datei öffnen, müssten Sie Folgendes sehen:

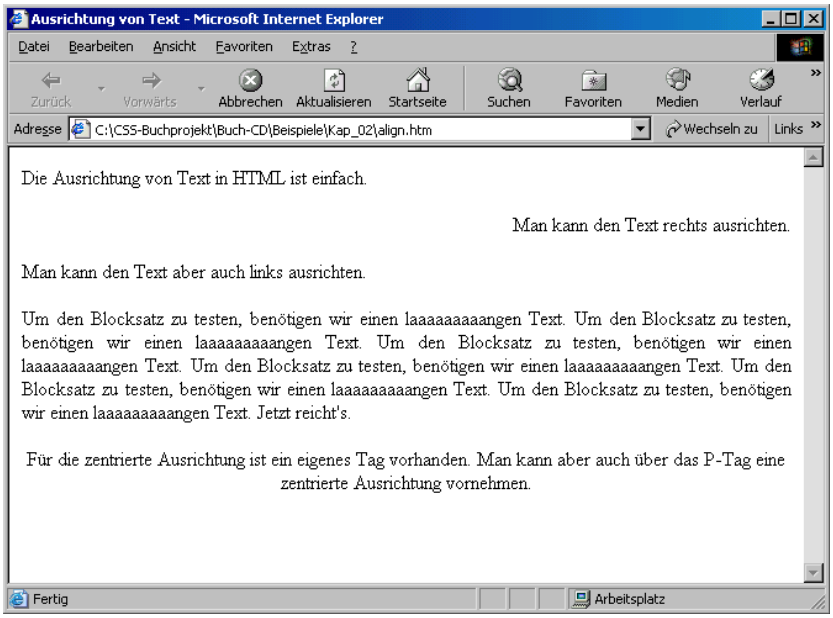

Abb. 2.10: Die Ausrichtung von Text ist in HTML ziemlich einfach.

Mit dem `<BR>`-Tag ist eine Ausrichtung von Text leider nicht möglich. Wenn Sie also nach einem Zeilenumbruch, statt nach einem Absatz, eine Veränderung an der Ausrichtung des Textes vornehmen möchten, so benutzen Sie das `<DIV>`-Tag. Dieses kennzeichnet einen Bereich. Über verschiedene Attribute kann man besondere Merkmale dieses Bereichs festlegen (beispielsweise `align` für die Ausrichtung). Das Tag wird von Zeilenumbrüchen umschlossen, sodass das `<BR>`-Tag vor `<DIV>` nicht erforderlich ist. Hier noch ein Beispiel dazu. Sie finden es auf der Buch-CD unter *Beispiele\Kap_02\DIV.htm*.

```
<HTML>
   <HEAD>
      <TITLE>Ausrichtung mit DIV</TITLE>
   </HEAD>

   <BODY>
      Normaler Text.<BR>
      Eine Ausrichtung über das BR-Tag ist nicht möglich. Will man also eine
Ausrichtung für die nächste Zeile erreichen, so
      <DIV align="center">setzt man das DIV-Tag ein.</DIV>
   </BODY>
</HTML>
```

2.2.7 Fett & Co.

Sie wissen nun alles, was es über die Schriftformatierung mit Hilfe von <FONT> zu sagen gibt. Es gibt aber noch mehr Möglichkeiten zur Textdarstellung. Wie zum Beispiel in Word können Sie auch in HTML Text fett, kursiv, unterstrichen und durchgestrichen darstellen.

In der folgenden Tabelle finden Sie die dazu einsetzbaren Tags.

Tabelle 2.4 Zeichenformate

Tag	Darstellung
<B>...</B>	fett
<I>...</I>	kursiv
<U>...</U>	unterstrichen
<S>...</S>	durchgestrichen
<TT>...</TT>	Schreibmaschine
<CITE>...</CITE>	Darstellung eines Zitats

Das folgende Beispiel wendet alle oben aufgelisteten Formate an:

```
<HTML>
   <HEAD>
      <TITLE>Fett & Co.</TITLE>
   </HEAD>

   <BODY>
      Ein Text kann <B>fett</B> sein, <I>kursiv</I> oder auch <U>unterstri-
chen</U>. Außerdem ist es möglich, den Text <S>durchzustreichen</S> und ihn
als <TT>Schreibmaschine</TT> darzustellen. Auch ein <CITE>Zitat</CITE> kann
eingefügt werden.
   </BODY>
</HTML>
```

Speichern Sie den Code unter *fettuco.htm* ab. So sieht's aus:

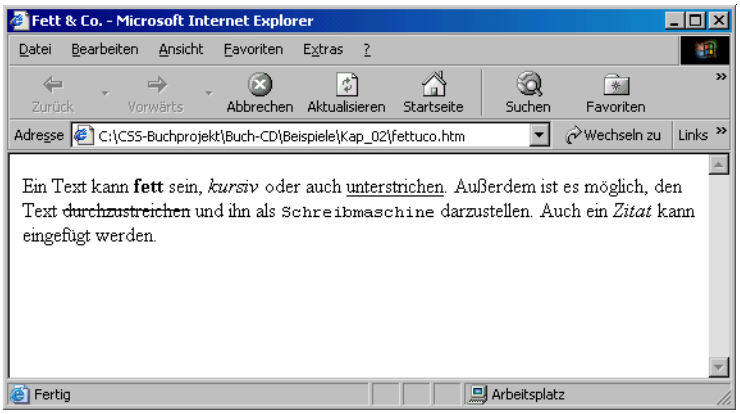

Abb. 2.11: Nun sind nahezu alle HTML-Formatierungsmöglichkeiten komplett.

2.2.8 Buntes HTML-Dokument

Nicht nur die Farbe des Textes kann verändert werden, nein, auch die Hintergrundfarbe des gesamten HTML-Dokuments.

Sie benötigen lediglich das `<BODY>`-Tag und ein neues Attribut namens `bgcolor`. Mit `bgcolor` setzen Sie die Hintergrundfarbe im RGB-Farbcode.

Dieses Beispiel zeigt es:


```
<HTML>
   <HEAD>
      <TITLE>Ein HTML-Dokument mit Hintergrundfarbe</TITLE>
   </HEAD>

   <BODY bgcolor=#FFFF00 text=#FF0000>
      Ein roter Text auf gelbem Hintergrund
   </BODY>
</HTML>
```

Speichern Sie den Code unter *bgcolor.htm* ab. Das Resultat:

Abb. 2.12: Roter Text auf gelbem Hintergrund (wegen Schwarz-Weiß-Druck nicht zu sehen).

2.2.9 Zusammenfassung

Nun wissen Sie allerlei über die Textgestaltung. Damit nicht das eine oder andere wieder in Vergessenheit gerät, eine kurze Zusammenfassung:

- Das `<FONT>`-Tag ist zuständig für Schriftart, -größe und -farbe.
- `<BASEFONT>` gibt das Standardschriftbild an. Das Tag ist wie `<FONT>` zu gebrauchen.
- Man kann mehrere Schriftarten als Standard angeben, für den Fall, dass eine Schriftart auf dem PC des Betrachters nicht installiert ist.
- Es gibt sechs verschiedene vordefinierte Überschriftengrößen. Die Tags dafür sind `<H1>` bis `<H6>`.
- Die Standardfarbe kann auch im `<BODY>`-Tag definiert werden. Das erforderliche Attribut ist `text`.
- Das Hexadezimalsystem basiert auf der Zahl 16.
- Sie haben das RGB-Farbsystem kennen gelernt.
- Der Farbanteil einer Grundfarbe an der Gesamtfarbe kann zwischen 255 und 0 liegen.
- Man kann Text auch fett, kursiv, usw. formatieren.
- Die Hintergrundfarbe des HTML-Dokuments wird im `<BODY>`-Tag festgelegt. Das notwendige Attribut ist `bgcolor`.

2.3 Hyperlinks – hyperwichtig

Schriftbild schön und gut, aber was ist eine Homepage ohne Hyperlinks? Hyperlinks sind die Verbindungen zwischen zwei Seiten im WWW. Es gibt zwei verschiedene Arten von Hyperlinks: interne Hyperlinks und externe Hyperlinks.

2.3.1 Interne Hyperlinks

Wenn Sie eine Homepage haben, so haben Sie automatisch eine Startseite, auch Home genannt. Auf dieser Startseite haben Sie in den meisten Fällen eine Linktafel, das heißt eine Auflistung von Themen, die Sie auf Ihrer Homepage anbieten. Diese Links sind interne Links, denn sie verweisen auf Dokumente, die zu Ihrer Homepage gehören.

Mit den Hyperlinks führen wir ein neues Tag ein: `<A>`. Das A steht hierbei für »anchor«, Englisch für Anker. Damit ist das Ziel des Hyperlinks gemeint.

Damit der Browser weiß, welchen URL[1] er anspringen soll, benötigen wir ein neues Attribut: `href`. Ihm weisen Sie den URL zu. Zum Beispiel:

```
<A href="test2.htm">
```

Das Endtag werden Sie sicher kennen. Natürlich `</A>`.

Und was steht nun zwischen Start- und Endtag? Der Text, der als Hyperlink angezeigt werden soll:

```
<A href="test2.htm">Ich bin ein Link</A>
```

Dazu gleich mal ein Beispiel:


```
<HTML>
   <HEAD>
      <TITLE>Der erste Hyperlink</TITLE>
   </HEAD>

   <BODY>
      Klicken Sie <A href="IntLink2.htm">hier</A>, um zur nächsten Seite zu
gelangen.
   </BODY>
</HTML>
```

Speichern Sie diesen Code unter *IntLink1.htm* ab. Um einen Link zu testen sind zwei Dateien notwendig. Also brauchen wir noch eine Datei:

1. URL ist die Abkürzung für *Uniform Resource Locator* und heißt ganz einfach »die Adresse« oder »der Ort« einer Datei im Web.

```
<HTML>
   <HEAD>
      <TITLE>Der erste Hyperlink</TITLE>
   </HEAD>

   <BODY>
     Klicken Sie <A href="IntLink1.htm">hier</A>, um zur Ausgangsseite zurück-
zukehren.
   </BODY>
</HTML>
```

Diesen Code speichern Sie unter *IntLink2.htm*. Und so sieht's aus:

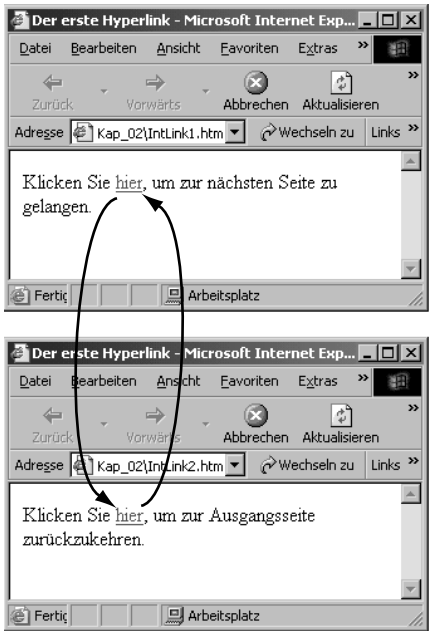

Abb. 2.13: Klickt man auf einen Hyperlink, so gelangt man zum angegebenen URL.

2.3.2 Externe Hyperlinks

Das Gegenstück zu den internen sind die externen Hyperlinks. Externe Hyperlinks verweisen auf Seiten, die zu einem anderen Web gehören, z.B. zur Homepage von Markt+Technik.

Die Programmierung ist denkbar einfach. Hier wird einfach der URL angegeben, wie Sie ihn beim Surfen im Internet auch eintippen. In diesem Falle wäre das *http://www.mut.de*. Ein kleines Beispiel dazu:

```
<HTML>
   <HEAD>
      <TITLE>Ein externer Hyperlink</TITLE>
   </HEAD>

   <BODY>
      Meine Lieblingsseite ist die <A href="http://www.mut.de">Homepage von
Markt und Technik</A>.
   </BODY>
</HTML>
```

Um dieses Beispiel testen zu können, benötigen Sie einen Internetanschluss. Da sich an dem Erscheinungsbild gegenüber einem internen Link nichts ändert, ist ein Screenshot wohl nicht notwendig. Sie finden das Beispiel auf der Buch-CD unter dem Namen *ExtLink.htm*.

2.3.3 Verknüpfungen innerhalb eines Dokuments

Bisher haben wir immer auf andere HTML-Dokumente verwiesen. Natürlich ist es auch möglich, innerhalb eines Dokuments mit Hyperlinks eine Verknüpfung herzustellen.

Dazu müssen Sie an der Stelle des HTML-Dokuments, die Sie über einen Hyperlink ansteuern wollen, einen Anker setzen. Das geschieht mit dem Attribut `name`.

Mit dem Attribut `href` wird an einer anderen Stelle im Dokument auf diesen Anker verwiesen. Dazu wird einfach der für den Anker definierte Name (der Wert von `name`) mit vorangestelltem # eingegeben.

Ein Beispiel soll das Ganze verdeutlichen:

```
<HTML>
   <HEAD>
      <TITLE>Eine Verknüpfung innerhalb eines Dokuments</TITLE>
   </HEAD>

   <BODY>
      Dies ist die Stelle, an der der <A name="Ziel">Anker</A> definiert wird.
Es folgt ein endloser Text, der sich ziemlich oft wiederholt. Es folgt ein end-
loser Text, der sich ziemlich oft wiederholt. Es folgt ein endloser Text, der
sich ziemlich oft wiederholt. Es folgt ein endloser Text, der sich ziemlich oft
wiederholt. Es folgt ein endloser Text, der sich ziemlich oft wiederholt. Es
folgt ein endloser Text, der sich ziemlich oft wiederholt. Es folgt ein endlo-
ser Text, der sich ziemlich oft wiederholt. Es folgt ein endloser Text, der
sich ziemlich oft wiederholt. Es folgt ein endloser Text, der sich ziemlich oft
wiederholt. Es folgt ein endloser Text, der sich ziemlich oft wiederholt. Es
```

```
folgt ein endloser Text, der sich ziemlich oft wiederholt. Es folgt ein endlo-
ser Text, der sich ziemlich oft wiederholt. Es folgt ein endloser Text, der
sich ziemlich oft wiederholt. Es folgt ein endloser Text, der sich ziemlich oft
wiederholt. Es folgt ein endloser Text, der sich ziemlich oft wiederholt. Es
folgt ein endloser Text, der sich ziemlich oft wiederholt. Es folgt ein endlo-
ser Text, der sich ziemlich oft wiederholt. Es folgt ein endloser Text, der
sich ziemlich oft wiederholt. Es folgt ein endloser Text, der sich ziemlich oft
wiederholt. Es folgt ein endloser Text, der sich ziemlich oft wiederholt. An
dieser Stelle kann an den <A href="#Ziel">Anfang</A> gesprungen werden.
   </BODY>
</HTML>
```

Verkleinern Sie das Browserfenster so, dass Sie gezwungen sind, zum Link zu scrollen. Nur so lässt sich das Beispiel testen.

Speichern Sie den Code unter *Link.htm*. Das Ergebnis:

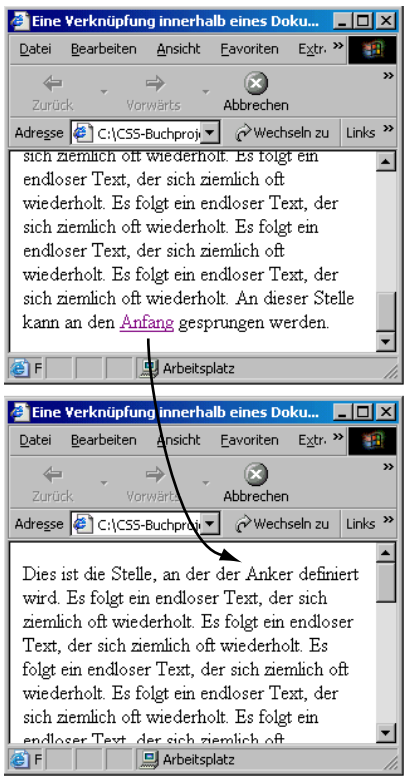

Abb. 2.14: Eine Verknüpfung innerhalb einer Seite.

Sie können auch auf einen Anker in einem anderen HTML-Dokument verweisen. Das würde dann folgendermaßen aussehen:

```
<A href="datei.htm#Ziel">
```

2.3.4 E-Mail für dich

Last but not least ist es möglich, einen Link zu einer E-Mail-Adresse einzurichten. Wenn auf diesen Link geklickt wird, öffnet sich das Standard-E-Mail-Programm, zum Beispiel Outlook Express. Dem Attribut `href` wird dabei die E-Mail-Adresse zugewiesen, mit vorangestelltem `mailto:` – beispielsweise:

```
<A href="mailto:daniel.jokat@mut.de">E-Mail an den Autor</A>
```

Man kann sogar den Betreff schon festlegen, sowie den Empfänger der Kopie. Selbst einen Text kann man schon vordefinieren. Nach der E-Mail-Adresse folgt das Zeichen »?«, dann die Anweisung für Betreff, Kopie-Empfänger oder Text. Die einzelnen Kriterien können durch das Zeichen »&« verbunden werden, um sie alle zu verwenden. Sie finden verschiedene Beispiele in der folgenden Tabelle.

Tabelle 2.5 E-Mail-Links

Beispiel	Zweck
`<A href="mailto:me@prov.de?subject=Test-Mail">`	E-Mail an `me@prov.de` mit Betreff »Test-Mail«
`<A href="mailto:me@prov.de?cc=you@prov.de">`	E-Mail an `me@prov.de` mit Kopie-Empfänger `you@prov.de`.
`<A href="mailto:me@prov.de?body=Ein Text">`	E-Mail an `me@prov.de` mit vordefiniertem Text »Ein Text«.
`<A href="mailto:me@prov.de?subject=Test-Mail&cc=you@prov.de">`	E-Mail an `me@prov.de` mit Betreff »Test-Mail« und Kopie-Empfänger `you@prov.de`.
`<A href="mailto:me@prov.de?subject=Test-Mail &body=Ein Text">`	E-Mail an `me@prov.de` mit Betreff »Test-Mail« und vordefiniertem Text »Ein Text«.

Wie in den beiden letzten Zeilen der Tabelle zu sehen, können mehrere Parameter kombiniert werden. Nach dem gleichen Muster können Sie auch alle drei Parameter (Betreff, Kopie-Empfänger und Text) zusammen angeben.


```
<HTML>
   <HEAD>
      <TITLE>E-Mail an den Autor</TITLE>
   </HEAD>

   <BODY>
      Schreiben Sie dem Autor des Buches doch eine <A
href="mailto:daniel.jokat@mut.de?subject=Meine Meinung&body=Ich finde das Buch
[bitte Meinung eintragen]">E-Mail über Ihre Meinung</A>.
   </BODY>
</HTML>
```

Speichern Sie den Code unter *E-Mail.htm*. Wenn Sie den Link anklicken und Sie Outlook als Standard-E-Mail-Client verwenden, sieht das Ergebnis so aus:

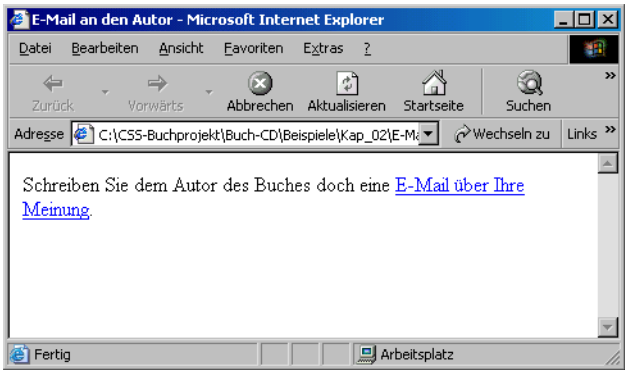

Abb. 2.15: So sieht's aus, wenn man einen Betreff und einen Text vorgibt

2.3.5 Zusammenfassung

Über Hyperlinks haben Sie in diesem Abschnitt Folgendes gelernt:

- ✘ Was ein interner Hyperlink ist.
- ✘ Was ein externer Hyperlink ist.
- ✘ Worin der Unterschied zwischen einem internen und einem externen Link besteht.
- ✘ Dass es möglich ist, auf eine E-Mail-Adresse zu verweisen.
- ✘ Dass man für eine E-Mail den Betreff, den Kopie-Empfänger und einen Text vorgeben kann.

2.4 Multimedia in HTML

Die Erwartungen der Surfer sind inzwischen recht hoch – und mit multimedialen Inhalten können Sie diesen Ansprüchen gerecht werden.

Sie sollten es jedoch mit Multimedia nicht übertreiben! Zwar wird sich der Betrachter über etwas Abwechslung freuen, andererseits wird er verärgert sein, wenn wegen zuviel Multimedia die Ladezeiten zu hoch sind.

Auch hier kommen sich *Netscape* und *Microsoft* wieder in die Haare. Für das gleiche Resultat erwarten nämlich beide Browser verschiedene Tags. Das Problem lässt sich mit JavaScript zwar lösen, schöner wäre es jedoch, wenn man erst gar kein Problem hätte.

Doch beginnen wir zunächst mit einem ganz unproblematischen Element, den Bildern.

2.4.1 Ein Bild sagt mehr als tausend Worte

Am Anfang des Abschnitts wurde der sparsame Einsatz von Multimedia angeraten. Bilder fallen hier jedoch weniger ins Gewicht. Wo Sie auf Videos verzichten, können Sie ruhig ein Bild verwenden.

Wieder wird ein neues Tag fällig: `<IMG>`. Und was brauchen wir noch? Wenn Sie diese Frage jetzt mit »Endtag« beantwortet haben, so liegen Sie leider falsch, denn für das `<IMG>`.-Tag gibt es kein Endtag.

Vielmehr `<IMG>`-Attribute: Um ein Bild anzeigen zu können, müssen Sie dem Browser mitteilen, wo sich das Bild befindet. Das Attribut ist allerdings nicht `href`, sondern `src` (engl. für »source« = Quelle).


```
<HTML>
   <HEAD>
      <TITLE>Ein Bild</TITLE>
   </HEAD>

   <BODY>
      Das Einbinden von Bildern geht über das IMG-Tag<IMG src="IMG.jpg">
   </BODY>
</HTML>
```

Speichern Sie den Code unter *IMG.htm*.

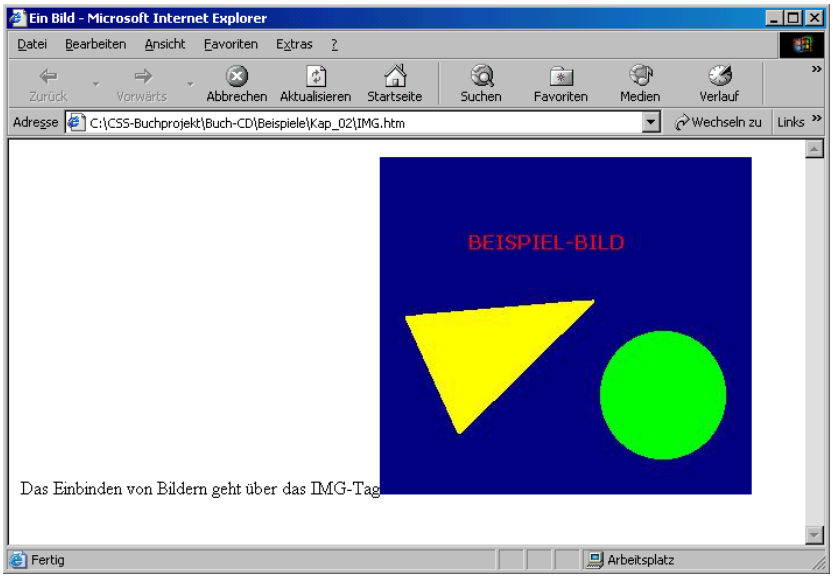

Abb. 2.16: Nun können Sie auch Bilder einbinden.

In sehr alten Browsern oder reinen Text-Browsern werden Bilder nicht angezeigt. Das W3C hat daher ein Attribut vorgesehen, mit dem ein Text anstelle eines Bildes angezeigt wird. Es heißt `alt`. Wenn das Bild im *Internet Explorer* angezeigt wird, so erscheint dieser Text als Tooltip, wenn der Mauszeiger auf dem Bild länger verharrt. Erweitern Sie die 7. Zeile des vorherigen Beispiels wie folgt:

```
Das Einbinden von Bildern geht über das IMG-Tag <IMG src="IMG.jpg" alt="Hier ist
ein Bild">
```

Speichern Sie den abgeänderten Code ab, abermals unter *IMG.htm*. Auf den ersten Blick sieht alles so aus wie vorher, aber lassen Sie den Mauszeiger mal ein paar Sekunden auf dem Bild verharren.

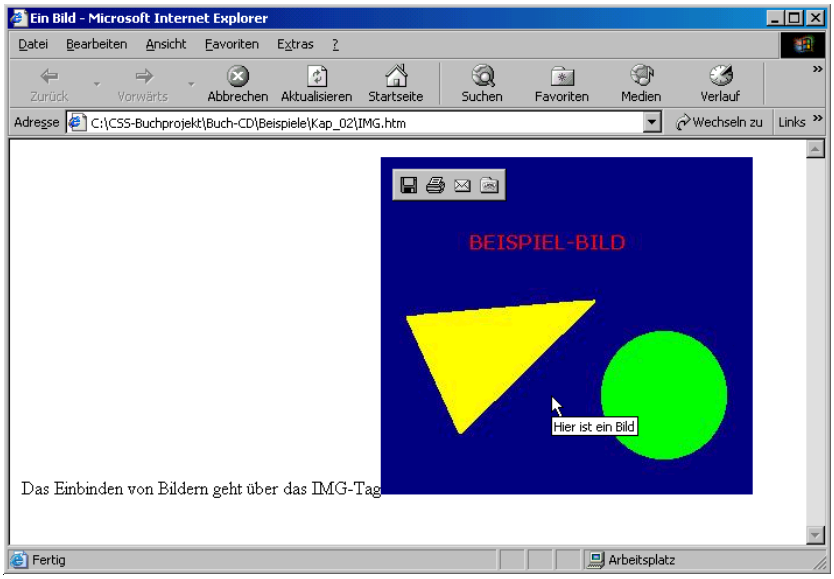

Abb. 2.17: Ein Bild mit Tooltip.

Achtung beim Grafikformat

Wenn Sie eine Grafik in ein HTML-Dokument einfügen wollen, so sollten Sie ausschließlich zwei Grafikformate verwenden: GIF und JPG. Diese sind hinsichtlich ihrer Dateigröße und ihrer Qualität unübertroffen, und Browser können sie sehr schnell vom Server herunterladen, um Sie anzuzeigen. Die Verwendung von zum Beispiel BMP-Grafiken könnte zu Problemen führen, da

a) sie von älteren Browsern unter Umständen nicht angezeigt werden und

b) lange Ladezeiten Ihrer Seite im Internet entstehen können (vor allem, wenn sie mit einem analogen Anschluss abgerufen werden).

Denken Sie also daran, möglichst nur die beiden Formate GIF und JPG zu verwenden.[1]

Darüber hinaus haben Sie die Möglichkeit, das Bild selbst sowie den Text neben dem Bild auszurichten.

Das Bild können Sie nur rechts und links ausrichten. Dies geschieht mit dem Attribut `align`, dessen Bedeutung Sie aus Abschnitt 2.2 bereits kennen. Die möglichen Werte hierfür sind `left` und `right`.

1. Zukunftsweisend ist auch das PNG-Format, welches eine hohe Auflösung mit Transparenz verbinden soll. Das entspräche JPG und GIF in einem.

Den Textfluss um das Bild richten Sie ebenfalls mit `align` aus. Hierfür gelten die Werte `top`, `middle` und `bottom`.

Ebenfalls veränderbar sind die Höhe und Breite des Bildes. Die Attribute `height` (Höhe) und `width` (Breite) geben Ihnen die Möglichkeit, die Bildattribute über Pixel- und Prozentangaben zu verändern.

Ein Beispiel, das diese Attribute im Einsatz zeigt:


```
<HTML>
   <HEAD>
      <TITLE>Texte am Bild, und Bilder an sich ausrichten</TITLE>
   </HEAD>

   <BODY>
      <IMG src="IMG.jpg" alt="Hier ist ein Bild" align="left">
      Wenn das Bild ausgerichtet wurde, umfließt der Text das Bild.
  <P><IMG src="IMG.jpg" alt="Hier ist noch ein Bild" align="middle">Hier wurde
nur der Text mittig ausgerichtet.</P>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *IMGALIGN.htm* und öffnen Sie die Datei im Browser:

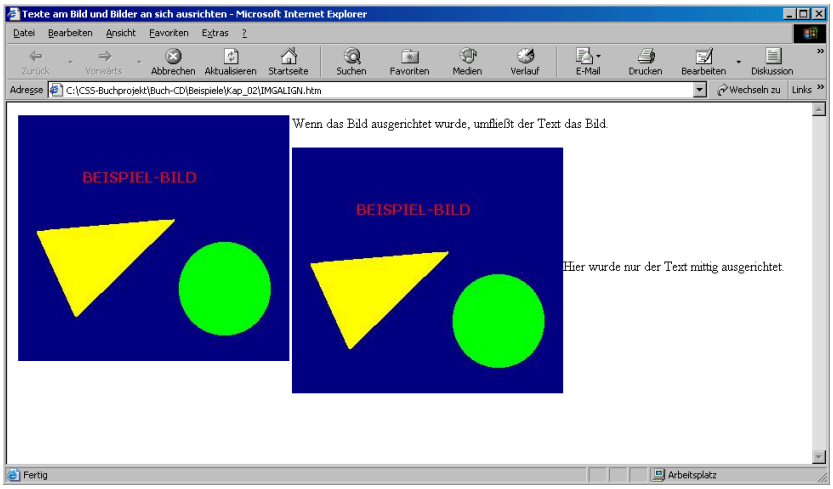

Abb. 2.18: Die Ausrichtung von Bild und Text am Bild.

Achten Sie auf die Ausrichtung des Bildes und den Textfluss um das Bild in Abbildung 2.18.

Das rechte Bild im Dokument wird nicht unter dem ersten Bild angezeigt, sondern daneben. Das kommt daher, dass durch die Ausrichtung des ersten Bildes durch `align="left"` der weitere Text um das Bild herumfließt. Das gilt auch für andere Elemente, wie eben Bilder. Ändern Sie die Ausrichtungseigenschaft für das erste Bild einmal ab und streichen Sie das `align`-Attribut. Damit wird das zweite Bild unter dem ersten angezeigt.

Sie können Bilder übrigens auch als Link definieren. Setzen Sie dazu einfach das `<IMG>`-Tag zwischen die schon bekannten `<A>`-Tags. Wenn Sie der Rahmen um das Bild stört, ergänzen Sie das `<IMG>`-Tag um das Attribut `border` und setzen Sie dieses auf `0`.

Sie können auch ein Hintergrundbild für das HTML-Dokument angeben. Geben Sie im `<BODY>`-Tag das Attribut `background` an und übergeben Sie diesem den Pfad der Bilddatei. Wenn Sie bewirken wollen, dass der Hintergrund nicht mitgescrollt wird, setzen Sie zusätzlich das Attribut `bgproperties` auf `fixed`.

2.4.2 Videos einbinden

Bilder findet man im Internet »an jeder Ecke«. Was dagegen noch nicht so häufig eingesetzt wird, sind Videos.

Zum Einbetten von Videos wird ein neues Tag benötigt: `<EMBED>`. Über das Attribut `src` geben Sie an, wo sich das Video befindet.


```
<HTML>
   <HEAD>
      <TITLE>Ein Video...</TITLE>
   </HEAD>

   <BODY>
      <CENTER>
      <H2>Ein Video:</H2>
      <EMBED src="video.avi">
      </CENTER>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *video.htm* ab. Die Videodatei *video.avi* finden Sie auf der Buch-CD unter *Beispiele\Kap_02*.

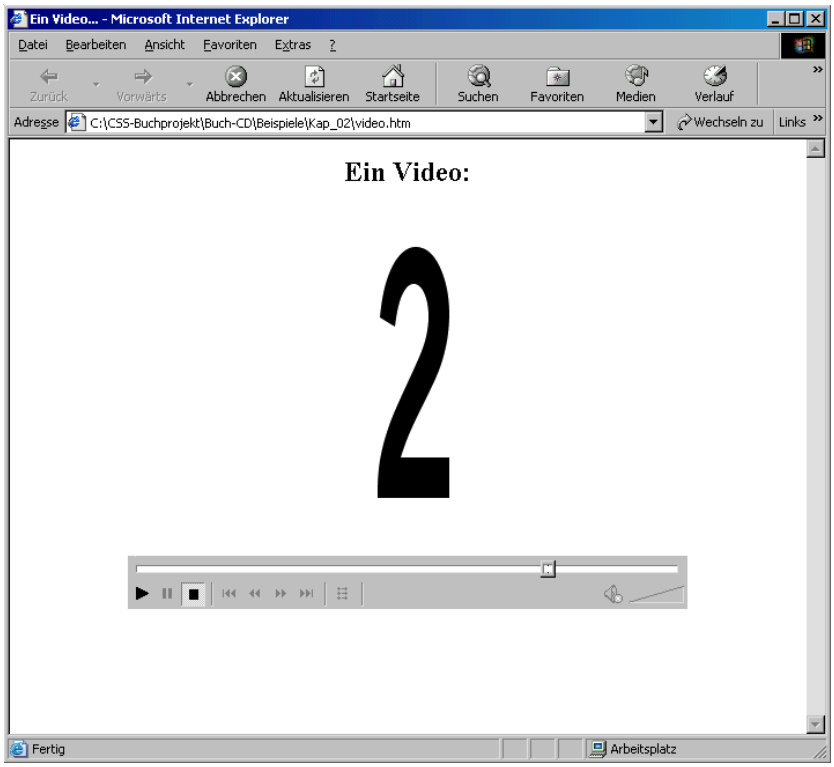

Abb. 2.19: Hier wird ein Video abgespielt.

Man kann sehen, dass eine Bedienleiste erscheint. Im IE wird das Video außerdem schon beim Öffnen der Seite gestartet. Beim NN 4.x müssen Sie auf das Video doppelklicken, um es zu starten. Ändern können Sie das über das Attribut `autoplay`. Hier regulieren die Werte `true` (Autostart an) und `false` (Autostart aus), wie sich das Video verhalten soll.

Es folgt das Tuning: Sie können festlegen, ob das Video wiederholt werden soll. Dazu benötigen Sie das Attribut `loop`. Diesem weisen Sie entweder `true` oder `false` zu. Bei `true` wird das Video unendlich oft abgespielt, bei `false` nur einmal. Der Standardwert ist `false`.

Für den *Internet Explorer* gibt es die Möglichkeit, die Anzahl der Wiederholungen festzulegen. Dazu benutzt man das `<IMG>`-Tag. Über das Attribut `dynsrc` wird der Ort des Videos angegeben. Dem Attribut `loop` übergeben Sie einfach die Anzahl der Wiederholungen. Für eine Endlos-Wiederholung wird der Wert –1 eingesetzt.

Beispiel: `<IMG dynsrc="video.htm" loop=-1>`

Schließlich können Sie noch das Attribut `controller` einfügen. Damit ermöglichen Sie dem Betrachter das Steuern des Videos, sprich Starten, Stoppen, Pause, Rück- und Vorlauf. Hierbei geben Sie einfach `true` für die Anzeige und `false` für die Nichtanzeige der Bedienungselemente an.

Wie bei einem Bild, können Sie auch das Video in Höhe und Breite über die entsprechenden Attribute `height` und `width` verändern. Diese Angabe der Werte wird in Pixel oder in Prozent gemacht.

Auch für Videos ist es möglich, Links zu definieren. Sie brauchen lediglich das `href`-Attribut einzufügen und diesem den Ort der Zieldatei zuweisen.

2.4.3 Sounds mit HTML

Nach den Bildern und den Videos sollen nun auch die akustischen Bedürfnisse befriedigt werden. In den letzten Jahren hat es sich immer stärker durchgesetzt, Websites mit einem Sound zu hinterlegen (zum Beispiel einem Song der eigenen Band). Doch mit den Sounds kommen auch die Probleme der Web-Besucher, die einen sehr alten PC für das Internet nutzen.

Sie sollten nur dann einen Sound anbieten, wenn Sie Ihr Angebot ausschließlich auf modernere PCs abstimmen.

Welche Möglichkeiten bieten sich Ihnen, Sounds auf Ihrer Webseite anzubieten? Sie können

1. einen Hintergrundsound oder
2. einen manuell zu startenden Sound

anbieten.

Hierzu verwenden Sie das schon bekannte `<EMBED>`-Tag. Als `src` geben Sie den abzuspielenden Sound an, zum Beispiel:

```
<EMBED src="musik.wav" loop="true">
```

In diesem Falle wird die Datei *musik.wav* beim Öffnen des HTML-Dokuments abgespielt, endlos lange wiederholt und die Bedienelemente werden angezeigt – zumindest beim *Internet Explorer*. Beim *Netscape* ist die Leiste mit den Bedienelementen nicht breit genug und der Sound wird nicht automatisch abgespielt. Was also oben im IE für Autostart und Bedienleiste reicht, muss für NN so aussehen:

```
<EMBED src="musik.wav" loop="true" autostart="true" width=145>
```

`width=145` ist wegen der Bedienleiste notwendig, die fälschlicherweise nicht korrekt angezeigt wird (NN 4.x).

Auch hier kann wieder Vieles geändert werden. Aus dem obigen Code-Fragmenten geht schon die Verwendung von `loop` hervor. Die gültigen Werte sind `true` und `false`.

Außerdem können Sie bestimmen, ob der Sound automatisch oder manuell gestartet werden soll. Für automatisches Starten geben Sie `autostart="true"` an. Für manuelles Starten gilt `autostart="false"`.

In diesem Zusammenhang muss an den menschlichen Faktor erinnert werden: Sollten Sie angegeben haben, dass der Sound manuell gestartet werden soll, dürfen Sie keinesfalls die Bedienleiste ausblenden. Dem bereits bekannten Attribut `controller` darf dann unter keinen Umständen `false` zugewiesen. Der Standardwert ist allerdings `true`. `controller="false"` und `autostart="true"` ergeben die Kombination für einen Hintergrundsound.

Der *Internet Explorer* ermöglicht es über das Tag `<BGSOUND>`, den Sound so oft abspielen zu lassen, wie Sie wollen. Dabei handelt es sich dann allerdings um reine Hintergrundmusik. Dem Attribut `src` übergeben Sie den Ort der Musikdatei und `loop` einfach die Anzahl der Wiederholungen. Für unendliche Wiederholung wird der Wert `infinite` übergeben.

Auch `width` und `height` können wieder angegeben werden. Da der NN die Bedienleiste nicht breit genug anzeigt, sollte für diesen `width=145` angegeben werden. Doch dann wird auch beim IE nicht mehr die korrekte Breite angezeigt. Sie sehen, es ist schlimm…

Die bisher vorgestellten Angaben waren Ihnen schon von den Videos her bekannt. Zusätzlich gibt es für Sounds noch folgende Attribute:

- `volume` (Lautstärke, Werte von 1-10, wobei 10 am lautesten ist)
- `hidden` (Ersatz für `controller=false`)

Aufgrund möglicher Probleme, die beim Abspielen des Sounds auf Systemen mit unzureichend ausgestatteter Soundkarte auftreten können, sollten Sie nur

folgende Soundformate verwenden, die sich unter anderem durch unterschiedliche Tonqualitäten voneinander unterscheiden:

Tabelle 2.6 Soundformate

Format	Tonqualität	Anforderung
Quicktime	div. Auflösungen	Quicktime 2.5 und höher
MIDI	feste Klänge in der Soundkarte oder aus Quicktime	MIDI-fähige Soundkarte oder Quicktime 2.5 und höher
WAV	div. Auflösungen	Soundkarte MAC: Brians Soundtool oder ab *Netscape Navigator* 3.0 bzw. *Internet Explorer* 3.0
AU	11 kHz, 8 Bit, µ.law	
AIFF	div. Auflösungen	MAC/OS und Windows
SND	div. Auflösungen	nur Windows und externer Player
MP3	div. Auflösungen	MP3-Player

Sie sollten auf einer Webseite niemals CD-Qualität für einen Sound verwenden, da eine viel zu großer Datenmenge übertragen werden müsste. Davon ist vor allem das WAV-Format betroffen. Denken Sie bei der Verwendung von WAV-Dateien daher daran, die Qualität des Sounds herunterzusetzen (Sound-Programme bietet üblicherweise CD-Qualität, Radio-Qualität und Telefon-Qualität an). Wollen Sie sich aber nicht unnötig Arbeit machen, so verwenden Sie das Format MP3, welches etwa zehn Mal so wenig Speicherplatz für die nahezu gleiche Qualität benötigt.

Ein Beispiel:


```
<HTML>
   <HEAD>
      <TITLE>Ein Sound</TITLE>
   </HEAD>

   <BODY>
      <CENTER>
      <H2>Hier hören Sie einen Sound</H2>
      <EMBED src="laugh.wav">
      </CENTER>
   </BODY>
</HTML>
```

Speichern Sie den Code als *sound.htm*.

Abb. 2.20: Gerade wird ein Sound abgespielt.

2.4.4 Zusammenfassung

In der Welt der multimedialen Elemente in HTML kennen Sie sich inzwischen schon ziemlich gut aus. Eine Auflistung der Elemente, die Ihnen in diesem Abschnitt vorgestellt wurden, soll das Gelernte noch einmal in Erinnerung rufen. Sie können:

- Bilder einbinden,
- Bilder im HTML-Dokument ausrichten,
- Den Text an einem Bild ausrichten (den Textfluss bestimmen),
- Videos einbinden und einstellen (Autostart, Wiederholung, Kontrollleiste),
- Sounds einbinden und
- Hintergrundsound einbinden.

2.5 Besondere Tags

Nach einigen längeren Abschnitten nun mal wieder ein eher kurzes Unterkapitel. Es ist der Übersichtlichkeit eines HTML-Dokuments gewidmet. So können Sie HTML-Dokumente mit Trennlinien optisch etwas verschönern oder Kommentare im HTML-Code verwenden, die Ihnen und auch anderen Web-Programmierern helfen, den Code zu verstehen.

2.5.1 Horizontale Trennlinien

Es ist möglich, durch horizontale Trennlinien den Text auf Ihrer Webseite zu untergliedern. Dafür gibt es ein ganz simples neues Tag: <HR>.

Wenn Sie dieses Tag genau so wie in unserem Beispiel in den Code einbinden, wird eine Trennlinie vom linken bis zum rechten Rand des HTML-Dokuments gezogen. Dabei ist die Linie von Zeilenumbrüchen umgeben, sodass Sie das
-Tag nicht extra einsetzen brauchen, um die Linie vom restlichen Text zu trennen.


```
<HTML>
   <HEAD>
      <TITLE>Horizontale Trennlinien</TITLE>
   </HEAD>

   <BODY>
      Ein Text, der irgendwas beschreibt.
      <HR>
     Die horizontale Trennlinie schneidet das Textbild. Das BR-Tag muss dabei
nicht eingesetzt werden.
   </BODY>
</HTML>
```

Speichern Sie den Code unter *HR.htm* ab und öffnen Sie die Datei.

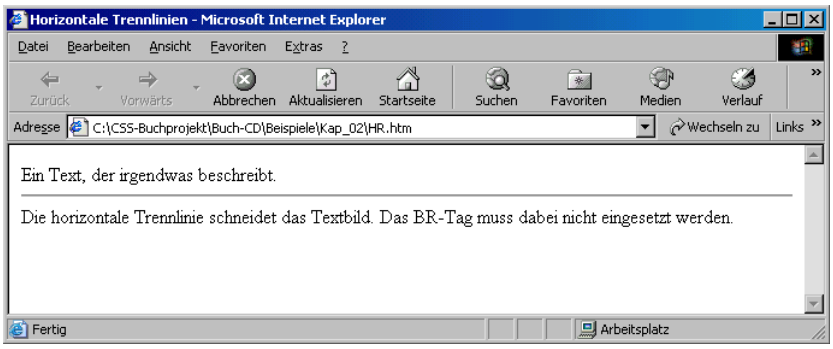

Abb. 2.21: So sieht die horizontale Trennlinie im HTML-Dokument aus.

Auch die Trennlinie können Sie konfigurieren:

- Die Standard-Trennlinie ist dreidimensional. Das können Sie ändern, indem Sie das Attribut noshade verwenden. Es ist kein Wert dafür erforderlich.

- Über `size` und `width` kann nun wieder Höhe bzw. Breite angegeben werden. Diese Angabe erfolgt in Pixel oder in Prozent.
- Mit `align` ist die Ausrichtung möglich. Hierbei gelten die bekannten Werte.
- Mit `color` kann eine Farbe angegeben werden. Hierbei wird das RGB-Farbsystem verwendet.

Ein abschließendes Beispiel zeigt den Einsatz dieser Attribute:


```
<HTML>
   <HEAD>
      <TITLE>Horizontale Trennlinien</TITLE>
   </HEAD>

   <BODY>
     <H1>Dieses HTML-Dokument zeigt eine horizontale Trennlinie mit allen Einstellmöglichkeiten</H1>
      <HR noshade size=20 width=50% align="center" color=#8000FF>
      Wieder normaler Text.
   </BODY>
</HTML>
```

Speichern Sie den Code unter *HR2.htm* ab. Das Resultat:

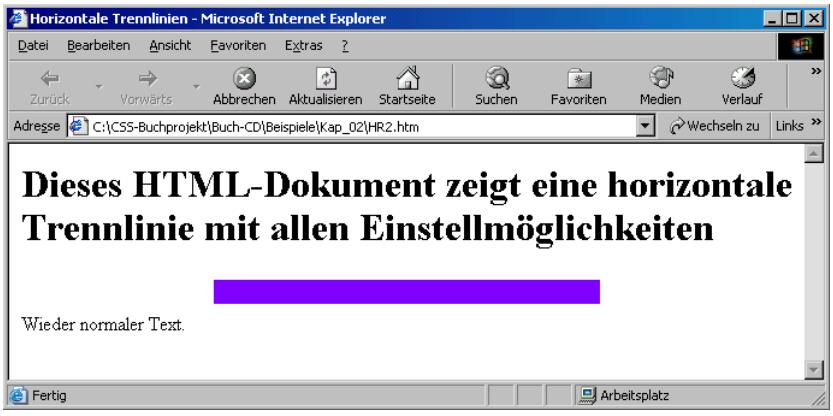

Abb. 2.22: Es wurden alle Einstellungen an der Trennlinie verändert.

2.5.2 Kommentare

Kommentare können für die Beschreibung Ihres Codes sehr wichtig sein. Der HTML-Code ist insbesondere bei großen HTML-Dokumenten ziemlich lang und sieht sehr unübersichtlich aus. Wenn Sie Kommentare einsetzen, erken-

nen Sie später schneller wieder, was Sie vor vielleicht vielen Monaten programmiert haben.

Kommentare werden zwischen die Klammern `<!--` und `-->` geschlossen. Es gibt aber auch das Tag `<COMMENT>`. Der zwischen diesem Starttag und dem zugehörigen Endtag gestellte Text ist ein Kommentar.

```
<!-- Ich bin ein HTML-Kommentar -->

<COMMENT>Ich auch</COMMENT>
```

Kommentare werden im Browser natürlich nicht angezeigt.

Es ist wirklich so einfach, dass kein Beispiel hierfür nötig ist.

2.5.3 Zusammenfassung

Nach diesem Abschnitt können Sie

- ✘ horizontale Trennlinien verwenden und diese konfigurieren und
- ✘ Kommentare erstellen.

2.6 Auflistungen

Angenommen, Sie wollen eine Liste über Ihre guten Eigenschaften auf eine Webseite bringen. Sie könnten natürlich mit einer Strichaufzählung arbeiten. Eleganter ist die Arbeit jedoch mit den Listenformatierungen in HTML.

Es gibt in HTML zwei verschiedene Arten von Auflistungen mit jeweils eigenen Tags:

1. die geordnete Liste
2. die ungeordnete Liste

2.6.1 Geordnete Listen

Geordnete Listen. Was sagt uns das nun wieder?

Eine geordnete Liste ist eine Liste, die fortlaufend durchnummeriert ist. Dabei kann die Art der Nummerierung definiert werden. Standard ist eine Nummerierung durch arabische Zahlen. Das Tag, das Sie dafür benötigen, lautet `<OL>`. Zwischen diesem Starttag und seinem Endtag werden die Listenpunkte gesetzt. Diese sind durch `<LI>` zu definieren. Hier ist ein Endtag nur erforderlich, wenn die Auflistung kurz, zum Beispiel für einen beschreibenden Text, unterbrochen werden soll. Durch das Attribut `type` kann die Art der Nummerierung

verändert werden (in Bezug auf <OL>). Entnehmen Sie der folgenden Tabelle die möglichen Werte:

Tabelle 2.7 Mögliche Nummerierungen für Listen

Wert für type	Darstellung
1	arabische Zahlen
A	alphabetische Nummerierung in Großbuchstaben
a	alphabetische Nummerierung in Kleinbuchstaben
I	große römische Nummerierung
i	kleine römische Nummerierung

Auch Listen können mit `align` ausgerichtet werden – mit Hilfe der bekannten Werte `left`, `right`, `center` und `justify`.

Zu guter Letzt können Sie dem Attribut `start` die Startzahl übergeben. Ab dieser Zahl beginnt die Nummerierung.

Das folgende Beispiel verdeutlicht den Einsatz von geordneten Listen:


```
<HTML>
   <HEAD>
      <TITLE>Geordnete Listen</TITLE>
   </HEAD>

   <BODY>
      <H2>Einsatz von geordneten Listen</H2>
      <OL type="I">
         <LI>Dies ist
         <LI>eine Auflistung
         <LI>mit römischer
         <LI>Nummerierung
      </OL>
      <OL type="1" start=5>
         <LI>Dies ist
         <LI>eine Auflistung
         <LI>mit arabischer
         <LI>Nummerierung
      </OL>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *OL.htm* und öffnen Sie das Dokument im Browser:

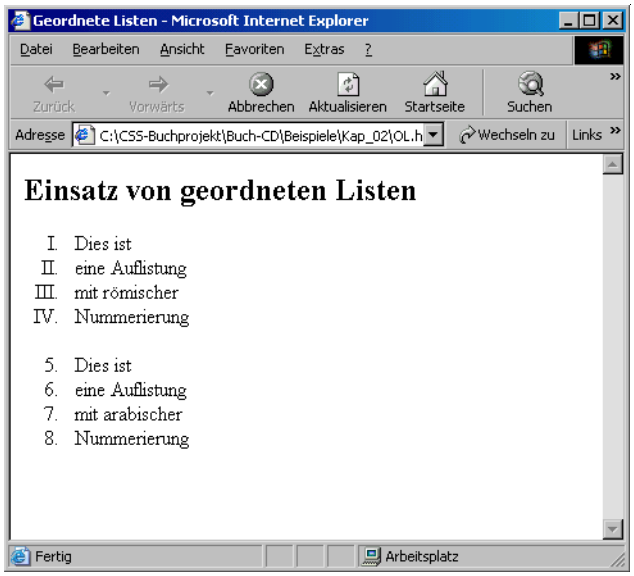

Abb. 2.23: Zwei verschiedene Typen der geordneten Liste.

2.6.2 Ungeordnete Listen

Bei der ungeordneten Liste werden die Elemente nicht nummeriert, sondern mit Aufzählungspunkten versehen. Ansonsten ist sie ähnlich zu handhaben wie die geordnete Liste. Das Aussehen des Aufzählungspunkts können Sie selbst festlegen, eine eigene Grafik kann man aber leider nicht einbinden.

Das Tag für ungeordnete Listen ist `<UL>`. Das `start`-Attribut existiert hier nicht, da man bei Aufzählungspunkten identische Objekte (die so genannten Bullets) vor sich hat.

Bei einer ungeordneten Liste können Sie mit `type` zwischen drei verschiedenen Darstellungen für die Aufzählungspunkte wählen. Für das Attribut `type` gibt es folgende Werte:

Wert für type	**Darstellung**
`disc`	ausgefüllter Kreis
`circle`	leerer Kreis
`square`	Quadrat

Tabelle 2.8 Aufzählungspunkte

`disc` ist der Standardwert für `type`. Wie bei den geordneten Listen überprüfen wir auch hier die Möglichkeiten an einem abschließenden Beispiel:


```
<HTML>
   <HEAD>
      <TITLE>Ungeordnete Listen</TITLE>
   </HEAD>

   <BODY>
      <H2>Einsatz von ungeordneten Listen</H2>
      <UL>
         <LI>Dies ist
         <LI>eine Auflistung
         <LI>mit ausgefüllten Kreisen.
      </UL>
      <UL type="square">
         <LI>Dies ist
         <LI>eine Auflistung
         <LI>mit Quadraten.
      </UL>
   </BODY>
</HTML>
```

Speichern Sie das Beispiel unter *UL.htm*. Das Ergebnis ist Folgendes:

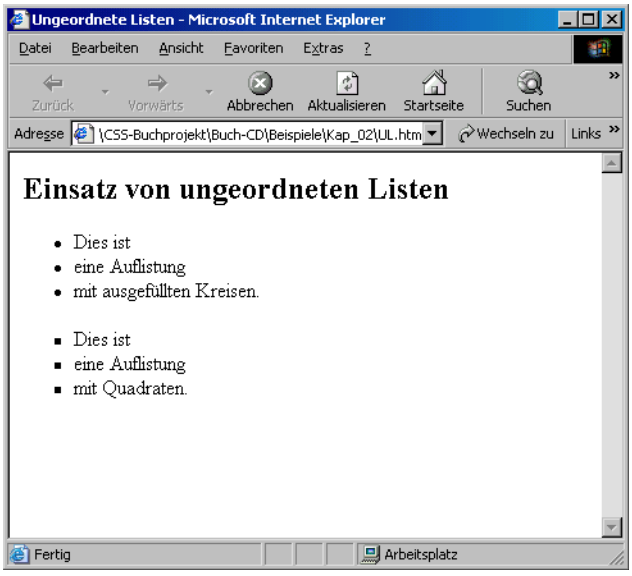

Abb. 2.24: Zwei Typen der ungeordneten Liste.

2.6.3 Listen verschachteln

Listen verschachteln – was bedeutet das? HTML bietet die Möglichkeit, innerhalb einer Liste weitere, untergeordnete Listen zu definieren. Diese werden dann eingerückt. Lassen Sie uns das anhand einer Einkaufsliste demonstrieren (obwohl man die ja nicht gerade auf seiner Homepage anpreisen sollte).


```
<HTML>
   <HEAD>
      <TITLE>Listen schachteln</TITLE>
   </HEAD>

   <BODY>
      Dies ist meine Einkaufsliste vom 13. April 1924. Sie ist zwar nicht
      mehr aktuell, aber ich denke Sie können etwas damit anfangen.
      <OL type="1">
         <LI>Globus (na gut, den gab's damals noch nicht, nehmen wir den Real)
         <UL>
            <LI>Käse
            <LI>Wurst
            <LI>Gemüse
         </UL>
         <LI>Bäcker
         <UL>
            <LI>7 Brötchen
            <LI>1000g Brot
         </UL>
         <LI>Blumenladen
         <UL>
            <LI>Strauß für Marthas Grab
         </UL>
      </OL>
   </BODY>
</HTML>
```

Das Beispiel finden Sie auf der Begleit-CD unter dem Namen *OLUL.htm*.

2.6.4 Zusammenfassung

Auflistungen können sehr nützlich sein. Nach diesem Abschnitts können Sie

- ✘ geordnete Listen einbinden,
- ✘ für geordneten Listen eine Startzahl festlegen,
- ✘ den Typ für eine geordnete Liste festlegen,
- ✘ ungeordnete Listen einbinden,
- ✘ den Typ für eine ungeordnete Liste festlegen.

2.7 Achtung, Sonderzeichen!

Die Internet-Nutzer mit Uralt-Browser machen es dem Webmaster schon richtig schwer: Nicht nur, dass zum Teil Bilder nicht angezeigt werden können – nein, da ist noch mehr!

Wenn Sie die Beispiele bis hierher aufmerksam verfolgt haben, ist Ihnen bestimmt aufgefallen, dass wir deutsche Umlaute verwendet haben, die international als Sonderzeichen gelten. Und wenn Sie diese in älteren Browsern betrachten, erkennen Sie das Problem: Ältere Browser können Sonderzeichen nicht korrekt darstellen. Doch keine Panik, es gibt einen Ausweg.

Diese Lösung ist zwar nicht gerade immer komfortabel, aber Hauptsache, das Problem ist überhaupt aus der Welt zu schaffen. Für jedes Sonderzeichen gibt es eine passende Codierung. Dabei unterscheidet man zwei Arten:

1. Nummerncodes
2. Zeichen-Entitäten

In der nun folgenden Tabelle liste ich einige der wichtigsten Nummerncodes und Zeichen-Entitäten auf. Den Rest finden Sie in Anhang F.

Tabelle 2.9 Sonderzeichen in HTML

Sonderzeichen	Nummerncode	Zeichen-Entitäten
Ä	`&#196;`	`&Auml;`
Ö	`&#214;`	`&Ouml;`
Ü	`&#220;`	`&Uuml;`
ä	`&#228;`	`&auml;`
ö	`&#246;`	`&ouml;`
ü	`&#252;`	`&uuml;`
ß	`&#223;`	`&szlig;`
©	`&#169;`	`&copy;`
®	`&#174;`	`&reg;`
™	`&#153;`	
Ç	`&#199;`	`&Ccedil;`
ç	`&#231;`	`&ccedil;`
<	`&#60;`	`<`
>	`&#62;`	`>`
&	`&#38;`	`&`
"	`&#34;`	`&quot;`

Statt des Sonderzeichens geben Sie einfach eine der Codierungen an. Der HTML-Text

```
Übung macht den Meister
```

wird demnach zu

```
&Uuml;bung macht den Meister
```

oder zu

```
&#220;bung macht den Meister
```

Es ist zwar nicht schwer, doch man muss sich daran gewöhnen.

Damit Sie wirklich lernen, an die Codierungen zu denken, werden im weiteren Verlauf des Buches diese Codierungen angewendet.

2.8 Tabellenlayout

Daten kann man natürlich trist auflisten, doch leichter hat es der Betrachter, wenn man die Daten optisch etwas aufbereitet, übersichtlich präsentiert – zum Beispiel in Form einer Tabelle.

Das neue Tag, das Sie benötigen, heißt <TABLE>. Eine Tabelle besteht aus Zeilen und Spalten. Für neue Daten, die in einer Tabelle dargestellt werden sollen, müssen Sie also zunächst eine neue Zeile anlegen. Dies geschieht mit Hilfe des Tags <TR>, der Abkürzung für Table Row oder Tabellenzeile. In dieser Zeile fehlt es nun noch an Zellen.

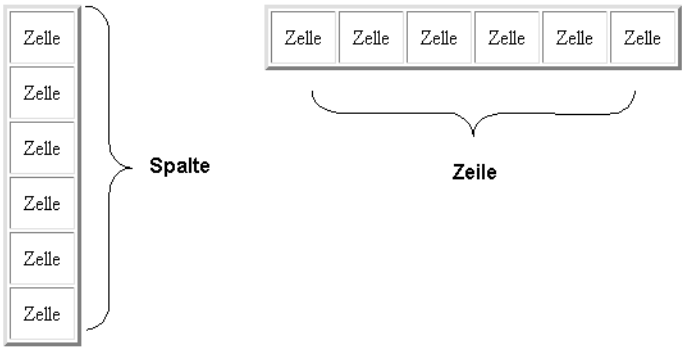

Abb. 2.25: Zellen, Zeile, Spalte.

Eine neue Zelle erstellen Sie mit <TD>. Und in diese Zelle schreiben Sie nun den gewünschten Wert.


```
<HTML>
   <HEAD>
      <TITLE>Eine Tabelle</TITLE>
   </HEAD>

   <BODY>
      <P>Dies ist eine Tabelle:</P>
      <TABLE>
         <TR>
            <TD>Wert f&uuml;r die Zelle
            <TD>Die zweite Zelle
         <TR>
            <TD>Dies ist die erste Zelle der zweiten Zeile.
            <TD>Zweite Zelle der zweiten Zeile.
      </TABLE>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *Table1.htm* und nehmen Sie das Ergebnis kritisch unter die Lupe:

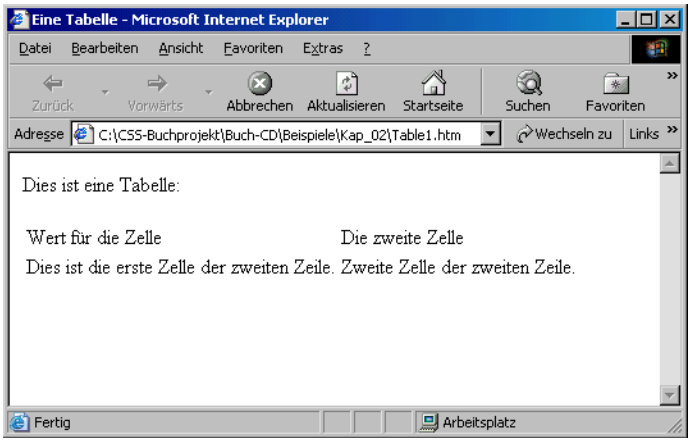

Abb. 2.26: Wo ist die Tabelle?

Eine Tabelle ohne Linien ist nicht sichtbar – versehen wir sie also mit einem Rahmen. Dazu müssen Sie das Attribut `border` angeben und diesem als Wert eine natürliche Zahl (ganze Zahlen, größer oder gleich Null) zuweisen. Ändern Sie die 8. Zeile wie folgt ab:

```
<TABLE border=1>
```

Sehen Sie sich das abgeänderte HTML-Dokument erneut an. Besser?

Natürlich können Sie die Tabelle auch weiter konfigurieren. Sie können

- ✘ den Zellenabstand,
- ✘ ein Hintergrundbild für die Tabelle,
- ✘ den Textabstand vom Rand der Zelle innen,
- ✘ Höhe und Breite der Tabelle und
- ✘ einen Tabellenkopf

festlegen. Wir gehen das nun der Reihe nach durch:

Für den Abstand der Zellen untereinander ist das Attribut `cellspacing` erforderlich. Geben Sie als Wert die gewünschte Anzahl der Pixel an.

Ein Hintergrundbild legen Sie mit `background` fest. Geben Sie hier einfach den Pfad der Bilddatei an.

Nur der *Internet Explorer* versteht dieses Attribut »richtig«. Der *Netscape* verwendet das angegebene Bild als Hintergrund für jede einzelne Zelle.

Nochmals: Verwenden Sie auf jeden Fall nur GIF- und JPG-Grafiken! Den Grund dafür können Sie in Abschnitt 2.4.1 nachlesen.

Das Hintergrundbild wird nur dann angezeigt, wenn die Tabelle mindestens eine Zelle enthält.

Der Abstand des Zellentextes zum Zellenrand lässt sich in Pixel angeben. Verwenden Sie dazu das Attribut `cellpadding`. Über `height` und `width` lassen sich Höhe und Breite in Pixel und Prozent angeben. Diese Attribute beziehen sich auf das `<TABLE>`-Tag.

Um einen Tabellenkopf zu definieren, geben Sie statt `<TD>` einfach `<TH>` an.

Sie können außerdem noch eine Zelle über mehrere Spalten strecken und eine Zelle über mehrere Zeilen. Das Attribut `colspan` dient dabei für die Streckung der Zelle über mehrere Spalten. Geben Sie einfach die Anzahl der Spalten ein, über die die betroffene Zelle gestreckt werden soll. Bei `rowspan` geben Sie die Anzahl der Zeilen an, über die die betroffene Zelle gestreckt werden soll. Diese Attribute gehören in das `<TD>`- bzw. in das `<TH>`-Tag.

Und da Probieren über Studieren geht, auch gleich ein abschließendes Beispiel, das alle diese Elemente berücksichtigt. Höhe und Breite sind dabei auf die Größe des Hintergrundbildes festgelegt, welches Sie auf der Buch-CD unter *Beispiele\Kap_02* finden:

```
<HTML>
   <HEAD>
      <TITLE>Die Tabelle mit ihren Eigenschaften</TITLE>
   </HEAD>

   <BODY>
      Dies ist eine Tabelle mit Hintergrundbild:
      <TABLE border=2 cellpadding=10 cellspacing=5 height=330 width=420
background="TABLE.jpg">
         <TR>
            <TH>Ein Tabellenkopf
            <TH>Tabellenkopf f&uuml;r die 2. Spalte
            <TD>Normaler Text
         <TR>
            <TD>Text f&uuml;r Zelle 1
            <TD>Text f&uuml;r Zelle 2
            <TD>Text f&uml;r Zelle 3
         <TR>
            <TD colspan=3>&Uuml;ber 3 Spalten gestreckte Zelle
      </TABLE>
   </BODY>
</HTML>
```

Speichern Sie den Code als *Table2.htm*. Folgendes Resultat ist zu sehen:

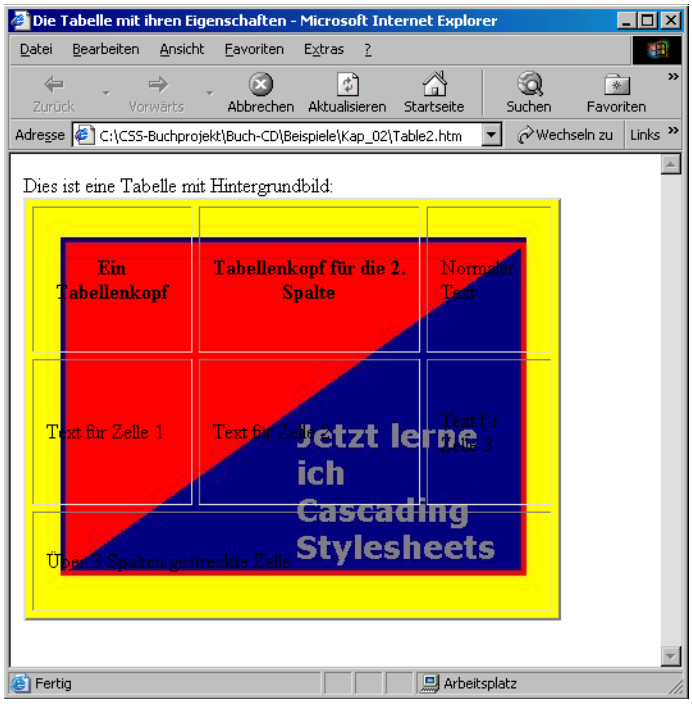

Abb. 2.27: Eine Tabelle mit Hintergrundbild, Tabellenkopf und einer über 3 Spalten gestreckten Zelle.

2.9 XML – Was ist das?

XML ist, genauso wie HTML, eine Auszeichnungssprache. Auch hier gelten bestimmte Regeln. Während HTML aber mit einem begrenzten Satz von vorgegebenen Tags auskommen muss, können Sie in XML eigene Tags kreieren.

Für XML ist erst eine Version verabschiedet worden. Da XML der neue Web-Standard werden soll, können Sie sich sicher denken, welches Konsortium diesen Standard entwickelte – das W3C.

Zunächst mal ist XML selbst eine Metasprache. Sie soll strukturierte Daten beschreiben. Während man HTML nur zur Programmierung von Webseiten nutzen kann, kann XML für Printmedien, Internet, WAP oder PDAs verwendet werden. Mit XML soll auch Inkompatibilität zwischen Datenbanken oder Textverarbeitungsprogrammen behoben werden. Sie können – irgendwann einmal – ein XML-Dokument in allen Textverarbeitungsprogrammen bearbeiten. Ein weiterer wesentlicher Aspekt ist die Plattformunabhängigkeit: XML ist auf allen Betriebssystemen anwendbar, auf Windows, Linux, Macintosh, Unix, DOS etc. Jedes Betriebssystem arbeitet mit einer anderen Kodierung (außer Linux und Unix, die ja miteinander verwandt sind), aber mit XML kann das Problem gelöst werden.

Was hat das alles mit CSS zu tun? Einiges, denn XML beschreibt lediglich die Daten, nicht aber deren Darstellung! Hier kommt CSS ins Spiel: CSS wird genutzt, um das Erscheinungsbild eines XML-Dokuments festzulegen. Es gibt zwar auch andere Sprachen, die XML grafisch darstellen können, wir beschränken wir uns hier aber auf CSS.

Wenn Sie Lust auf »mehr« bekommen haben, sollten Sie unbedingt mal in »Jetzt lerne ich XML« von Günter Born reinschauen. Dieses Buch erschien 2001 ebenfalls bei Markt und Technik und kostet 24,95 € (ISBN: 3-8272-5924-X).

2.10 Einsatz von XML

Nach diesen eher allgemeinen Informationen sollen Sie jetzt erfahren, wie man XML einsetzt.

Zunächst ist es notwendig, dem Browser die XML-Version zu übermitteln (damit er bei einem veralteten Standard auf diesen zurückgreifen kann). Da es derzeit nur eine Version der XML-Sprache gibt, sieht die erste Zeile einer XML-Datei immer so aus:

```
<?xml version="1.0" ?>
```

Darunter folgen Ihre Daten. Ein Vergleich zweier Dokumente – eines in HTML und eines in XML – bietet sich an.

HTML-Code:

```
<HTML>
   <HEAD>
      <TITLE>JLI-B&uuml;cher</TITLE>
   </HEAD>

   <BODY>
      <TABLE border=1>
         <TR>
            <TH>Autor
            <TH>Werk
            <TH>Erscheinungsjahr
            <TH>Preis/EUR
         <TR>
            <TD>Jokat, Daniel
            <TD>JLI Cascading Stylesheets
            <TD>2002
            <TD>24,95
         <TR>
            <TD>Steyer, Ralph
            <TD>JLI JavaScript und HTML
            <TD>1999
            <TD>24,95
         <TR>
            <TD>Taglinger, Harald
            <TD>JLI HTML
            <TD>2000
            <TD>19,95
      </TABLE>
   </BODY>
</HTML>
```

Speichern Sie den Code unter *JLI.htm*.

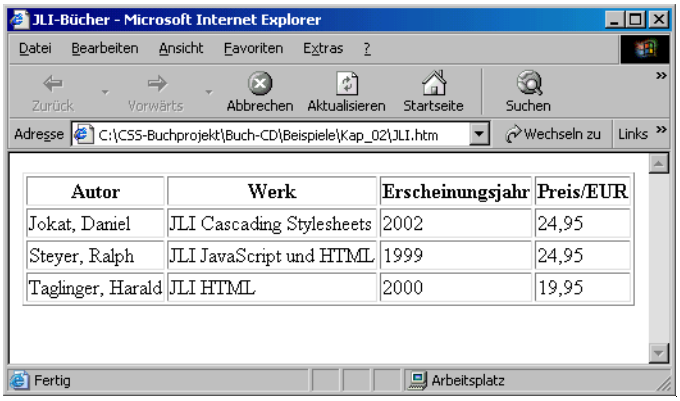

Abb. 2.28: Das HTML-Dokument.

In XML sieht der Code wesentlich übersichtlicher aus:


```
<?xml version="1.0" ?>
<JLI>
   <buch-daten>
      <autor>Jokat, Daniel</autor>
      <werk>JLI Cascading Stylesheets</werk>
      <erscheinungsjahr>2002</erscheinungsjahr>
      <preis einheit="EUR">24,95</preis>
   </buch-daten>

   <buch-daten>
      <autor>Steyer, Ralph</autor>
      <werk>JLI JavaScript und HTML</werk>
      <erscheinungsjahr>2001</erscheinungsjahr>
      <preis einheit="EUR">24,95</preis>
   </buch-daten>

   <buch-daten>
      <autor>Taglinger, Harald</autor>
      <werk>JLI HTML</werk>
      <erscheinungsjahr>2000</erscheinungsjahr>
      <preis einheit="EUR">19,95</preis>
   </buch-daten>
</JLI>
```

Speichern Sie den XML-Code unter *JLI.xml*. Das Resultat im *Internet Explorer*:

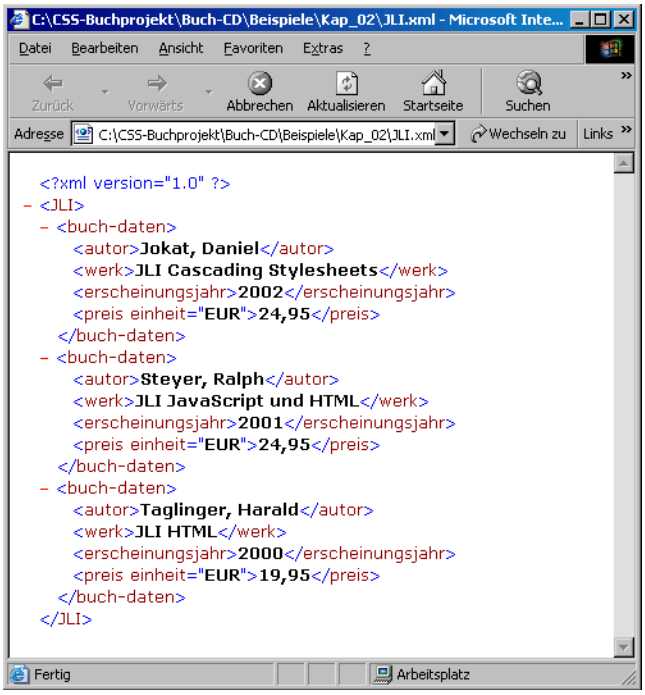

Abb. 2.29: Ihr erstes XML-Dokument.

Wieso soll das nun besser sein als HTML? Das Resultat sieht doch im Gegensatz zu HTML ziemlich schlecht aus. Wie bereits gesagt: XML ist nur zur Beschreibung von strukturierten Daten gedacht und nicht zu deren grafischer Anzeige. Das Erscheinungsbild wird mit CSS festgelegt (Näheres dazu in Kapitel 9).

Was im direkten Vergleich zum HTML-Code sicherlich zuerst auffällt, sind die Endtags. Hier hat wirklich jedes Starttag auch seinen Endtag! Löschen Sie ruhig mal ein Endtag aus dem vorherigen Beispiel und stellen Sie fest, was passiert.

Achten Sie unbedingt darauf, dass Sie jedem Starttag, das Sie definieren, auch ein Endtag zuweisen! Ansonsten bricht der Browser die Verarbeitung des XML-Codes ab!

Auch in XML ist die korrekte Darstellung von Sonderzeichen ein Problem. Allerdings fand man hier eine andere Lösung als in HTML:

In XML erfolgt die Kodierung der Zeichen durch den UTF-Zeichensatz, welcher keine Sonderzeichen unterstützt. Was Sie also machen müssen, um Sonderzeichen verwenden zu können, ist diese Kodierung zu ändern. Und damit lernen Sie Ihr erstes XML-Attribut kennen: `encoding`. Diesem weisen Sie einen der folgenden Zeichensätze zu:

Tabelle 2.10 Zeichensätze

Zeichensatz	**Bemerkung**
UTF-8	internationaler Zeichensatz (8 Bit)
UTF-16	internationaler Zeichensatz (16 Bit)
ISO-8859-1	Westeuropa (Latin-1)
ISO-8859-2	Osteuropa (Latin-2)
ISO-8859-3	Südeuropa (Latin-3)
ISO-8859-4	Nordeuropa (Latin-4)
ISO-8859-5	Kyrillisch
ISO-8859-6	Arabisch
ISO-8859-7	Griechisch
ISO-8859-8	Hebräisch
ISO-8859-9	Türkisch (Latin-5)
ISO-8859-10	Nordisch (Latin-6)

Dazu gleich mal ein – noch fehlerhaftes – Beispiel:


```
<?xml version="1.0" ?>
<text>
   <nebensatz>Wenn Sie bei Sonderzeichen in XML nicht aufpassen,</nebensatz>
   <hauptsatz>tappen Sie in eine blöde  Falle!</hauptsatz>
</text>
```

Wir haben in diesem Code ein Sonderzeichen aber keinen Zeichensatz, der Sonderzeichen unterstützt. Speichern Sie den Code unter *Latin.xml* und sehen sich das fehlerhafte Ergebnis an:

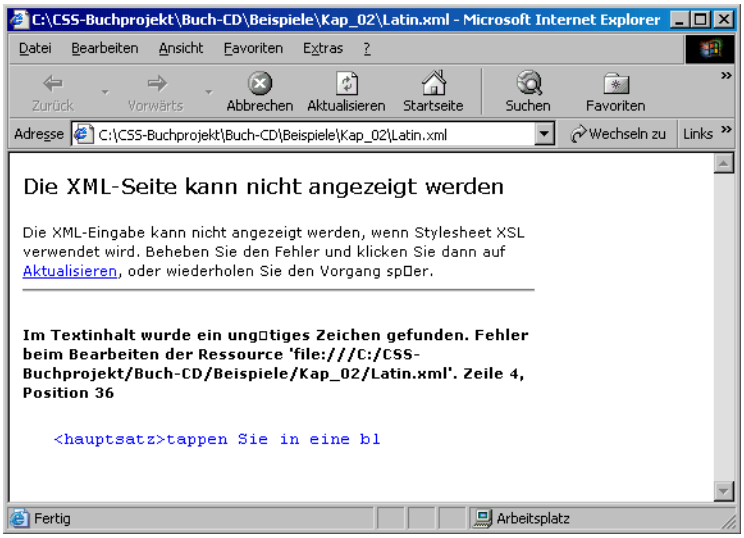

Abb. 2.30: Der Zeichensatz UTF kann keine Sonderzeichen anzeigen.

Betrachten Sie Abbildung 2.30. Das Problem trat also im Tag `<hauptsatz>` auf. Bei dem »ö« bricht die Darstellung ab.

Ändern Sie die erste Zeile des Beispiels wie folgt ab:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

Speichern Sie die Datei abermals als *Latin.xml*. Das verbesserte Ergebnis sieht am Ende so aus:

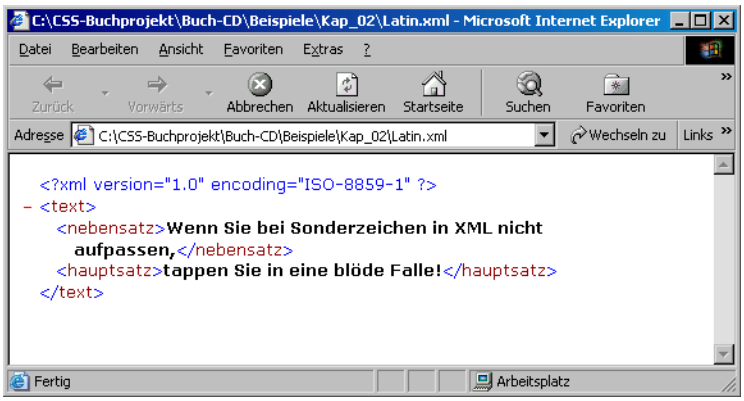

Abb. 2.31: Jetzt sieht's schon besser aus.

Übrigens ist das hier gewählte Beispiel nicht gerade das Beste. Wer trennt schon einen Satz im XML-Dokument mit zwei Tags?

In XML – das ist äußerst wichtig – wird zwischen Groß- und Kleinschreibung unterschieden. Das heißt, dass `<nebensatz>` nicht dasselbe ist wie `<Nebensatz>` oder `<NEBENSATZ>`. Auch an die Notwendigkeit eines Endtags sei nochmals erinnert. Sollten Sie ein Endtag mal nicht brauchen, schließen Sie das Starttag mit einem Leerzeichen und einem Slash, beispielsweise `<demo/>`. Außerdem ist es nicht möglich, Tags in einer anderen Reihenfolge zu beenden, als sie geöffnet wurden. `<B><I>Fett-Kursiver Text</B></I>` ist in XML nicht zulässig. Dies sind ein paar Beispiele aus dem Themenbereich »Wellformed XML«, auf den ich im weiteren Verlauf des Buches aber nicht näher eingehen kann. Obige Grundregeln sind für die Arbeit mit diesem Buch absolut ausreichend.

Zu guter Letzt folgen noch die Kommentaranweisungen. Diese sehen genauso aus wie die HTML-Kommentare:

```
<!-- Kommentar -->
```

Doch aufgepasst! Auch hier muss auf Sonderzeichen geachtet werden. Überlegen Sie sich doch selbst einmal ein Beispiel, in dem Sie in einem Kommentar Sonderzeichen verwenden. Probieren Sie zunächst die »falsche Version«, dann die »richtige Version«.

2.10.1 Zusammenfassung

Sie haben zwar bei weitem nicht alles, aber das Wesentlichste über XML gelernt. Und das reicht für unsere Zwecke. Sie wissen jetzt:

- ✘ XML ist eine Auszeichnungssprache.
- ✘ Es gibt erst eine XML-Version vom W3C.
- ✘ Der Standard-Zeichensatz kann keine Sonderzeichen darstellen.
- ✘ In XML wird zwischen Groß- und Kleinschreibung unterschieden.
- ✘ Zu jedem Starttag gehört ein passendes Endtag.
- ✘ Wenn kein Endtag erforderlich ist, wird ein Slash (»/«) nach dem Tag-Namen in der Klammer angegeben (z.B. `<demo/>`).
- ✘ XML-Kommentare werden wie HTML-Kommentare erstellt.

2.11 Fragen und Antworten

2.11.1 Fragen

1. Wie ist eine HTML-Datei aufgebaut?
2. Wofür ist der HEAD-Bereich zuständig?
3. Welches Tag konfiguriert das Schriftbild?
4. Welches Attribut gibt das Ziel eines Hyperlinks an?
5. Welche Syntaxregeln der XML-Programmierung kennen Sie?

2.11.2 Antworten

1. Der HTML-Code besteht aus einem HTML-Bereich, in dem der HEAD- und der BODY-Bereich liegen.
2. Der HEAD-Bereich umfasst die Daten, die nicht im Dokument angezeigt werden, sondern andere Funktionen haben.
3. Das `<FONT>`-Tag. Es ist zuständig für Schriftart, -größe und -farbe.
4. Das `<A>`-Tag ist für den Hyperlink bestimmt. Das Ziel dieses Links wird durch das Attribut `href` definiert.
5. Die bekannten Syntaxregeln sind:
   - Jedes XML-Tag besitzt ein Endtag.
   - XML unterscheidet Groß-/Kleinschreibung.
   - Ein Tag, das kein Endtag benötigt, erhält ein Slash nach dem Tagnamen (z.B. `<demo/>`).
   - Zur Darstellung von Sonderzeichen muss ein entsprechender Zeichensatz angegeben werden.

2.12 Aufgaben

1. Schreiben Sie ein HTML-Dokument, das Ihren Steckbrief zeigt! Nutzen Sie dazu eine Tabelle. Außerdem soll ein Hyperlink auf Ihre E-Mail-Adresse mit vordefiniertem Betreff eingefügt werden.
2. Erweitern Sie das obige Beispiel um ein persönliches Passfoto. Wenn Sie keine Möglichkeit haben, ein Passbild einzuscannen, malen Sie mit *Paint* einen Ersatz.

3. Schreiben Sie ein XML-Dokument, das ein Adressbuch darstellen soll. Überlegen Sie, welche Tags Sie dazu benötigen, und setzen Sie diese dann sinnvoll ein!

Die Lösungen finden Sie in Anhang E.

KAPITEL 3

Sauberes Programmieren

Bevor es richtig mit der Programmierung von Stylesheets losgeht, sollten Sie einige Tipps bzw. Regeln kennen, die Ihnen später sicherlich nützlich sein werden.

Angenommen Sie bewerben sich bei einem kleinen Betrieb als Webprogrammierer und erfahren nach der Einstellung, dass der alte Webprogrammierer entlassen wurde. Bei der Bearbeitung der aktuellsten Webseite Ihres neuen Betriebs wird Ihnen auch der Grund klar. Wahrscheinlich hatte Ihr Chef beim Betrachten des chaotischen Codes die gleichen Verständnisschwierigkeiten wie Sie.

Eins liegt auf der Hand: der Computer kann den HTML- oder CSS-Code immer lesen, egal wie nachlässig Sie ihn strukturieren. Doch wenn Sie oder gar Andere den Code später verstehen sollen, treten oft Schwierigkeiten auf. In diesem Kapitel lernen Sie, wie Sie Ihren CSS-Code am besten aufbauen und eine geeignete und übersichtliche Struktur finden. Außerdem verfassen Sie Ihr erstes CSS-Stylesheet.

Sie lernen:

- ✘ wie CSS-Code aufgebaut ist,
- ✘ wie CSS-Code verfasst und in HTML eingebettet wird,
- ✘ wie Sie CSS-Code strukturieren sollten.

3.1 Der Aufbau des CSS-Codes

Ein typischer CSS-Code sieht so aus:

```
BODY
{
   font-family: "Verdana, Arial";
}
```

Dieser Code wird Schritt für Schritt analysiert:

In der ersten Zeile erscheint das Wort `BODY`. Hierbei handelt es sich um einen so genannten CSS-Selektor. Der Selektor bezeichnet das HTML-Tag, für das der Stil definiert wird. Wollen Sie also den Stil für alle `<A>`-Tags (also Links) definieren, dann heißt der Selektor hierzu `A`. Es können auch mehrere Selektoren gleichzeitig angegeben werden. Die Stildefinition

```
H1, H3
{
   font-family: "Verdana";
}
```

gilt beispielsweise für die Tags `<H1>` und `<H3>`.

Es folgt der Codeteil für die Gestaltung dieses HTML-Tags. Dieser Codeteil wird von zwei geschweiften Klammern eingeschlossen ({,}). Die Klammern umschließen die eigentlichen Formatierungsbefehle. All das, was zwischen den Klammern festgelegt wird, gilt für den entsprechenden Selektor.

Der Codeteil setzt sich wie folgt zusammen: Zunächst wird eine Stileigenschaft, hier `font-family`, angegeben. Es folgt ein Doppelpunkt und danach wird die Einstellung für diese Stileigenschaft vorgenommen, das heißt, der Eigenschaft wird ein Wert (value) zugewiesen. Jede Eigenschaft-Wert-Zuweisung wird mit einem Semikolon (`;`) abgeschlossen!

Und wie »kommt« der CSS-Code in Ihren HTML-Code? Ein HTML-Tag bringt hier die Lösung.

3.1.1 Das <STYLE>-Tag

Das `<STYLE>`-Tag wird verwendet, um CSS-Code direkt in eine HTML-Datei einzubinden.

Zwischen `<STYLE>...</STYLE>` wird der CSS-Code mit dem oben besprochenen Aufbau eingefügt. Dabei sollte möglichst auch eine bestimmte Struktur eingehalten werden (mehr dazu in Abschnitt 3.3). Ein Beispiel für CSS-Code im `<STYLE>`-Tag:

```
<STYLE>
   BODY
   {
      font-family: "Arial";
   }
</STYLE>
```

Eine Besonderheit des <STYLE>-Tags ist das Attribut media. Mit Hilfe dieses Attributs können Sie angeben, welches Gerät die Stildefinitionen übernehmen soll:

Tabelle 3.1 Werte des media*-Attributs*

Werte für media	**angesprochenes Gerät**
all	alle Geräte
braille	Geräte mit Unterstützung von Blindenschrift
print	Drucker
projection	Projektor
screen	Monitor (Bildschirm)
speech	Sprachsynthesizer

Der Vorteil ist klar ersichtlich: durch die unterschiedliche Darstellung auf verschiedenen Geräten, die man hier festlegen kann, ist es beispielsweise möglich, einen Ausdruck in einer Formatierung zu ermöglichen, die von der Formatierung auf dem Bildschirm abweicht. Beispiel »Arial« versus »Times New Roman«: Auf dem Bildschirm ist der Schrifttyp »Arial« unschlagbar und »Times New Roman« eher trist. Drucken Sie das Dokument jedoch aus, sieht »Arial« wie eine Kinderschrift aus (»Arial« wird übrigens für viele Kinderbücher verwendet), während »Times New Roman« professionell wirkt.

Wir werden später noch einmal auf diese Technik zurückkommen und, sobald wir einige CSS-Stileigenschaften kennen gelernt haben, auch ein Beispiel dazu schreiben.

Ein Attribut, das auch sehr oft im Zusammenhang mit <STYLE> verwendet wird, heißt type. Dies ist lediglich für den Browser von Bedeutung, denn hier wird die verwendete Stylesheet-Sprache angegeben. Dies ist in unserem Falle CSS. So kommen wir auf den Wert text/css (wie Sie sehen wird hier auch der Inhaltstyp text berücksichtigt).

Um das obige Beispiel so zu verändern, dass die Attribute type und media Verwendung finden, müssen Sie nur die erste Zeile anpassen. Wenn das media-Attribut weggelassen wird, gilt das Stylesheet für alle Medientypen. Um diese Einstellung und den Gültigkeitsbereich des letzten Beispiels nicht einzugren-

zen, müssen Sie dem Attribut `media` den Wert `all` zuweisen. Folglich sieht die abgeänderte erste Zeile so aus:

```
<STYLE type="text/css" media="all">
```

Auf das `media`-Attribut werden wir im weiteren Verlauf des Buches verzichten. Folglich gelten die Beispiele im Buch für alle Medientypen.

Zu guter Letzt bleibt nur noch zu klären, wo das `<STYLE>`-Tag eingefügt wird? Kurz und bündig: Es kommt in den Header, sprich in den HEAD-Bereich.

3.2 Das erste CSS-Stylesheet

Nun folgt ein komplettes CSS-Beispiel und dessen Analyse. Da wir den Code, auch den HTML-Code, zeilenweise behandeln werden, wurden die Zeilen mit Buchstaben durchnummeriert. Diese gehören nicht zum Code, tippen Sie sie also nicht mit ab.


```
A   <HTML>
B      <HEAD>
C         <TITLE>Ihr erstes CSS-Stylesheet</TITLE>
D         <STYLE type="text/css">
E            BODY
F            {
G               font-family: "Verdana";
H               font-size: 24pt;
F            }
D         </STYLE>
B      </HEAD>

I      <BODY>
J         Herzlichen Gl&uuml;ckwunsch! Sie haben Ihr
J   erstes CSS-Stylesheet verfasst.
I      </BODY>
A   </HTML>
```

Speichern Sie den Code unter *firstcss.htm* ab. So sieht Ihr erstes HTML-Dokument mit Stylesheets aus (erinnern Sie sich: Die Standard-Schrift wäre »Times New Roman«):

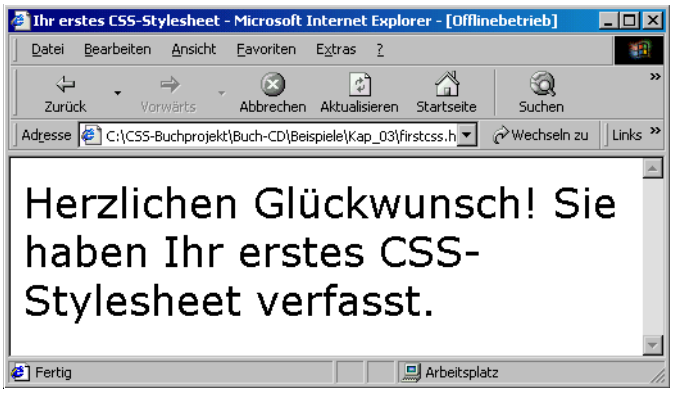

Abb. 3.1: Der Text der Abbildung sagt es bereits! Sie haben es geschafft, wenn's so aussieht

Die zeilenweise Erklärung des Codes erläutert detailliert, wie dieses Dokument »funktioniert«:

Zeile A:

Der HTML-Bereich.

Zeile B:

Der HEAD-Bereich.

Zeile C:

Der Titel des HTML-Dokuments.

Zeile D:

Das `<STYLE>`-Tag für die Einbettung unseres Stylesheets.

Zeile E:

Der CSS-Selektor. Er heißt `BODY` und konfiguriert das gleichnamige HTML-Tag.

Zeile F:

Die geschweiften Klammern, die den Bereich der Einstellungen für den CSS-Selektor festlegen.

Zeile G:

Die Stileigenschaft `font-family`. Es ist zuständig für die Schriftart, die für das HTML-Tag verwendet werden soll. Sie lernen es in Kapitel 4 näher kennen.

Zeile H:

Die Stileigenschaft `font-size`. Es ist zuständig für die Schriftgröße, die für das HTML-Tag verwendet werden soll. Sie lernen es in Kapitel 4 näher kennen.

Zeile I:

Der BODY-Bereich.

Zeile J:

Der Text, der im `<BODY>`-Tag gesetzt wurde. Da die CSS-Einstellungen aus Zeile E-H für das `<BODY>`-Tag gelten, wird dieser Text in den vorgenommenen Einstellungen formatiert, im Beispiel also in der Schriftart »Verdana« mit Schriftgröße 24.

Diese erste Begegnung mit einem einfachen CSS-Stylesheet soll eventuell vorhandene Befürchtungen vor vermeintlich Kompliziertem ausräumen und dazu motivieren, weiterzumachen. Dieses erste Stylesheet ist so konzipiert, dass es Ihnen das Grundkonzept von CSS in einem praxisnahen Beispiel vor Augen führt. Und mit gutem Gewissen lässt sich sagen: Es wird nicht viel schwerer.

3.3 Eine geeignete Struktur für CSS-Code

Wie Ihr Code strukturiert ist, ist für den Browser gänzlich unwichtig. Solange die Syntaxregeln eingehalten werden, ist ihm das äußere Erscheinungsbild des Codes egal. Schwierig wird's erst, wenn Sie oder andere Ihren Code beim Überarbeiten richtig verstehen sollen, und deshalb ist es ratsam, den Code übersichtlich zu strukturieren.

3.3.1 Übersichtlicher Aufbau von Stylesheets

Wie Sie mittlerweile wissen, besteht CSS-Code aus Selektoren, Stileigenschaften und Werten. Diese drei Bestandteile bestimmen auch die Struktur. Greifen wir ein CSS-Codefragment auf, das weiter oben schon zur Veranschaulichung verwendet wurde:

```
BODY
{
   font-family: "Verdana, Arial";
   font-size: 10pt;
}
```

Anhand dieses Beispiels sehen Sie, was mit einer guten Struktur gemeint ist. Der Selektor steht ohne Einrückung oben ganz am linken Rand. Dann sollte ein Zeilenumbruch folgen.

Das nächste Merkmal ist die geöffnete geschweifte Klammer, die dem Browser mitteilt, dass nun die Einstellungen für den Selektor folgen. Schließen Sie auch diese Codezeile mit einem Zeilenumbruch ab.

In der dritten Zeile folgt die erste CSS-Eigenschaft. In diesem Falle `font-family`. Hier sollten Sie einen Tabulator setzen (⇤⇥-Taste) oder zumindest mit der Leertaste-Taste eine Einrückung vornehmen. Eine geeignete Einrückungsweite sind drei Leerzeichen (wie sie in den Beispielen im Buch auch verwendet werden).

Hinter der Stileigenschaft steht ein Doppelpunkt, dem der Browser entnehmen kann, dass nun der Wert für die Eigenschaft folgt. Trennen Sie den Wert und die Eigenschaft durch ein Leerzeichen.

Der Wert wird manchmal in Anführungszeichen gesetzt, was aber nicht immer erforderlich ist (für `font-family` können Sie – wie in HTML auch – durch Kommata getrennt mehrere alternative Schriftarten als Wert angeben). Nachdem Sie den Wert eingegeben haben, schließen Sie die Einstellung mit einem Semikolon (`;`) ab. Vor der nächsten Einstellung nehmen Sie einen Zeilenumbruch vor.

Die gleichen Regeln gelten für alle Stileigenschaften. Wenn Sie die gewünschten Eigenschaften für den Selektor aufgesetzt haben, drücken Sie die Taste ↵, um einen Zeilenumbruch einzufügen. Die geschweifte Klammer wird wieder geschlossen.

Wenn Sie einen weiteren Selektor konfigurieren wollen, erstellen Sie einen neuen Absatz (2 Zeilenumbrüche von der geschlossen Klammer aus gezählt, sprich 2x die Taste ↵). Nun folgt das gleiche Spiel: Selektor, Zeilenumbruch, öffnende geschweifte Klammer, Zeilenumbruch usw.

Nach diesem Muster können Sie Stylesheets aufsetzen, die mit Hilfe des `<STYLE>`-Tags in den HTML-Code einer Webseite eingebettet (siehe oben) oder in eine externe Datei geschrieben werden (siehe Kapitel 4.1).

3.3.2 Aufbau von Stildefinitionen für das style-Attribut

Wie bereits in Kapitel 1 erwähnt, gibt es aber auch die Möglichkeit, Stileigenschaften mit Hilfe des HTML-Attributs `style` direkt an einzelne HTML-Elemente im BODY-Bereich einer Webseite (beispielsweise einem einzelnen Absatz, einem Listenelement oder auch dem `BODY`-Tag selbst) zuzuweisen. Diese Form der Zuweisung erfordert allerdings eine eigene Syntax.

Zuerst einmal fallen der Selektor und die geschweiften Klammern weg:

```
<BODY style="font-family: 'Verdana, Arial'; font-size: 1Opt">
```

Da das Attribut `style` im `<BODY>`-Tag verwendet wird, versteht sich wohl von selbst, dass die Einstellungen für eben dieses Tag gelten. Daher fallen der Selektor und die geschweiften Klammern weg.

Was nun aber statt der geschweiften Klammern zu beachten ist, sind die doppelten Anführungszeichen, in die die Einstellungen gesetzt werden. Die vormals doppelten Anführungszeichen für die Stilwerte werden zu einfachen Anführungszeichen (').

Nachdem Sie also das doppelte Anführungszeichen geöffnet haben, wird die erste Stileigenschaft eingefügt, im Beispiel `font-family`. Es folgt der Doppelpunkt. Nun sollte wieder ein Leerzeichen eingefügt werden, um den/die Wert(e) vom Eigenschaftenname abzugrenzen. Falls erforderlich, öffnen Sie nun das einfache Anführungszeichen.

Nehmen Sie nun die Einstellung für die Eigenschaft vor. Im Beispiel wurden die Schriftarten »Verdana« und »Arial« eingestellt. Vergessen Sie nicht das abschließende einfache Anführungszeichen!

Es folgt das Semikolon. Dadurch weiß der Browser, dass die Definition der ersten Stileigenschaft zu Ende ist. Für die nächste Stileigenschaft fügen Sie wieder ein Leerzeichen ein.

Nun können Sie weitere Angaben machen, dürfen aber die doppelten Anführungszeichen noch nicht schließen. Nehmen Sie nun die nächste Einstellung vor, im Beispiel `font-size`.

Weiter: Doppelpunkt, Leerzeichen, Wert, Semikolon, Leerzeichen, Eigenschaftsname, Doppelpunkt, Leerzeichen, Wert, Semikolon usw.

Wenn Sie die Einstellungen für das Tag beenden haben, setzen Sie die schließenden doppelten Anführungszeichen.

Zur Verdeutlichung des style-Attributs soll der Code aus *firstcss.htm* umgeschrieben werden:


```
<HTML>
   <HEAD>
      <TITLE>Die Verwendung von style</TITLE>
   </HEAD>

   <BODY style="font-family: 'Verdana'; font-size: 24pt;">
     Herzlichen Gl&uuml;ckwunsch! Sie haben Ihr erstes CSS-Stylesheet mit dem
Attribut <TT>style</TT> verfasst.
   </BODY>
</HTML>
```

Das Beispiel finden Sie auf der Buch-CD unter *Beispiele\Kap_03\style.htm*.

3.4 Zusammenfassung

Dies war Ihr erster Kontakt mit CSS-Stylesheets. Wiederholen wir kurz die wichtigsten Punkte:

- ✘ eine Stylesheetdefinition beginnt mit einem Selektor,
- ✘ der Einstellungsbereich wird durch geschweifte Klammern eingegrenzt,
- ✘ jede Einstellung wird mit einem Semikolon (**;**) abgeschlossen,
- ✘ bemühen Sie sich um eine übersichtliche Code-Struktur,
- ✘ meist werden Stylesheets über das `<STYLE>`-Tag eingebunden, sei es direkt oder über den Befehl `@import` (den Sie noch kennen lernen werden),
- ✘ Stildefinitionen, die nur für ein einzelnes HTML-Element im BODY-Bereich gelten, können auch direkt über das HTML-Attribut `style` definiert werden,
- ✘ Stylesheets können für die Ausgabe auf verschiedenen Geräten eingesetzt werden, indem man dem `<STYLE>`-Tag das Attribut `media` übergibt. Mit diesem können Sie das/die Gerät(e) festlegen, für die das Stylesheet gelten soll.

3.5 Fragen und Antworten

3.5.1 Fragen

1. Welche Bestandteile muss ein Stylesheet, das eine Einstellung vornimmt, mindestens aufweisen?
2. Welches Tag wird benutzt, um den CSS-Code direkt in HTML einzubinden?
3. Was wird mit dem Attribut `media` festgelegt?
4. Stimmt diese Aussage: »Die Struktur des CSS-Codes muss immer gleich bleiben.«

3.5.2 Antworten

1. Selektor, geschweifte Klammern (auf/zu), Stileigenschaft, Wert (Doppelpunkt, Semikolon).
2. Das `<STYLE>`-Tag.
3. Mit `media` kann das/die Gerät(e) festgelegt werden, für das das Stylesheet gilt. So kann am Bildschirm und auf dem Papier ein unterschiedliches Layout gelten.
4. Nein, diese Aussage ist falsch. Die Struktur kann frei gewählt werden. Sie sollte aber einheitlich sein, damit man den Code später selbst wieder richtig interpretieren kann.

3.6 Aufgaben

1. Schreiben Sie eine HTML-Datei, zunächst ohne Stylesheet, die einen Ausschnitt aus einem Roman zeigt. Verwenden Sie dabei den Standard des Browsers, das heißt, benutzen Sie keine Tags oder Attribute, die das Aussehen des Textes verändern. Den Text finden Sie auf der Buch-CD als *Beispiele\Kap_03\roman.rtf*. Das RTF-Format kann, wenn Sie *Word 97/2000/2002* nicht besitzen, auch mit *WordPad* eingesehen werden (finden Sie im Startmenü unter ZUBEHÖR).
2. Erweitern Sie Aufgabe 1 um ein Stylesheet, das den Text in der Schriftart »Tahoma« mit Größe 14 Punkt formatiert. Versuchen Sie es zu schaffen, ohne nach unserem ersten Beispiel zu blättern.

Die Lösung zu Aufgabe Zwei entspricht dem ersten Beispiel in diesem Kapitel.

KAPITEL 4

CSS im Einsatz

Nun ist es an der Zeit, die einzelnen CSS-Befehle, -Eigenschaften und -Werte im Detail kennen zu lernen – endlich. Sie werden nach der Lektüre dieses Kapitels weit mehr als nur Schriftart und -größe ändern können:

- die Verwendung des Befehls `@import`
- die Verwendung des Tags `<LINK>`
- die Veränderung von Schriftart und -größe
- die Veränderung des Schriftbildes
- die Ausrichtung von Elementen an anderen
- die Veränderung von Farbe mit CSS
- die Konfiguration von Hyperlinks (Farbe, Aussehen)
- Pseudo-Klassen
- die Auswahl von Aufzählungszeichen für Listen
- die Änderung von Tabelleneinstellungen
- der Einsatz von Rahmen
- der Einsatz von CSS-Klassen
- die Darstellung von Absätzen
- die Änderung des Buchstaben-/Zeilenabstandes
- die Stilzuweisung einzelner Textpassagen
- die Priorität von Styles im Dokument

4.1 Die Einbindung von externem CSS-Code in HTML

Statt den CSS-Code mit den Stildefinitionen direkt in den Code der Webseite zu schreiben, können Sie den CSS-Code auch in eine externe Datei auslagern und diese dann in die betreffenden Webseiten importieren. Dieses Verfahren ist vor allem dann von Vorteil, wenn Sie mehrere Webseiten mit dem gleichen CSS-Code formatieren möchten.

Stellen Sie sich vor, Sie haben eine Homepage, die sich aus 100 HTML-Dokumenten zusammensetzt. Sie entschließen sich, mit dem `<FONT>`-Tag eine eigene Schriftart vorzugeben, beispielsweise Arial. Irgendwann erkennen Sie, dass die verwendete Schriftart doch nicht so gut aussieht und Sie lieber Verdana verwenden würden. Jetzt erkennen Sie den Fehler: Weil Sie in jedem einzelnen Dokument das `<FONT>`-Tag verwendet haben, müssen Sie jetzt in allen 100 Dateien die Schrift gesondert ändern. Hätten Sie eine externe CSS-Datei erstellt und diese importiert, müssten Sie die Schriftart nur einmal in der CSS-Datei neu setzen.

Wenn Sie sich für externen Code entscheiden, so bedenken Sie, dass Sie eine eigene Datei anlegen müssen, in der sich der CSS-Code befindet. Diese Datei muss die Endung ».css« tragen.

4.1.1 Der Befehl @import

Dieser Befehl gehört in das `<STYLE>`-Tag, das Sie schon in Kapitel 3 kennen gelernt haben. Er teilt dem Browser mit, dass an dieser Stelle CSS-Code aus einer anderen Datei gesetzt werden soll.

Zunächst verfassen Sie den CSS-Code in einer eigenen Datei. Den Namen dieser Datei können Sie frei wählen, achten Sie aber darauf, dass sie die Endung *.css* trägt.


```
BODY
{
   font-family: "Verdana";
   font-size: 14pt;
}
```

Speichern Sie diesen Code als *import.css*.

Wenn Sie beim Schreiben des HTML-Codes zum `<STYLE>`-Tag gekommen sind, importieren Sie den Code aus der CSS-Datei über den Befehl `@import`. Der Ort der CSS-Datei wird mit Hilfe von `url()` angegeben:

```
<HTML>
   <HEAD>
      <TITLE>Import von CSS-Code &uuml;ber @import</TITLE>

      <STYLE type="text/css">
         @import url(import.css);
      </STYLE>
   </HEAD>

   <BODY>
      Der Text dieses HTML-Dokuments wurde mit externem CSS-Code formatiert.
Der CSS-Code wurde &uuml;ber den @import-Befehl eingef&uuml;gt.
   </BODY>
</HTML>
```

Beachten Sie, dass zwischen dem Befehl `@import` und der `url()`-Angabe ein Leerzeichen eingefügt ist.

Speichern Sie diesen Code unter *import.htm*. Stellen Sie sicher, dass sich *import.css* und *import.htm* im selben Verzeichnis befinden, sonst klappt das Beispiel nicht.

Öffnen Sie die Datei *import.htm* in Ihrem Browser und betrachten Sie das Ergebnis.

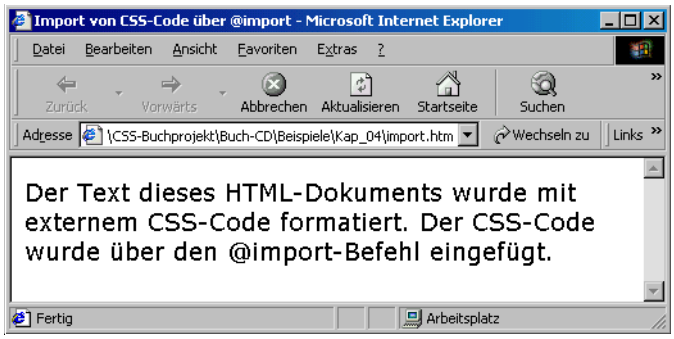

Abb. 4.1: Von einer CSS-Datei auf's HTML-Dokument übertragen.

Bei *Netscape* funktioniert der `@import`-Befehl erst ab Version 6!

Wie das Ganze mit einer einzigen Zeile geht, erfahren Sie im nächsten Unterabschnitt.

4.1.2 Das <LINK>-Tag

Neben `@import` bietet das `<LINK>`-Tag eine weitere Möglichkeit der Einbindung externen CSS-Codes.

In HTML ist `<LINK>` ein Tag, das Beziehungen zwischen zwei Dokumenten einrichtet. Um CSS-Code damit einzubinden, merken Sie sich diese eine Zeile:

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

Damit wissen Sie über das Einbinden von externem CSS-Code in HTML mit `<LINK>` schon alles. Ein Beispiel noch dazu:


```
BODY
{
   font-family: "Tahoma";
   font-size: 12pt;
}
```

Speichern Sie diesen Code als *LINK.css*.

```
<HTML>
   <HEAD>
      <TITLE>Einbinden von externem CSS-Code mit &lt;LINK&gt;</TITLE>
      <LINK rel=Stylesheet href="LINK.css" type="text/css">
   </HEAD>

   <BODY>
      Genau wie mit <TT>@import</TT>, kann man mit &lt;LINK&gt; externen CSS-
Code ins HTML-Dokument importieren.
   </BODY>
</HTML>
```

Speichern Sie den Code als *LINK.htm*.

Sehen Sie sich die Datei im Browser an.

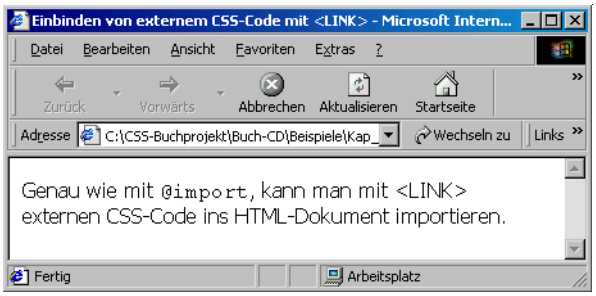

Abb. 4.2: Mit `<LINK>` geht's auch.

4.2 Textformatierung mit CSS

4.2.1 Schriftart und -größe einstellen

Soweit Sie das vorherige Kapitel gelesen haben, kennen Sie die CSS-Eigenschaften für das Setzen der Schriftart und -größe schon. Der Vollständigkeit halber seien Sie hier aber trotzdem kurz dargestellt.

Der CSS-Stileigenschaft `font-family` wird die Schriftart zugewiesen. Aus Kapitel 2 wissen Sie bereits, dass Sie in HTML mit `<FONT>` eine Mehrfachdeklaration von Schriftarten vornehmen können. Das ist auch mit CSS möglich.

Schriftart-Mehrfachdeklaration

Falls Sie das 2. Kapitel nicht gelesen haben, wissen Sie vielleicht noch nicht, was eine Mehrfachdeklaration von Schriftarten ist und wozu sie dient.

Angenommen, Sie bieten Ihre Homepage in der Schriftart »Wide Latin« an. Diese Schriftart ist bei Windows nicht vorinstalliert, was dazu führt, dass Ihre ganzen Formatierungen bei den Betrachtern, die unter Windows arbeiten, verloren gehen.

Sie sollten also Ihr Angebot vorher noch in einer anderen Formatierung testen, zum Beispiel mit »Arial«, und die Darstellung in beiden Schriftarten vorsehen.

Dazu geben Sie einfach beide Schriftarten an. Die Schriftart, die Sie zuerst angeben, ist für Ihre Homepage die bevorzugte Schrift, die andere wird nur verwendet, wenn die zuerst angeführte Schriftart nicht auf dem System des Betrachters gefunden wird.

Die Schriftart wird in Anführungszeichen gesetzt.

Die Schriftgröße wird über die Stileigenschaft `font-size` definiert. Sie wird standardmäßig in Punkt angegeben. Das erfordert den Zusatz `pt`, sodass eine Schriftgrößendefinition wie folgt aussieht:

```
font-size: 12pt;
```

Anführungszeichen sind hier nicht notwendig. Statt durch Zahlenwerte können Sie die Schriftgröße auch durch vordefinierte Konstanten, neun an der Zahl, angeben.

Tabelle 4.1 Konstanten für die Angabe der Schriftgröße

Konstante für font-size	Schriftgröße
`xx-small`	winzig
`x-small`	sehr klein
`smaller`	größer als `x-small` und kleiner als `small`
`small`	klein
`medium`	normal
`large`	groß
`larger`	größer als `large` und kleiner als `x-large`
`x-large`	sehr groß
`xx-large`	riesig

Daneben besteht die Möglichkeit, einen Prozentwert anzugeben (dieser bezieht sich auf die Schriftgröße des umliegenden HTML-Elements). Außerdem können Sie die Länge in »em« angeben. Diese Einheit ist der Platz, den der Buchstabe »M« im umliegenden Text einnimmt, bei Times New Roman Schrift mit 10 Punkt entspricht das ca. ¼ Zentimeter.

Ein Beispiel ist an dieser Stelle nicht nötig. Sehen Sie sich das aus Abschnitt 4.1.1 oder Abschnitt 4.1.2 an. Dort werden diese Eigenschaften verwendet.

Die Eigenschaft zur Einstellung der Schriftarten heißt `font-family`, was übersetzt Schriftfamilie bedeutet. Das heißt, dass Sie neben speziellen Schriftarten auch eine der Schriftfamilien angeben können.

Die Schriftfamilien sind: »Serif«, »Sans-Serif«, »Cursive«, »Fantasy« und »Monospace«. An einem kleinen Beispiel lässt sich die Programmierung und das Aussehen dieser Schriftfamilien schnell aufzeigen. Hierfür müssen wir das HTML-Attribut `style` verwenden. Eine weitere Lösung werden Sie übrigens später kennen lernen.


```
<HTML>
   <HEAD>
      <TITLE>Die Schriftfamilien</TITLE>
   </HEAD>

   <BODY>
      <BASEFONT size=4>
      <H1 align="center">Die Schriftfamilien</H1>
      Dieses HTML-Dokument zeigt die verschiedenen Schriftfamilien an.
      <P style="font-family: 'Serif';">Zun&auml;chst zeigen wir die Schrift-
familie &quot;Serif&quot;.</P>
      <P style="font-family: 'Sans-Serif';">Die zweite Schriftfamilie
hei&szlig;t &quot;Sans-Serif&quot;.</P>
```