Kapitel 3

Layouthilfen und Ordnung schaffen

Als Beispiel werden wir in diesem Kapitel ein kleines Inhaltsverzeichnis für verschiedene Texte eines Autors anlegen. Diese Texte sollen dann mehrspaltig auf der Webpage erscheinen. Um nicht unnötige Längen in diesem Buch zu erzeugen, werden nur die relevanten Teile eines Codes hier wiedergegeben. Am Schluss des Kapitels steht der gesamte Source-Code zur Sicherheit noch einmal codiert aufgeführt.

Nach den ersten Übungen in HTML wollen wir in diesem Kapitel lernen, wie man mehr Ordnung in seine Web-Texte bekommt. Zuerst schauen wir uns deshalb die Möglichkeiten von Listen an und befassen uns dann mit Tabellen und verschachtelten Tabellen. Das ist ein guter Sprungpunkt zum Spaltensatz in HTML und sogar zur pixelgenauen Positionierung von Elementen, denn das lösen Webdesigner zu großen Teilen auch über unsichtbare Tabellen.

Sie üben in diesem Kapitel:

- ✘ Anlegen von verschiedenen Listenarten
- ✘ Anlegen von Tabellen
- ✘ Verschachteln von Tabellen
- ✘ Tabelle als Layouthilfe
- ✘ Pixelgenauer Satzstand und Präformatierungen

3.1 Aufzählungen

Nehmen Sie an, Sie möchten nicht immer nur alles in einem Fließtext webpublishen. Eine Möglichkeit wäre es, mit Aufzählungen zu arbeiten. Und hier hätten Sie nach dem ersten Viertel dieses Buches auch schon das Know-how dazu, eine Lösung mit den bisherigen Tags zu finden: Sie würden einfach nach jedem Aufzählungspunkt einen *
*- oder *<P>*-Tag nutzen und das Ganze vielleicht mit *<BLOCKQUOTE>*-Tags einrücken.

Aber das wäre ja öde.

HTML sieht mehrere Möglichkeiten vor, hier per Tag verschiedene, auch automatisch durchnummerierte Lösungen, dafür bereitzustellen. Wir haben schon bei den Anker-Tags festgestellt, dass ein Befehl oft aus zwei Tags besteht, die unterschiedlich voneinander und sogar an vollkommen anderen Stellen des Dokuments wirksam werden können. Hier haben wir noch einmal eine Besonderheit, die wir auch im nachfolgenden Kapitel brauchen werden:

HTML legt Listen so an, dass zuerst einmal der generelle Listentyp festgelegt wird. Wir haben hier folgende Typen als Tag zur Auswahl:

`<DL></DL>`	als Definitionsliste
`<DIR><DIR>`	als Directoryliste (für Inhaltsverzeichnisse)
`<OL></OL>`	als nummerierte Liste
`<UL></UL>`	als unnummerierte Liste

Damit haben wir noch nichts über die Listenelemente gesagt. Wir haben zuerst einmal einen Grundtyp der Liste festgelegt. Bei den Listenelementen lassen sich drei verschiedene aufführen, wobei zwei Tags ganz klar zueinander gehören:

`<LI>`	eine einfache Aufzählung
`<DT>`	Definitionsterm (also eine Art von Überschrift)
`<DD>`	Definitionsdaten (also der Inhalt dazu)

Meistens reicht der *<LI>*-Tag, da die anderen beiden sehr auf wissenschaftliche Arbeiten abgestimmt sind. Sie merken also immer wieder, wie stark HTML den Bedürfnissen der Wissenschaftler in CERN entgegenkam. Allerdings sollten Sie sich wirklich fragen, ob Sie eine ähnliche Differenzierung für – sagen wir – eine Angler-Homepage brauchen. Es liegt ganz bei Ihnen.

Wir geben also immer zuerst einen Grundtyp der Liste an, zwischen deren beiden Tags dann die Listenelemente geschachtelt werden.

Listenelemente besitzen keinen Schluss-Tag. Das ist etwa mit einem *
*-Tag vergleichbar, und wir werden auch gleich sehen, dass hier durchaus Ähnlichkeiten vorhanden sind.

Gehen wir diese verschiedenen Typen an unserem Kapitelbeispiel durch.

Angenommen, Sie haben eine Website vor, auf der ein Inhaltsverzeichnis mit folgenden Einträgen zu finden sein wird:

```
Inhaltsverzeichnis

1. Der erste Text: Montag
2. Der zweite Text: Dienstag
3. Der dritte Text: Mittwoch
4. Der vierte Text: Donnerstag
```

In einem üblichen HTML-File würden Sie diesen Text wie folgt umsetzen:

```
<HTML>
   <HEAD>
      <TITLE>Inhaltsverzeichnis</TITLE>
   </HEAD>

<BODY>

<H1> Inhaltsverzeichnis</H1>

1. Der erste Text: Montag<BR>
2. Der zweite Text: Dienstag<BR>
3. Der dritte Text: Mittwoch<BR>
4. Der vierte Text: Donnerstag<BR>

</BODY>

</HTML>
```

Wie gesagt: Das wäre banal. Die Umsetzung per Ordered List sieht deshalb in einem HTML-File so aus:

```
A   <OL>
B     <LI> Der erste Text: Montag
      <LI> Der zweite Text: Dienstag
      <LI> Der dritte Text: Mittwoch
      <LI> Der vierte Text: Donnerstag
   </OL>
```

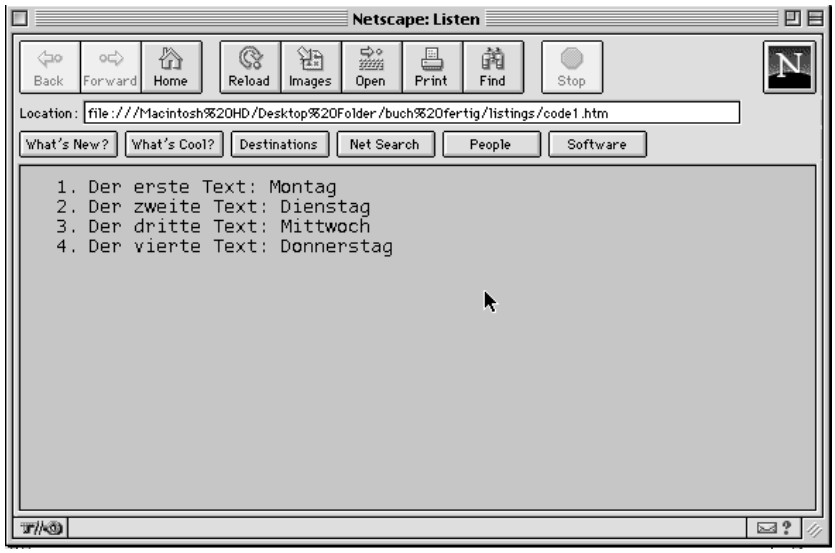

Bild 3.1: Schon ist die erste Liste fertig.

```
A   <OL>
```

Am Ende des Kapitels finden Sie den kompletten HTML-Code, in den Sie die nun folgenden unvollständigen Beispiele CODE 1 bis CODE 9 und CODE A bis CODE G einfügen können.

Hier wird im BODY-Bereich des HTML-Files die Ordered List aufgemacht. Die folgenden Zeilen rücken automatisch ein und werden bis zum Schluss-Tag zur Aufzählung.

```
B   <LI> Der erste Text: Montag
```

Durch diesen einfachen Tag, der keinen Schluss-Tag besitzt, werden jetzt automatisch die Nummern der Aufzählungen dazu addiert. Das ist immens praktisch und sicher. Der folgende Fließtext wird nun eingerückt. Der erneute *<LI>*-Tag ersetzt hier den *
*-Tag. Ein Zeilenumbruch erfolgt also automatisch.

Wenn Sie nur den *<OL>*-Tag mit einem Schluss-Tag nutzen, dann rückt der Text ähnlich wie *<BLOCKQUOTE>* ein. Aber ich empfehle Ihnen, nur sehr vorsichtig mit Code zu hantieren, der nicht sauber angewandt wird.

HTML ist ursprünglich für wissenschaftliche Arbeiten konzeptioniert und zum Word Wide Web portiert worden. Deshalb sind diese Aufzählungen so bequem zu programmieren.

Schreiben Sie nun unsere Beispielliste vom Kapitelanfang als Unordered List um.


```
A   <UL>
B     <LI> Der erste Text: Montag
      <LI> Der zweite Text: Dienstag
      <LI> Der dritte Text: Mittwoch
      <LI> Der vierte Text: Donnerstag
    </UL>
```

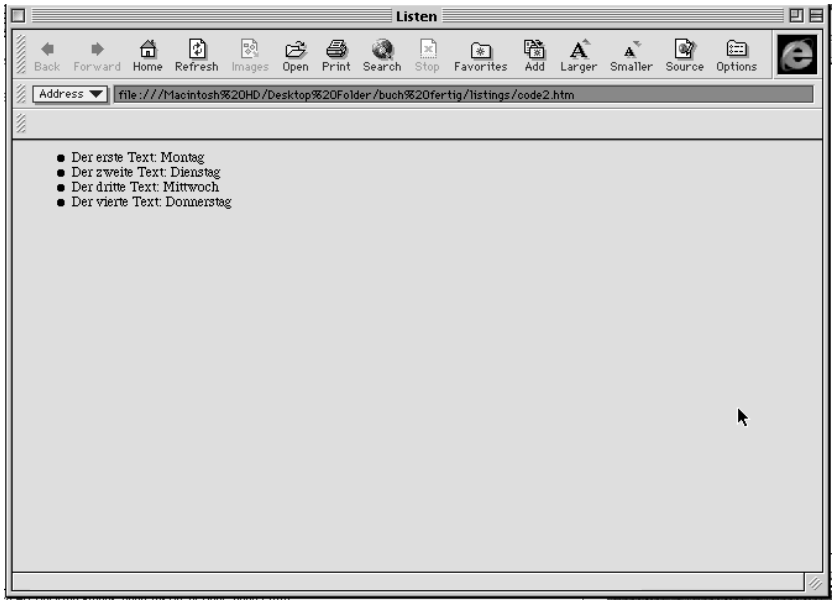

Bild 3.2: Statt der Zahlen entstehen jetzt Listenpunkte.

```
A   <UL>
```

Dieser Tag eröffnet eine Unordered List, die ebenfalls einrückt und die weiteren Zeilen bis zum Schluss-Tag als Listeninhalt behandelt.

```
B   <LI> Der erste Text: Montag
```

Jetzt verschwinden die Aufzählungen, und statt dessen wird ein Listenpunkt eingefügt.

Der *<DIR></DIR>*-Tag reagiert bei Browsern von Microsoft und von Netscape ähnlich, deshalb wird er hier nicht mehr eigens behandelt.

Wir haben jetzt also schon zwei Arten von Listen kennen gelernt. Aber angenommen, das Inhaltsverzeichnis soll wie folgt aussehen:

```
Der erste Text:
   Montag
(Etc.)
```

Kommt hier die weitere Listenform zum Einsatz, die für Definitionslisten vorgesehen ist. Unser Inhaltsverzeichnis sieht somit in HTML wie folgt aus:

```
A   <DL>
B      <DT> Der erste Text:
C         <DD>Montag
       <DT> Der zweite Text:
          <DD>Dienstag
       <DT> Der dritte Text:
          <DD>Mittwoch
       <DT> Der vierte Text:
          <DD>Donnerstag
   </DL>
A   <DL>
```

Hier wird die Definitionsliste in der bereits bekannten Art wieder geöffnet und aktiv bis zum Schluss-Tag. Im Unterschied zu bisher schon bekannten Listen rückt dabei der Listeninhalt nicht nach links.

```
B   <DT> Der erste Text:
```

Diese Überschriftenzeile wird nicht eingerückt, das bleibt der *<DD>*-Aufzählung vorbehalten.

```
C   <DD>Montag
```

Die Datenzeile rückt nach einem Zeilenumbruch einen Tabulator nach rechts.

Wir haben gesehen, dass diese Listen mit der Kombination von zwei Tags funktionieren. Wenn Sie wollen – das sieht allerdings eher unschön aus –, können Sie die *<LI>*- *<DT>*- und *<DD>*-Tags auch miteinander kombinieren.

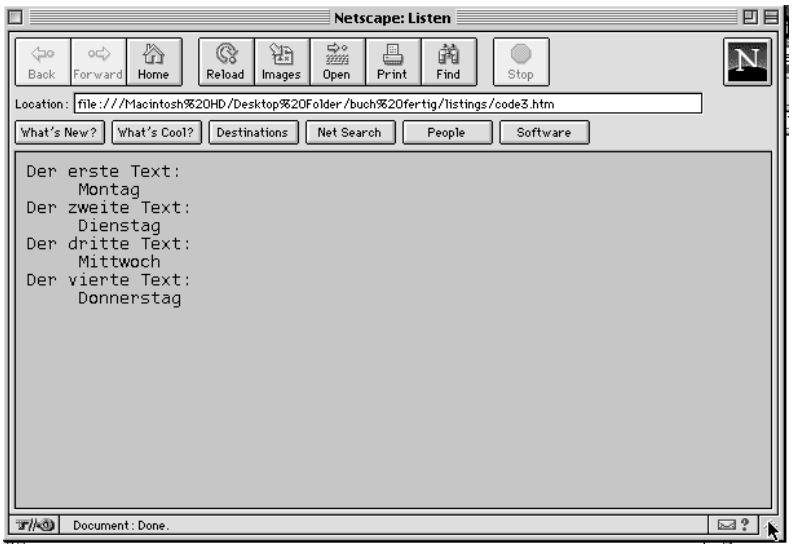

Bild 3.3: Hier stellt der »Navigator« die Treppenliste dar.

Sie ahnen es schon – innerhalb der Listeninhalte können die Texte in der bereits gewohnten Art und Weise formatiert und verlinkt werden. Fetten Sie in einer Definitionsliste jeweils das Wort »Text«, und linken Sie die Wörter »erste, zweite, dritte, vierte« jeweils mit einer WWW-Site Ihrer Wahl.


```
   <DL>
A     <DT> Der <A HREF=http://www.yahoo.de>erste</A>
B        <B>Text</B>:
         <DD>Montag
      <DT> Der <A HREF=http://www.web.de>zweite</A>
         <B>Text</B>:
         <DD>Dienstag
      <DT> Der <A HREF=http://www.lycos.de>dritte</A>
         <B>Text</B>:
         <DD>Mittwoch
      <DT> Der <A HREF=http://www.aladin.de>vierte</A>
         <B>Text</B>:
         <DD>Donnerstag
   </DL>
```

```
A     <DT> Der <A HREF=http://www.yahoo.de>erste</A>
```

Hier habe ich auf vier deutsche Suchmaschinen verlinkt. Die Syntax ist schon alter Käse. Sie sehen: Listen lassen sich leicht wie Fließtext verlinken …; soll auch ein Aufzählungspunkt mitverlinkt werden, muss der Tag logischerweise vor dem Listen-Tag stehen. Experimentieren Sie damit.

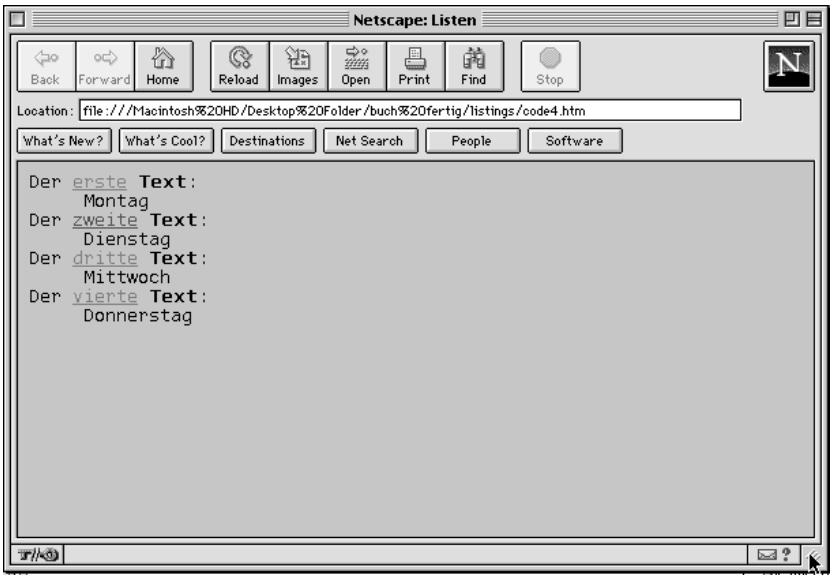

Bild 3.4: So einfach lässt es sich in Listen verlinken.

```
B    <B>Text</B>:
```

Es wären hier auch Font-Befehle denkbar. Allerdings ist es im Interesse der Browsersicherheit empfehlenswert, lieber mit einem Basefont zu arbeiten.

Wenn Ihnen die Aufzählungspunkte noch nicht differenziert genug sind, dann haben Netscape und Microsoft in ihren Browsern noch Untertags vorgesehen, die hier weiterhelfen. Die ersten beiden beziehen sich auf eine <OL>:

```
<OL TYPE=1 START=3>
```

Das heißt, dass bei einer Ordered List arabische Zahlen genommen werden, die allerdings erst bei der Zahl 3 mit der Zählung beginnen. Damit ist aber noch nicht Schluss. TYPE sieht noch folgende weitere Wahlmöglichkeiten vor:

```
TYPE=A    (große Buchstaben)
TYPE=a    (kleine Buchstaben)
TYPE=I    (große römische Zahlen)
TYPE=i    (kleine römische Zahlen)
```

Werden diese Untertags mit START kombiniert, so muss auch weiterhin eine Ziffer angegeben werden, um mit einer höheren Aufzählung zu starten. Also zum Beispiel START=2, um mit einer römischen »II« loszulegen.

Auch für den <*UL*>-Tag sehen die Browser von Microsoft und Netzcape einen Untertag vor, der hier noch weiter differenziert.

`TYPE=disc` (ausgefüllter Punkt)
`TYPE=circle` (Kreis)
`TYPE=square` (Viereck)

Der Standard für diese Browser ist die Zählweise ab »1.« in Zahlen und die Darstellung mit einem ausgefüllten Punkt. Für die anderen Browser, die diese Untertags nicht beherrschen, gilt das auch.

3.2 Tabellen

In vielen Fällen ist der Einsatz von Tabellen sehr praktisch. Auch in unserem Beispiel wäre es vielleicht schön, dieses Beispiel in der Form einer Tabelle abzuarbeiten.

Auch bei Tabellen hat HTML einen Tag dafür reserviert, dem Browser zu sagen, dass sich jetzt ein »Table« (= Tabelle) öffnet bzw. schließt. Der Tag dafür lautet:

```
<TABLE BORDER=1></TABLE>
```

Die BORDER-Angabe brauchen wir, um der Tabelle einen sichtbaren Rand von einem Pixel Breite zu geben. BORDER=2 würde den Pixelrand auf eine Breite von zwei Pixeln erhöhen usw.

Zwischen diesen Anfangs- und Schluss-Tags muss dem Browser jetzt auch gesagt werden, wie viele Zellen die Tabelle haben soll. HTML löst das in einer ähnlichen Art und Weise, wie jeder von uns Zeilen in einem Buch liest. Zuerst wird die Zeile eröffnet, hier nennt man das eine »Row« (Reihe), dann wird definiert, welche Zellen mit »Daten« vorhanden sind.

Der Tag für eine Tabellenreihe lautet:

```
<TR></TR>
```

Der Tag für eine Tabellenzelle, die immer (!) in einer Tabellenreihe sein muss, lautet:

```
<TD></TD>
```

Wenn wir diese drei Tags nun noch richtig ineinander verschachteln, haben wir schon eine Tabelle programmiert. Die richtige Syntax dafür lautet:

```
<TABLE BORDER=1>
   <TR>
      <TD>
      </TD>
   </TR>
</TABLE>
```

Zuerst wird also dem Browser signalisiert, dass sich ein Table öffnet, dann erhält er den Tag, der in dieser Tabelle eine Reihe öffnet, die eine einzige Datenzelle beinhaltet, und dann schließt zuerst der Daten-Tag, dann der Row-Tag und schließlich der generelle *<Table>*-Tag.

Jede Tabelle kann mehr als eine Reihe und darin mehr als eine Zelle enthalten. So sieht zum Beispiel die Syntax für eine einreihige Tabelle mit zwei Datenzellen aus:

```
<TABLE BORDER=1>
   <TR>
      <TD>
      </TD>

      <TD>
      </TD>
   </TR>
</TABLE>
```

Wir haben bisher noch nichts in die Zellen hineingeschrieben. Aber keine Bange. Text reagiert innerhalb der Zellen ähnlich wie außerhalb. Nutzen wir jetzt die Gelegenheit dazu und setzen unseren Beispieltext aus dem letzten Kapitel um. Bitte fügen Sie so viele Table-Rows ein, bis Sie diesen Text eingeordnet haben. Es genügt, wenn Sie pro Row eine Tabellenzelle eröffnen.


```
A   <TABLE BORDER=1>
B      <TR>
C         <TD>
          <B>Inhaltsverzeichnis</B><BR>
          </TD>
D      </TR>

       <TR>
          <TD>
          1. Der erste Text: Montag
          </TD>
       </TR>
       <TR>
          <TD>
```

```
         2. Der zweite Text: Dienstag
         </TD>
      </TR>

      <TR>
         <TD>
         3. Der dritte Text: Mittwoch
         </TD>
      </TR>

      <TR>
         <TD>
         4. Der vierte Text: Donnerstag
         </TD>
      </TR>
E  </TABLE>
A  <TABLE BORDER=1>
```

Ich habe hier zwischen den Tags *</TR>* und *<TR>* immer eine Zeile frei gelassen, damit Sie den Anfang und das Ende einer Tabellenreihe besser sehen. Das ist schon sauberer Code, aber es wäre noch besser, wenn Sie sich diesen Zeilendurchschuß sparen. Dann sind Sie perfekt.

Die Tabelle wird eröffnet. Sie hat einen Pixelrand von 1 eingestellt. Das entspricht nicht ganz dem eines Pixels, da für den 3-D-Effekt der Tabelle und andere Feineinstellungen noch ein wenig Platz verbraucht wird. Auf einem Computer mit dem Betriebssystem MAC/OS sehen diese Ränder zum Beispiel schlanker aus als auf einem, der unter WINDOWS läuft.

```
B  <TR>
```

Die erste Reihe wird eröffnet. In dieser Reihe soll die Zelle mit dem Wort »Inhaltsverzeichnis« stehen.

```
C        <TD>
         <B>Inhaltsverzeichnis</B><BR>
         </TD>
```

Die Zelle mit dem Wort »Inhaltsverzeichnis« habe ich noch mit Formatier-Tags bereichert. Das müssen Sie nicht, aber Sie sehen so, dass Text sich wirklich wie außerhalb der Tabellen gestalten lässt.

```
D     </TR>

      <TR>
```

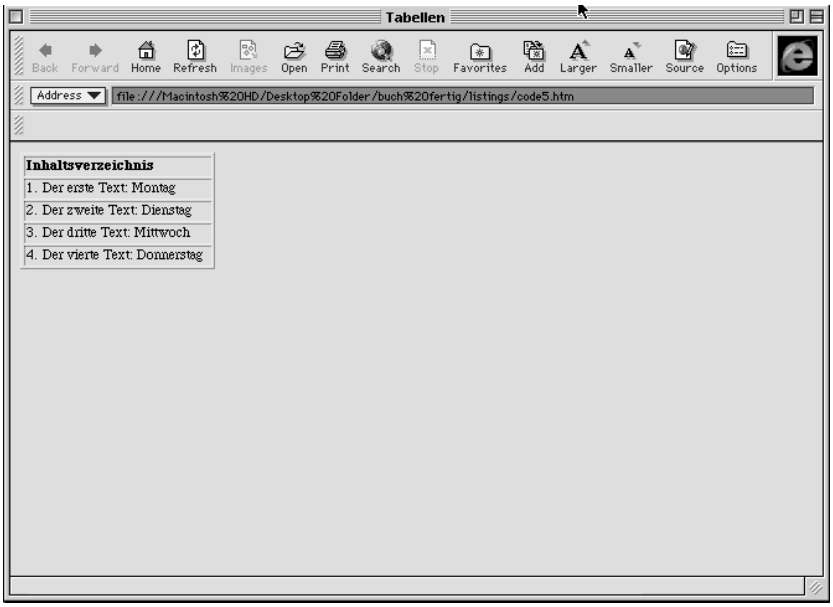

Bild 3.5: Jetzt haben wir die Tagesangaben in eine Tabelle eingebaut.

Die erste Reihe schließt, und die zweite Reihe öffnet sich, in der die Tabellenzelle mit der ASCII-Kette »1. Der erste Text: Montag« steht. Alle weiteren Reihen verfahren genau nach diesem Prinzip.

```
E   </TABLE>
```

Die Tabelle wird nach dem Schließen der letzten Reihe ebenfalls geschlossen.

Ein paar wichtige Anmerkungen zum Verhalten von Tabellen gibt es hier zu machen. Tabellen werden sich standardmäßig immer die Größe suchen, die ihnen der Inhalt vorgibt. Längere Sätze dehnen deshalb die Tabellen gleichmäßig. Sollte in der nächsten Reihe ein kürzerer Satz stehen, dominiert aber die längste Zeichenkette in der gesamten Spalte und dehnt auch diese Zelle in der ähnlichen Breite. Testen Sie den obigen Code einfach einmal mit verschieden langen Sätzen; dann sehen Sie schnell, wie dynamisch Tabellen auf Textlängen reagieren.

Sie werden auch feststellen, dass der längste Text hier immer auch vorgibt, wie sich der Rest der Texte in den anderen Zellen verhält. Wenn Sie mit unterschiedlichen Textlängen arbeiten, werden Sie sehen, dass die kürzeren Textabschnitte zentriert mit linksbündigem Zeilenanfang dargestellt werden.

Sollten Sie zufällig vergessen, den *<TABLE>*-Tag zu schließen, dann werden Sie auf den Netscape-Browsern plötzlich überhaupt nichts mehr auf dem

Bildschirm erkennen. Fehlerhafte Tabellen gehören zu den schlimmen Unfällen, die es in HTML gibt. Also Achtung!

Leider ist anzumerken, dass Netscape-Browser ziemlich unsaubere Darstellungen von Tabellen erzeugen. Da können Sie leicht wahnsinnig werden, wenn es um pixelgenaue Programmierungen geht. Profis beginnen deshalb, den Source-Code auf einem Netscape-Browser zu entwickeln, und schenken sich beim Aufrufen des »Internet Explorer« von Microsoft einen Kaffee ein. Der funktioniert meistens gleich mit. Umgekehrt soll es schon Hysterien und Umschulungen zum Töpfermeister gegeben haben.

Sollten Sie die Tabelle unsichtbar machen wollen, dann genügt es, den Untertag BORDER auf »0« zu stellen oder diesen Untertag ganz aus dem TABLE-Tag zu lassen. Allerdings empfehle ich Ihnen immer, diesen Schritt erst zu tun, wenn Sie mit der Tabelle soweit fertig sind.

Hier sehen Sie bei diesem etwas langwierigen Beispiel wahrscheinlich am deutlichsten, warum sauberes Code-Layout so wichtig ist. Auf diese Weise wird Ihnen nämlich sehr schnell klar, wo Sie hier noch einen Tag übersehen haben könnten. Gerade bei vielen Tabellenzellen ist hier Sorgfalt Gold wert. Eine Fehlersuche im unsauber gelayouteten Code kann Sie hier Stunden kosten.

Achten Sie auf diesen Aspekt vor allem in den folgenden Kapiteln.

Es gibt, vor allem für den »Internet Explorer« von Microsoft (Näheres können Sie dazu z.B. unter »http://www.microsoft.com/germany« erfahren), noch eine Reihe von sehr speziellen Tags, die sich um das Aussehen der Umrandungen und ähnliche eher marginale Dinge kümmern. Diese Tags sollen uns zum Teil erst in Kapitel 4 interessieren.

Ein Tag hilft uns allerdings gerade in unserem Beispiel weiter. Die Fettung von »Inhaltsverzeichnis« haben wir bisher von Hand per eigenem Tag erzeugt. Es gibt aber einen einfachen Befehl, der diese Funktion automatisch übernimmt. Mit

```
<TH></TH>
```

wird eine Zelle zur Überschrift, zum Header, erklärt.

Wenn Sie diesen Befehl statt der *<TD>*-Tags in der ersten Reihe verwenden, wird »Inhaltsverzeichnis« automatisch gefettet und zentriert.

3.3 Verschachtelte Tabellen

Wir haben in unserem Beispiel durch den Einsatz einer Tabelle nun bereits Ordnung geschaffen, aber oft gibt es den Fall, dass Tabellen innerhalb einer Zelle noch einmal Untertabellen benötigen. Wenn wir unser Beispiel noch einmal hernehmen, möchten Sie vielleicht bei einer Angabe der Texte auch die verschiedenen Autoren mitangeben. In diesem Fall benötigen Sie einen Trick, um hier diese Namen ebenfalls in eine Tabelle zu bekommen.

Die Lösung heißt TABLE im TABLE.

Solche verschachtelten Tabellen sind leicht zu erstellen, wenn Sie sich vergegenwärtigen, dass Sie dazu immer zuerst die oberste Ebene programmieren müssen, in eine von deren Datenzellen dann die nächste Ebene eingefügt wird. Also:

```
<TABLE BORDER=1>
   <TR>
      <TD>
         <TABLE BORDER=1>
            <TR>
               <TD>
               </TD>
            </TR>
         <TABLE>
      </TD>

      <TD>
      </TD>
   </TR>
</TABLE>
```

Auch hier werden Sie merken, dass der Zelleninhalt die Breite und Höhe der verschachtelten Tabellen beeinflusst. Hier sollten Sie auf den Gedanken kommen, der ersten Tabellenebene einen deutlich höheren BORDER-Wert zuzuweisen, damit der User später auch sehen kann, dass die zweite Ebene eine Untertabelle darstellt. Es gilt als Regel: Der größte BORDER-Wert sollte dabei die anderen Tabellen umschließen.

So entsteht eine Tabelle in einer Tabelle, indem wir hier in die erste Zelle der ersten Tabelle eine weitere Tabelle eingefügt haben. Sie könnten jetzt auch noch eine dritte Tabellenebene anlegen, allerdings zeigt die Erfahrung im

Webdesign, dass Sie solche extremen Verschachtelungen so gut wie nie brauchen werden. Trotzdem ist es gut zu wissen: Es geht.

Eine mögliche Anwendung dieser Verschachtelungen wollen wir uns jetzt genauer zur Übung ansehen.

Nehmen Sie wieder unseren Übungstext, und schreiben Sie unter den Wochentag Mittwoch noch zweispaltig vier beliebige Autorennamen dazu. Zum Beispiel Goethe, Schiller, Valentin und Blüm.


```
A   <TABLE BORDER=4>
      <TR>
B       <TH>
        Inhaltsverzeichnis<BR>
        </TH>
      </TR>

      <TR>
        <TD>
        1. Der erste Text: Montag
        </TD>
      </TR>

      <TR>
        <TD>
        2. Der zweite Text: Dienstag
        </TD>
      </TR>

      <TR>
        <TD>
C       3. Der dritte Text: Mittwoch<BR>
D       <TABLE BORDER=1>
          <TR>
E           <TD>
            Goethe<BR>
            </TD>

F           <TD>
            Valentin<BR>
            </TD>
          </TR>

G         <TR>
```

```
                <TD>
                Schiller<BR>
                </TD>

                <TD>
                Bl&uuml;m<BR>
                </TD>
             </TR>
H          </TABLE>
          </TD>
       </TR>

       <TR>
          <TD>
          4. Der vierte Text: Donnerstag
          </TD>
       </TR>
I   </TABLE>
```

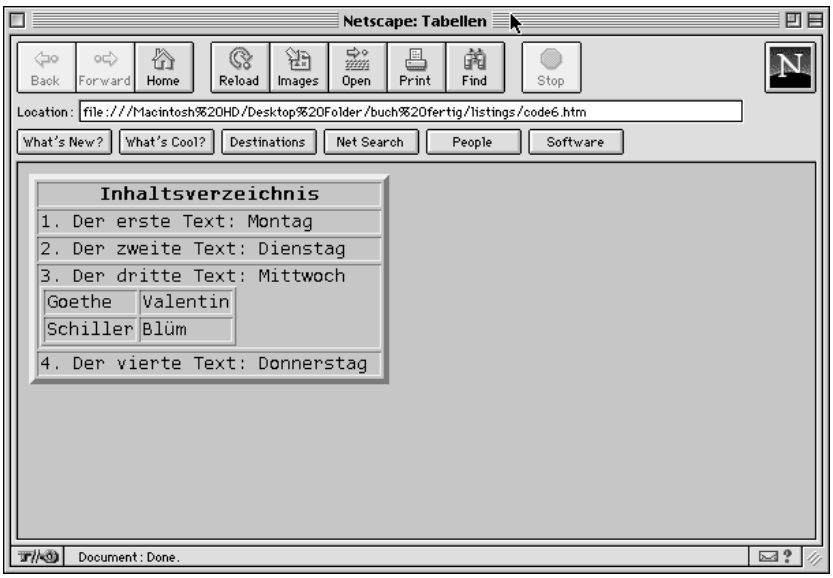

Bild 3.6: Die verschiedenen BORDER-Stärken haben keine Abstufung im Aussehen erzeugt.

```
A   <TABLE BORDER=4>
       <TR>
```

Wir eröffnen die erste Tabelle und geben ihrem Rand im Gegensatz zur folgenden Tabelle eine Pixelbreite von 4, damit sich der Tabellenrand hier deutlich von der darin verschachtelten Tabelle abhebt.

```
B   <TH>
```

Wie schon im letzten Kapitel angedeutet, erzeugen wir die Formatierung der Überschrift durch diesen Tag.

```
C   3. Der dritte Text: Mittwoch<BR>
```

Die Zelle, in der die zweite Tabelle erzeugt werden soll, beginnt in der üblichen Art und Weise mit dem Text, wechselt mit einem *
*-Tag die Zeile und kann nun auch problemlos die zweite Tabelle aufnehmen.

```
D   <TABLE BORDER=1>
      <TR>
```

Mit einer Breite von 1 am Rand eröffnen wir die zweite Tabelle und verfahren in der üblichen Art und Weise, indem wir zuerst eine Reihe eröffnen, in die dann die Zellen eingefügt werden. Aber wie wir schon gesehen haben: Der BORDER-Wert wird von der weiter oben eingestellten Tabelle deutlich abgehoben, da hier ein kleinerer Wert die Tabelle auch in der Hierarchie darunter erscheinen lässt.

```
E   <TD>
    Goethe
```

Da wir die Autorennamen zweispaltig einfügen wollen, fügen wir nun die erste Zelle der ersten Spalte ein. Dann schließen wir diese Zelle wieder.

```
F   <TD>
    Valentin<BR>
    </TD>
 </TR>
```

Jetzt öffnen wir die zweite Zelle in der ersten Spalte, schreiben hier den betreffenden Autoren hinein und schließen die Zelle wieder. Danach schließen wir auch die erste Reihe der Tabelle.

```
G             <TR>
                <TD>
                Schiller<BR>
                </TD>
```

Wir eröffnen die zweite Reihe und darin die erste Zelle. Das Verfahren läuft hier parallel zur vorherigen Reihe.

```
H        </TABLE>
```

Nach diesen vier Zelleneinträgen schließen wir diese Tabelle ab und fahren wie gewohnt in der ersten Tabelle weiter, die wir noch offen haben.

Solche Fälle sind nicht sehr häufig anzutreffen, wenn Sie sich auf eine strukturierte Form der Datendarstellung festlegen, aber Sie werden im nächsten Kapitel sehen, wie praktisch solche Verschachtelungen sein können, um ganz spezielle Fälle in der Darstellung auf HTML-Seiten zu lösen.

Wenn Sie für sich selbst einmal versuchen, Tabellen anzulegen, die in einer Reihe z.B. vier und in der anderen Reihe vielleicht sechs Zellen enthalten sollen, dann werden Sie schnell merken, dass irgend etwas nicht stimmt. Die meisten Browser können trotz einer sauber abgeschlossenen Reihe diese Tabelle nicht so darstellen, wie Sie das gerne hätten. Immer wieder gerät die Position der Zelleninhalte durcheinander, und es entstehen die merkwürdigsten Effekte.

Einfache Abhilfe schafft der Trick, bei jeder neuen Zellenzahl eine neue unsichtbare Tabelle anzulegen. So können Sie auch komplexe Änderungen der Daten-Cluster leicht darstellen. Ein einfaches Beispiel wäre hier (zur Verdeutlichung habe ich die BORDER-Stärken eingebaut; probieren Sie dieses Beispiel mit einem BORDER-Wert von 0):

```
<TABLE BORDER=1>
   <TR>
      <TD>
      1 Zelle
      </TD>
   </TR>
</TABLE>

<TABLE BORDER=4>
   <TR>
      <TD>
      1 Zelle
      </TD>
      <TD>
      2 Zellen
      </TD>
   </TR>
</TABLE>
```

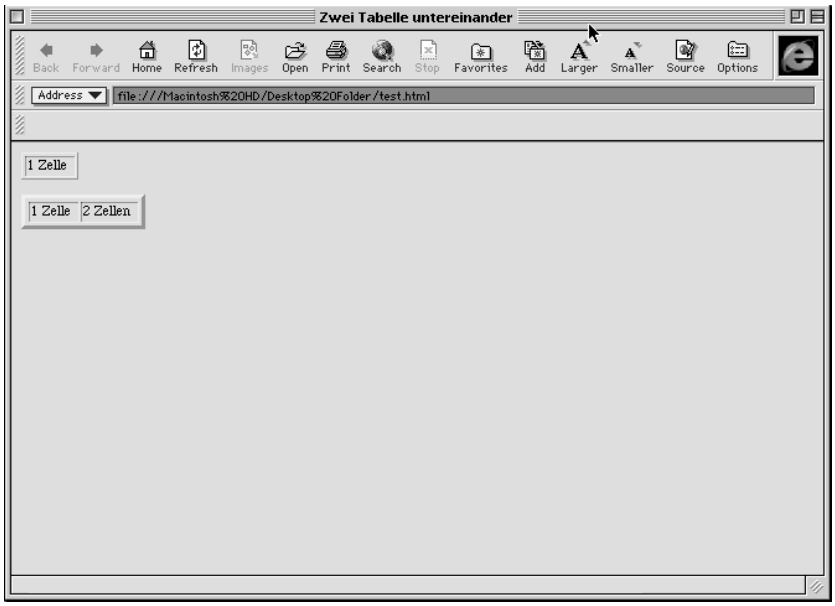

Bild 3.7: Jede Reihe kann so eine in sich geschlossene Tabelle darstellen.

Kein Problem für einen Browser. Theoretisch können Sie so unendlich viele Tabellen untereinander stellen. Aber wenn Sie die Tabellenränder nun wieder mit dem *BORDER*-Tag einschalten, werden Sie sehen, wie unschön die Ränder durch den neuen Tabellenbeginn doppelt dargestellt werden.

HTML hat deshalb eine andere Variante zu bieten, die aber Konzentration in der Programmierung verlangt, vor allem dann, wenn man diese Tags auch noch untereinander mischt. Diese Tags sollen die Tabellenzellen dann in ihrer Größe strecken, wenn in einer anderen Tabellenreihe mehr Zellen vorkommen.

Der Tag, um eine Zelle in ihrer Breite zu verdoppeln, lautet:

```
<TD COLSPAN=2></TD>
```

Der Tag, um eine Zelle in ihrer Höhe zu verdoppeln, lautet:

```
<TD ROWSPAN=2></TD>
```

Diese Tags sind dazu da, fehlende Zellen, die durch eine geringere Zellenzahl in einer Reihe entstehen, auszugleichen. Das ist am Anfang ein wenig heikel und fehleranfällig in der Programmierung, aber wenn man sich einmal in Ruhe eine solche Tabelle aufbaut, wird das Prinzip schnell klar.

Wir haben uns vorgenommen, eine Tabelle zu schreiben, die in der ersten Reihe eine und in der zweiten Reihe zwei Zellen besitzt. Wenn Sie so eine

Aufgabe vor sich haben, dann beginnen Sie am besten immer mit der Reihe, die die meisten Zellen besitzt, also hier mit der zweiten:

```
<TABLE BORDER=1>
   <TR>
      <TD>
      </TD>

      <TD>
      </TD>
   </TR>
</TABLE>
```

Jetzt fügen wir die erste Reihe ein:

```
<TABLE BORDER=1>
   <TR>
      <TD COLSPAN=2>
      Hier läuft die Zelle auf der ganzen
      Tabellenbreite
      </TD>
   </TR>

   <TR>
      <TD>
      Eine Zelle
      </TD>

      <TD>
      Zwei Zellen
      </TD>
   </TR>
</TABLE>
```

Sie sehen: Obwohl hier eine Zelle in der ersten Reihe angelegt wird, sagt der *COLSPAN*-Untertag dem Browser durch die Zahl 2, dass diese Zelle die Länge der zwei Zellen in der zweiten Reihe besitzt. Auf diese Weise kann hier nichts schiefgehen.

Diesen Fall können wir auch in der Senkrechten durchspielen. Wenn Sie eine Tabelle mit drei Reihen haben, in denen immer drei Zellen vorkommen, ist das für Sie inzwischen ein Standardfall.

Wir möchten aber noch, dass die Zelle eins der Reihe eins sich über die Zelle eins der Reihe zwei legt. Beginnen wir wieder mit dem leichtesten Fall, der Reihe drei, in der sich keine Änderung ergibt.

Bild 3.8: Die eine Zelle der ersten Reihe dehnt sich mühelos aus.

```
<TABLE BORDER=1>
   <TR>
      <TD>
      </TD>

      <TD>
      </TD>

      <TD>
      </TD>
   <TR>
</TABLE>
```

Jetzt wollen wir dem Browser sagen, dass sich die ersten Zellen der Reihen eins und zwei überlagern sollen.

Dazu geben wir zuerst die Reihe ein, deren Zelle eins später dominieren soll, also die Reihe eins.

```
<TABLE BORDER=1>
   <TR>
      <TD ROWSPAN=2>
      </TD>

      <TD>
```

```
      </TD>

      <TD>
      </TD>
   </TR>
   <TR>
      <TD>
      </TD>

      <TD>
      </TD>

      <TD>
      </TD>
   <TR>
</TABLE>
```

In dieser Reihe hat sich bis auf den Rowspan-Befehl noch nichts Spannendes ereignet. Jetzt steht es allerdings an, die zweite Reihe zu schreiben, deren erste Zelle ja verdeckt sein soll.

Und die Syntax dazu ist denkbar logisch: Da die erste Zelle in der ersten Reihe hier schon Platz genommen hat, wird diese Zelle in der zweiten Reihe einfach ausgespart. Der Code sieht also aus wie folgt:

```
<TABLE BORDER=1>
   <TR>
      <TD ROWSPAN=2>
      Erste Reihe, erste(!) Zelle
      </TD>

      <TD>
      Erste Reihe, zweite Zelle
      </TD>

      <TD>
      Erste Reihe, dritte Zelle
      </TD>
   </TR>

   <TR>
      <TD>
      Zweite Reihe, zweite(!) Zelle
      </TD>
```

```
      <TD>
      Zweite Reihe, dritte Zelle
      </TD>
   </TR>

   <TR>
      <TD>
      Dritte Reihe, erste Zelle
      </TD>

      <TD>
      Dritte Reihe, zweite Zelle
      </TD>

      <TD>
      Dritte Reihe, dritte Zelle
      </TD>
   <TR>
</TABLE>
```

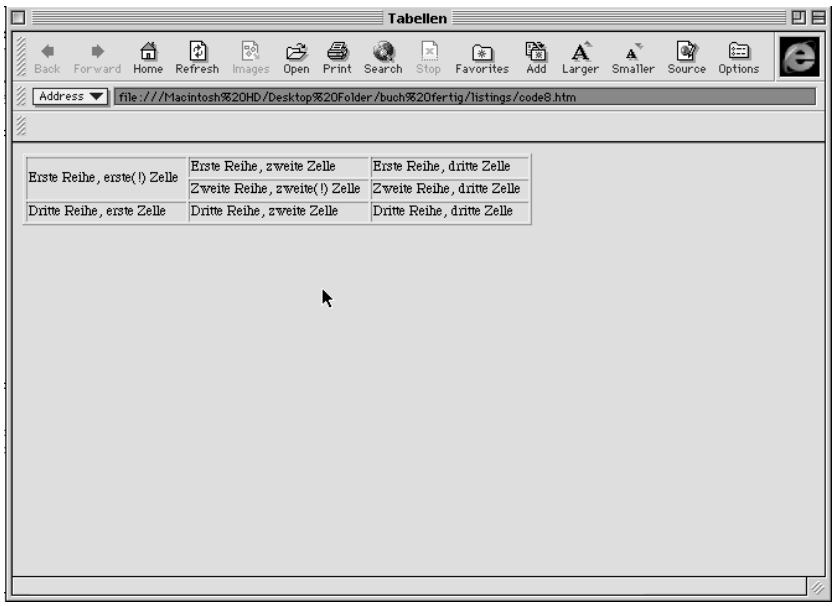

Bild 3.9: Schon haben wir auch senkrecht eine Zelle überbrückt.

Hier ist einfach stufenweises Programmieren angesagt. Das spart Hirnschmalz. Trotzdem oder gerade deshalb wollen wir uns jetzt einmal so richtig plagen und alle diese Verschachtelungen in einem einzigen File ausprobieren.

Nehmen Sie wieder unser Textbeispiel, und fügen Sie die Überschrift links vom gesamten Zellentext in eine eigene Spalte ein. Der Donnerstag sollte zudem noch eine rechte Zelle haben, in die Sie »danach: Pause« als Text einfügen. Eine kleine Sonderaufgabe gilt es auch noch zu lösen: Bitte verlinken Sie die Wochentage auf die Files »01.htm«, »02.htm«, »03.htm« und »04.htm«.


```
A   <TABLE BORDER=2>

      <TR>
B        <TD ROWSPAN=4>
         <B>Inhaltsverzeichnis</B>
         </TD>

C        <TD COLSPAN=2>
      1. Der erste Text: <A HREF=01.htm>Montag</A>
         </TD>
      </TR>

      <TR>
         <TD COLSPAN=2>
      2. Der zweite Text: <A HREF=02.htm>Dienstag</A>
         </TD>
      </TR>

      <TR>
         <TD COLSPAN=2>
     3. Der dritte Text: <A HREF=03.htm>Mittwoch</A><BR>
D         <TABLE BORDER=1>
            <TR>
               <TD>
               Goethe<BR>
               </TD>

               <TD>
               Valentin<BR>
               </TD>
            </TR>
```

```
            <TR>
               <TD>
               Goethe<BR>
               </TD>

               <TD>
               Valentin<BR>
               </TD>
            </TR>
         </TABLE>
         </TD>
      </TR>

      <TR>
         <TD>
   4. Der vierte Text: <A HREF=04.htm>Donnerstag</A>
         </TD>

E        <TD>
         Danach: Pause
         </TD>
      </TR>
   </TABLE>
```

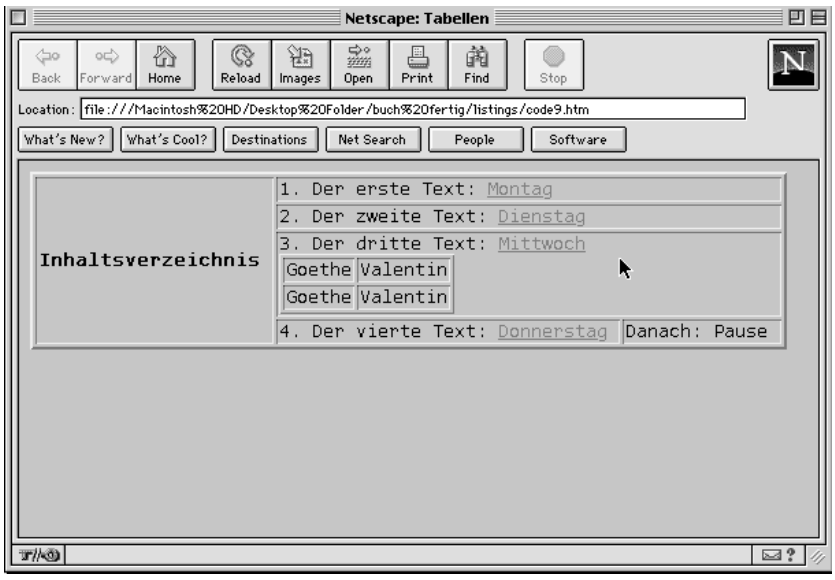

Bild 3.10: Na? War's schwer?

```
A   <TABLE BORDER=2>
```

Wir beginnen wieder mit einer Tabelle, deren Rand zwei Pixel stark ist:

```
      <TR>
B        <TD ROWSPAN=4>
         <B>Inhaltsverzeichnis</B>
         </TD>
```

Hier fügen wir die Zelle ein, die den Überschriftstext enthalten soll. Da sich diese Zelle auf alle vier Reihen der oberen Tabelle beziehen soll, besitzt der ROWSPAN-Tag den Wert 4.

```
C        <TD COLSPAN=2>
         1. Der erste Text: <A HREF=01.htm>Montag</A>
         </TD>
```

Die nächste Klippe: Wir setzen den COLSPAN-Wert jetzt auf 2, da bis auf die letzte Reihe alle Reihen der oberen Tabelle eigentlich nur eine eigene Zelle haben. Und genau wegen dieses Rowspan-Befehls können wir in der Folge die erste Zelle der Reihen weglassen und gleich mit der zweiten beginnen, die den bereits gewohnten Text enthält. Nicht vergessen: Verlinken Sie die Wochentage wie verlangt.

```
D        <TABLE BORDER=1>
```

Und wieder beginnen wir damit, die Autorennamen in einer eigenen Tabelle zu schachteln. Der BORDER-Wert liegt wieder bei 1, um eine Untertabelle zu signalisieren.

```
E        <TD>
         Danach: Pause
         </TD>
```

Jetzt fügen wir die Zelle mit dem neuen Text einfach dazu. In dieser vierten Reihe lassen wird deshalb den COLSPAN-Befehl einfach weg.

Uff, geschafft. ;-))

3.4 Tabellen als Spaltensatz

Sie haben sicher schon den Verdacht, dass das Aufziehen von Tabellen auch dazu dienen könnte, Spaltensatz auf den Webseiten zu simulieren. Mit diesem Verdacht liegen Sie richtig. Webdesigner nutzen die differenzierten Möglichkeiten solcher auch ineinander verschachtelter Tabellen, um selbst »unmögliche« Layouts umzusetzen. Als Webdesigner besitzen Sie sehr schnell ein inneres Auge, das eine umzusetzende Optik in senkrechte und

waagerechte Tabellenraster trennt und Ihnen eine Ahnung gibt, wie Sie die Tabelle aufziehen müssen, um diesen Anforderungen zu entsprechen. Nun werden Sie zu Recht sagen, dass es von den Tabellen der letzten Übung bis zu einem wirklichen Satzspiegel noch zwei Hinderungsgründe geben könnte. Zum einen müssen die »*TABLE*«-Tags unsichtbar sein. Aber das kennen wir bereits. Alle Spaltensatz-Layouts beginnen einfach mit einem *BORDER=0*-Tag oder lassen diese Angabe einfach weg (Ersteres ist im Hinblick auf all die verschiedenen Browser einfach sicherer).

Der zweite Einwand ist aber viel schwerwiegender: Wir haben durch die bisherigen Übungen ja bereits gesehen, dass Tabellen je nach Textinhalt verschiedene Zellengrößen annehmen können. Dieses eigentlich sehr nützliche Feature von HTML stört uns, wenn wir Spaltensatz über diese Tags aufbauen wollen. Hier darf sich dann nichts mehr ändern (mehr dazu in den folgenden Kapiteln).

Browser wie Netscape »Navigator« oder der Microsoft »Explorer« sehen aber hier eine einfache Möglichkeit vor, die Spaltenhöhe und -breite vorzugeben. Folgende Tabelle:

```
<TABLE BORDER=0 WIDTH=200 HEIGHT=300>
   <TR>
      <TD>
      </TD>
   </TR>
</TABLE>
```

besitzt eine Höhe von 300 Pixeln und eine Breite von 200 Pixeln. Wir kennen diese Tags bereits von den *<HR>*-Tags, also eher alte Bekannte, die uns hier wieder weiterhelfen. Und wie auch beim ersten Mal können diese Höhen- und Breitenangaben in Prozentzahlen des Bildschirms ausgedrückt werden, was vor allem Webdesignern, die mit dem Goldenen Schnitt arbeiten möchten, sehr helfen kann.

Wenn nur eine der Zellen eine gewisse Höhe oder Breite haben soll, dann müssen diese beiden Untertags in einen Zellen-Tag mit hineingeschrieben werden. Eine Tabellenzelle, die also 20 Prozent des Bildschirms breit sein soll und 30 Pixel hoch, wird mit folgender Syntax fixiert:

```
<TABLE BORDER=0>
   <TR>
      <TD WIDTH=20% HEIGHT=30>
      </TD>
      <TD>
      </TD>
   </TR>
</TABLE>
```

Durch diese Fixierung werden wir mit zwei Effekten konfrontiert, die durch zu kleine Fenster oder zu klein eingestellte Tabellenhöhen entstehen können. Reicht die Breite einer Tabelle jetzt über die eines Fensters hinaus, so addiert der Browser hier automatisch einen waagerechten Scrollbalken, um sich im Fenster bis an den rechten Rand bewegen zu können. Das gilt nur für absolute Angaben, denn bei Prozentangaben, die NIE (!) über 100 Prozent hinausgehen, wird das Fenster ja immer groß genug sein. Zum anderen kann es passieren, dass die angegebene HEIGHT-Pixelzahl eigentlich nicht ausreicht, um sich wirklich genügend Platz für den Zelleninhalt zu reservieren. In diesem Fall bricht der Browser die Höheneinstellung selbständig auf, oder er schluckt Text. Erfahrene Webdesigner werden deshalb nur in einem bestimmten Fall die Höhe wirklich angeben: wenn sie Bilder in eine Zelle kopieren, aber dazu kommen wir später noch.

Wir haben im letzten Kapitel einen Index aufgebaut, der vier Links besitzt, und genau diese vier Files mit verschiedenen Spaltenlayouts wollen wir jetzt herstellen. Nehmen wir einfach an, dass Sie diesen Index benötigt haben, um auf Ihrer Website eine Sammlung von Artikeln zu verlinken, die in ihrem jeweiligen Layout belassen werden sollen. Deshalb sollen die vier Texte auf vier verschiedenen Layoutseiten verteilt werden.

`01.htm`	Text links mit großem Rand rechts
`02.htm`	Text mittig mit Rand links und rechts
`03.htm`	Text rechts mit großem Rand links
`04.htm`	Überschriften weiter links als der Text

Bei all diesen vier Typen soll es zwei gemeinsame Vielfache geben: Der Text soll erst nach 10 Prozent des Bildschirms anfangen, und am Ende soll ein Link zurück zum Index existieren.

Spielen wir das File »01.htm« einmal durch (siehe Lösung am Ende des Kapitels).

Für alle diese Files gilt: Sie öffnen in einem HTML-File eine Tabelle.

```
<TABLE BORDER=0 WIDTH=95% HEIGHT=10%>
   <TR>
      <TD>
      <BR>
      </TD>
   </TR>
</TABLE>
```

Wie Sie sehen, dient diese Tabelle ohne einen Textinhalt dazu, 10 Prozent des Bildschirms zu bedecken und leer zu lassen. Das war es schon. So erzeugen Sie sicheren Leerraum und sorgen dafür, dass Ihr Layout nicht an den oberen Rand des Bildschirms geklatscht wird. Die Breite von 95 Prozent wählen wir leicht unter der tatsächlichen Breite des Bildschirms von logischerweise 100 Prozent. Manche Browser reagieren so besser. Genau aus dem gleichen Grund geben wir einen *
*-Tag in der Zelle an, weil wir bei manchen Browsern sonst keine Reaktion auf unsere Tabelle erreichen. Sie sind so programmiert, dass leere Tabellen keine Relevanz haben und deshalb nicht dargestellt werden.

Nach dem Layout werden wir dann einfach noch einen Link zurück zum Index bauen:

```
<A HREF=inhalt.htm>zurück</A>
```

In dieses Gerüst fügen wir jetzt jeweils die passenden Layouts ein und speichern das so gewonnene File unter dem passenden Namen ab.

01.htm Text links mit großem Rand rechts

```
<TABLE BORDER=0>
   <TR>
      <TD WIDTH=200>
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
      </TD>

      <TD>
      </TD>
   </TR>
</TABLE>
```

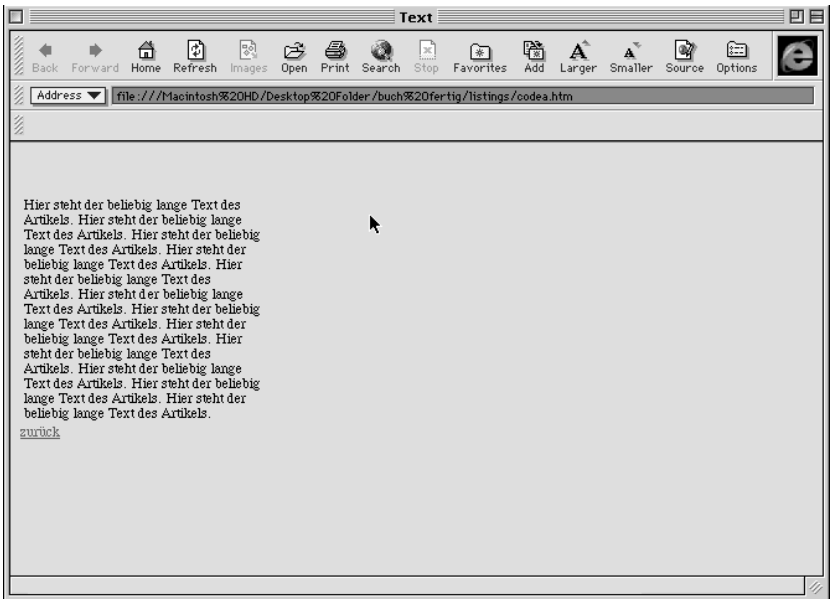

Bild 3.11: Endlich haben wir den Text in eine Spalte gegossen.

Hier habe ich zwei Tabellenzellen als Spalten aufgezogen, deren linke den Text beinhalten wird. Da ich eine Weite von 200 Pixeln zugeteilt habe, wird die rechte Spalte das restliche Fenster ausfüllen und sich je nach Fenstergröße dynamisch verhalten. Es fehlt eine Höhenangabe, weil hier ein Text zu sehen sein wird, dessen Ausdehnung ich durch die Features des Browsers nicht exakt sehen kann. Deshalb lasse ich die Tabelle nach unten hin ihre Tiefe selbst dynamisch generieren.

02.htm Text mittig mit Rand links und rechts

```
<TABLE BORDER=0>
   <TR>
      <TD WIDTH=30%>
         <BR>
      </TD>

      <TD WIDTH=200>
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
```

```
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
      </TD>

      <TD>
      </TD>
   </TR>
</TABLE>
```

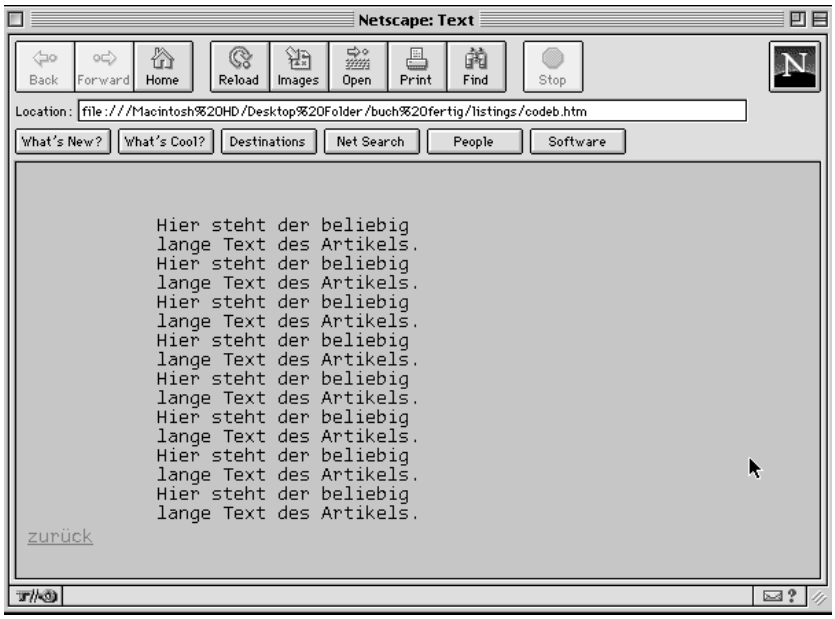

Bild 3.12: Jetzt rückt die Textspalte um ein Drittel der Seite nach rechts.

Auch hier lege ich eine Tabelle an, deren Textzelle eine Weite von 200 Pixeln erhält und sich die Tiefe je nach Textinhalt selbst dynamisch einstellt. Da wir links und rechts einen Rand haben wollten, arbeite ich eine dreispaltige Tabelle aus, deren linke und rechte Zelle leer bleibt. Allerdings muss ich hier einen Kompromiss schließen, denn nur die rechte Zelle gleicht sich selbständig an. Die linke braucht ebenfalls eine Weitenangabe (hier in 30 Prozent, also in einem Drittel des Bildschirms ausgedrückt) und einen *
*-Tag, damit die Zelle vom Browser nicht als leer und damit als nicht darzustellen definiert wird.

03.htm Text rechts mit großem Rand links

```
<TABLE BORDER=0>
   <TR>
```

```
      <TD WIDTH=400>
      <BR>
      </TD>

      <TD WIDTH=200>
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.
Hier steht der beliebig lange Text des Artikels.

      </TD>
   </TR>
</TABLE>
```

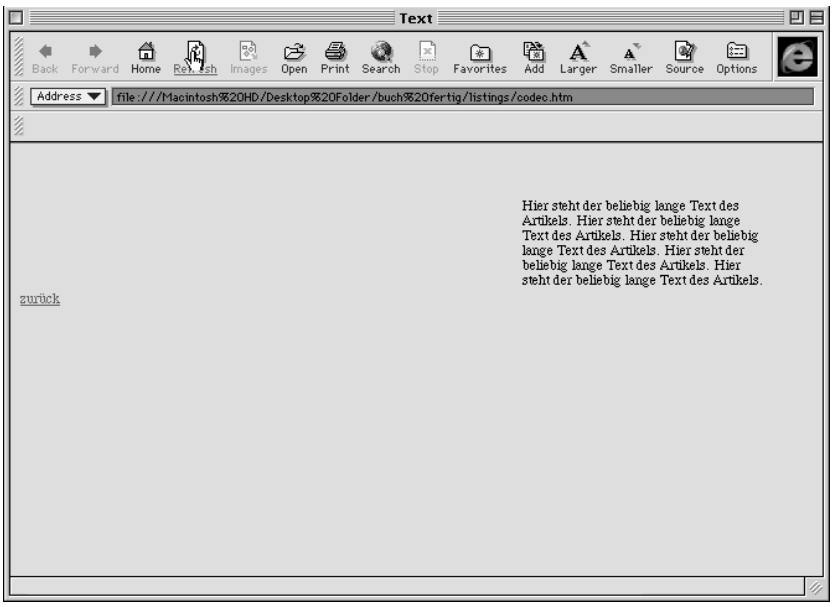

Bild 3.13: Hier steht der Text rechts im Bild.

Jetzt schaffen wir uns über eine leere Zelle, die 400 Pixel breit ist, Platz. Danach soll die rechte Zelle mit 200 Pixeln Breite erscheinen. 600 Pixel sind eine Standardbreite für einen 15"-Bildschirm, auf dem das Browserfenster ganz geöffnet ist. Eigentlich beträgt die maximale Breite hier 640 Pixel, aber das Browserfenster und ein eventuell auftauchender senkrechter Scrollbalken brauchen ja auch noch etwas Platz.

04.htm Überschriften weiter links als der Text

Fieserweise sollen Sie jetzt die spannendste Aufgabe lösen. Versuchen Sie, mit dem Wissen der letzten Kapitel eine Lösung zu finden, die die Überschriften größer darstellt als den Fließtext und weiter links im Spaltensatz ansetzt. Nutzen Sie dabei keine Listen oder Blockquotes. Gestalten Sie die Textspalte wieder 200 Pixel weit.


```
   <TABLE BORDER=0>
      <TR>
A        <TD COLSPAN=2>
   <FONT SIZE=5>Dies ist die &Uuml;berschrift</FONT>
         </TD>
      </TR>

      <TR>
B        <TD WIDTH=400>
         <BR>
         </TD>

C        <TD WIDTH=200>
   Hier steht der beliebig lange Text des Artikels.
         </TD>
      </TR>
   </TABLE>
```

```
A        <TD COLSPAN=2>
   <FONT SIZE=5>Dies ist die &Uuml;berschrift</FONT>
         </TD>
```

Nachdem wir in der üblichen Art und Weise eine Tabelle eröffnet haben, setzen wir die Überschriftenspalte ganz links an und lassen sie mit dem COLSPAN-Befehl bis an den rechten Rand der Tabelle laufen. Damit die Schrift noch ein wenig größer dargestellt wird, wählen wir die Fontgröße 5.

```
B        <TD WIDTH=400>
         <BR>
         </TD>
```

Hier ist der Knackpunkt des Ganzen: Wir eröffnen dann eine neue Reihe und bauen hier erst einmal eine leere Zelle ein, die den Abstand des Fließtextes erzeugt. Ich habe hier 400 Pixel Abstand vom linken Rand genommen. Sie können aber je nach Geschmack mehr oder weniger wählen.

```
C        <TD WIDTH=200>
   Hier steht der beliebig lange Text des Artikels.
         </TD>
```

Bild 3.14: Ungewöhnlich, aber möglich: springender Schriftsatz.

Daran schließt nun die rechte Tabellenzelle mit dem Text in einer Pixelbreite von 200 an. Fertig.

In diesem Kapitel haben wir nun vier verschiedene Layouttypen kennengelernt, die Sie alle nach Belieben modifizieren können. Denkbar sind ja auch mehrspaltige Texte mit verschieden breiten Abständen (= Leerspalten!) zwischen den Textspalten.

Bei mehrspaltigen Texten müssen Sie berücksichtigen, dass Sie diese Spalten nicht miteinander verketten können. Es ist also keine Automatik wie z.B. bei Quark Xpress und ähnlichen DTP-Programmen machbar, die den Text am Ende einer Spalte zu Beginn der nächsten Spalte weiterfließen lässt.

3.4.1 Textstand in den Tabellen

Wenn wir noch einmal zu unserem Beispiel zurückgehen, dann haben wir im Index ja für den Mittwoch einen besonderen Fall vorgesehen, in dem vier Autoren genannt werden. Würden wir also eine Tabelle mit vier verschieden langen Texten dieser vier Autoren aufziehen, sähe die Syntax logischerweise wie folgt aus:

```
<TABLE BORDER=0>
   <TR>
      <TD WIDTH=10%>
      <BR>
      </TD>

      <TD WIDTH=200>
Hier steht der 1. Text
      </TD>

      <TD WIDTH=10%>
      <BR>
      </TD>

      <TD WIDTH=200>
Hier steht der 2. Text
      </TD>
   </TR>

   <TR>
      <TD WIDTH=10%>
      <BR>
      </TD>

      <TD WIDTH=200>
Hier steht der 3. Text
      </TD>

      <TD WIDTH=10%>
      <BR>
      </TD>

      <TD WIDTH=200>
Hier steht der 4. Text
      </TD>
   </TR>
</TABLE>
```

Mit Zwischenspalten von 10 Prozent des Bildschirms an Breite habe ich tadellose vier Zellen für die Texte aufgebaut. Scheinbar alles in Ordnung. Wenn Sie jetzt aber probeweise verschieden lange Texte in die Zellen eingeben, dann stehen wir vor einem Problem, das alle Webdesigner zur Genüge kennen: HTML richtet den Zelleninhalt mittig in der Senkrechten ein.

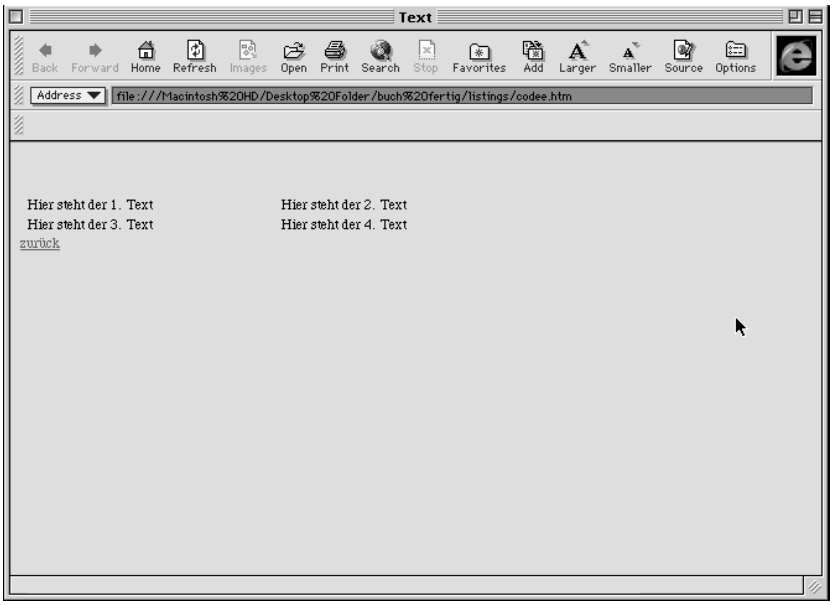

Bild 3.15: Hier haben wir Leerspalten eingefügt.

Was bei »richtigen« Tabellen absolut sinnvoll ist, stört hier im Fall des nachempfundenen Spaltensatzes. Netscape und Microsoft haben sich deshalb auf Untertags geeinigt, die den Text genauer in den Zellen plazieren lässt.

Zuerst schauen wir uns die Waagerechte an.

`ALIGN=left`	(Der Text wird linksbündig zur Zelle dargestellt)
`ALIGN=center`	(Der Text wird mittig in der Zelle dargestellt)
`ALIGN=right`	(Der Text wird rechtsbündig zur Zelle dargestellt)
`VALIGN=top`	(Der Text wird links oben in der Zelle begonnen)
`VALIGN=middle`	(Der Text wird in der Zelle zentriert)
`VALIGN=bottom`	(Der Text wird rechts unten in der Zelle ausgerichtet)

Der *<BLOCKQUOTE>*-Tag wäre zudem noch eine zusätzliche Hilfe, aber wir können uns diese Hilfskrücke mit zwei weiteren Untertags ersparen, wenn sich das Einrücken auf ganze Tabellen beziehen soll. Mit einer Einstellung von z.B.

```
<TABLE BORDER=0 CELLSPACING=5   CELLPADDING=2>
```

rückt der Inhalt innerhalb der Zellen jetzt fünf Pixel weg, und die Breite der Abstände zwischen (!) den Zellenumrandungen ist hier zwei Pixel.

Zugegebenermaßen werden im Spaltensatz diese letzten beiden Untertags selten benutzt. Allerdings sollten Sie CELLPADDING und CELLSPACING möglichst immer auf »0« oder »-1« stellen, um optische Anschlüsse sauber zu generieren, auch dann, wenn Sie eigentlich »nur normale« Tabellen erstellen wollen.

Es ist ratsam, die <*ALIGN*> bzw. /<*VALIGN*>-Tags jeweils pro Zelle einzeln einzugeben. Durch ein »ALIGN=left« im <*TABLE*>-Tag rückt die Tabelle sonst links des Textes, der nach der Tabelle folgt. Analog dazu verhält es sich mit »ALIGN=right« innerhalb des <*TABLE*>-Tags.

Nehmen Sie das eben vorgeführte Tabellenbeispiel mit den vier Textzellen, und richten Sie es wie folgt aus. Die erste Zelle lassen Sie in der Standardeinstellung. Die zweite Zelle lassen Sie mit zentriertem Text in der Mitte ausrichten, der Text der dritten Zelle soll linksbündig oben links beginnen, der der rechten unten rechts enden. Die Zellen sollten alle fünf Pixel voneinander entfernt sein und innerhalb der Zelle zwei Pixel Abstand zum Rand erzeugen. Ein Tip: Benutzen Sie verschieden lange Texte, dann sehen Sie die Ergebnisse klarer.


```
A   <TABLE BORDER=0 CELLPADDING=5 CELLSPACING=2>
      <TR>
         <TD WIDTH=10%>
         <BR>
         </TD>

B        <TD WIDTH=200>
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
   Hier steht der 1. Text.
         </TD>
         <TD WIDTH=10%>
         <BR>
```

```
         </TD>

C        <TD WIDTH=200 ALIGN=center VALIGN=middle>
   Hier steht der 2. Text.
   Hier steht der 2. Text.
   Hier steht der 2. Text.
         </TD>
      </TR>

      <TR>
         <TD WIDTH=10%>
         <BR>
         </TD>

D        <TD WIDTH=200 ALIGN=left VALIGN=top>
   Hier steht der 3. Text.
   Hier steht der 3. Text.
   Hier steht der 3. Text.
   Hier steht der 3. Text.
   Hier steht der 3. Text.
         </TD>

         <TD WIDTH=10%>
         <BR>
         </TD>

E        <TD WIDTH=200 ALIGN=right VALIGN=bottom>
   Hier steht der 4. Text. Sehr kurz.
         </TD>
      </TR>
   </TABLE>

A   <TABLE BORDER=0 CELLPADDING=5 CELLSPACING=2>
```

Wir eröffnen die Tabelle, indem wir die Angaben über CELLPADDING und CELLSPACING einfügen. Geben Sie dem *<BORDER>*-Tag einmal einen höheren Wert, dann werden Sie besser sehen können, was diese beiden Untertags auslösen.

```
B   <TD WIDTH=200>
   Hier steht der 1. Text
         </TD>
```

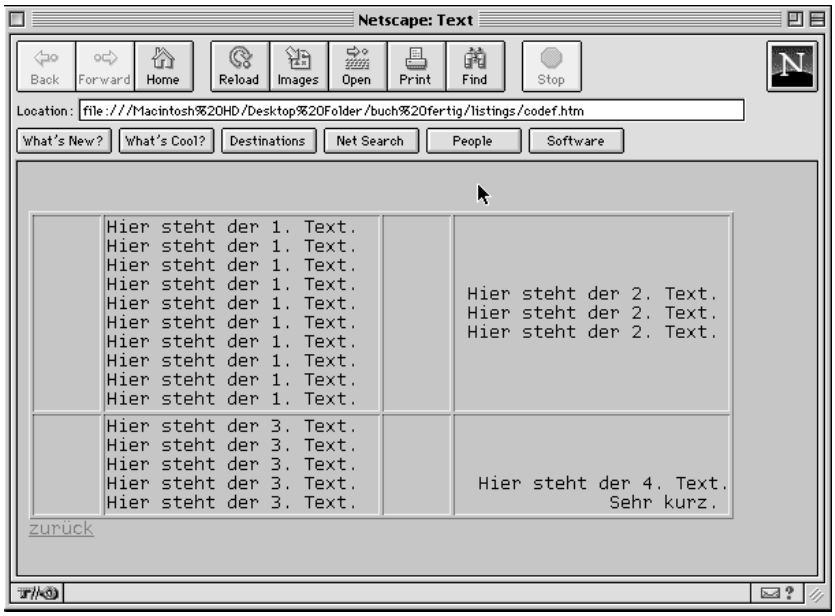

Bild 3.16: Zur besseren Übersicht steht der BORDER-Wert auf 1.

Da wir die Standardeinstellungen des Browsers noch einmal sehen wollen, ändern wir hier nichts. Der Text wird automatisch mittig und linksbündig in der Zelle plaziert. Experimentieren Sie in allen Zellen mit verschieden langen Texten, dann werden Sie sehen, wie stark solche Ausrichtungen im Gesamtbild Auswirkungen haben können.

```
C       <TD WIDTH=200 ALIGN=center VALIGN=middle>
   Hier steht der 2. Text
        </TD>
```

Hier haben wir den Text zentriert und auf jeden Fall mittig gestellt. Die verschiedenen Parameter für die Zentrierung können hier leicht zu Verwirrungen führen; deshalb empfiehlt sich hier aufmerksamstes Programmieren.

```
D        <TD WIDTH=200 ALIGN=left VALIGN=top>
   Hier steht der 3. Text
        </TD>
```

Dieser Text ist linksbündig ausgerichtet und beginnt immer links oben in der Zelle. Diese Werte, von denen der ALIGN-Tag bereits standardmäßig durch die Browser geleistet wird, kommt bei der Umsetzung von Spaltensatz über Tabellen am meisten zum Einsatz.

```
E       <TD WIDTH=200 ALIGN=right VALIGN=bottom>
   Hier steht der 4. Text
        </TD>
```

Rechtsbündig und an der untersten Linie der Zelle ausgerichtet kommt dieser Text daher. Um diese Effekte wirklich gut sehen zu können, sollten Sie in den verschiedenen Zellen verschieden lange Texte verwenden.

Mit diesen vier Fällen kann schon eine Menge an Layout über Tabellenspalten umgesetzt werden. Bei aller Euphorie sollten Sie aber nie vergessen, dass HTML ursprünglich nicht dazu gedacht war, perfekte Layouts in DTP-Manier zu generieren. Die Erwartungen sollten nicht zu hochgeschraubt sein, was eine pixelgenaue Umsetzung betrifft, auch wenn wir im nächsten Kapitel sehen werden, dass es durchaus Möglichkeiten gibt, die einzelnen Elemente einer Tabelle noch genauer zu plazieren.

3.5 Pixelgenaues Programmieren + PRE

Wir sind durch die Untertags im letzten Kapitel schon sehr firm geworden, um HTML-Tabellen in einer Page möglichst genau zu positionieren. Aber immer noch sind Tabellen eine unsichere Komponente für uns, wenn es darum geht, pixelgenaue »Spalten« aufzuziehen.

Man findet in diversen Veröffentlichungen relativ wenig darüber, wie der <*TABLE*>-Tag sich hier noch ein wenig genauer positionieren lässt, aber da hilft es manchmal, den anderen Webdesignern ein wenig auf die Finger zu schauen und dort den entscheidenden Trick zu finden.

Ich kann nur immer wieder empfehlen, vor allem in der Anfangsphase mindestens eine Stunde am Tag zu surfen und sich z.B. im Netscape »Navigator« über das Menü »View -> Source« anzuschauen, wie diese Seiten aufgebaut sind. Es dauert nicht lange, und man sieht vor allem durch die automatische Einfärbung der Tags innerhalb des Browsers, wie ein Webdesigner arbeitet. Es soll übrigens auch nicht schaden, den betreffenden Designer auf seiner Seite zu nennen, wenn man durch seine Tricks einen entscheidenden Input für die eigene Arbeit bekommen hat. Das ist eine Frage der Fairness und Höflichkeit.

Um die Katze gleich aus dem Sack zu lassen: Browser von Netscape und Microsoft können hier dazu verdonnert werden, die Tabellen pixelgenau aufzuziehen, wenn man ihnen eine Art von Kreuzreferenz bietet, in der die genauen Pixelzahlen zweimal vorkommen. Selbst hier kann es in Grenzfällen bei Netscape-Produkten zu Ungenauigkeiten kommen, aber das habe ich ja schon erwähnt. Angenommen, Sie möchten zwei Reihen mit je drei Tabellenzellen pro Reihe entwerfen, die alle ein festes Quadrat von 100 Pixeln aufbauen sollen, dann erzeugen Sie diese fixen Pixelzahlen wie folgt:

```
<TABLE BORDER=0 WIDTH=300 HEIGHT=200>
   <TR>
      <TD WIDTH=100 HEIGHT=100>
      <BR>
      </TD>

      <TD WIDTH=100 HEIGHT=100>
      <BR>
      </TD>

      <TD WIDTH=100 HEIGHT=100>
      <BR>
      </TD>
   </TR>

   <TR>
      <TD WIDTH=100 HEIGHT=100>
      <BR>
      </TD>

      <TD WIDTH=100 HEIGHT=100>
      <BR>
      </TD>

      <TD WIDTH=100 HEIGHT=100>
      <BR>
      </TD>
   </TR>
</TABLE>
```

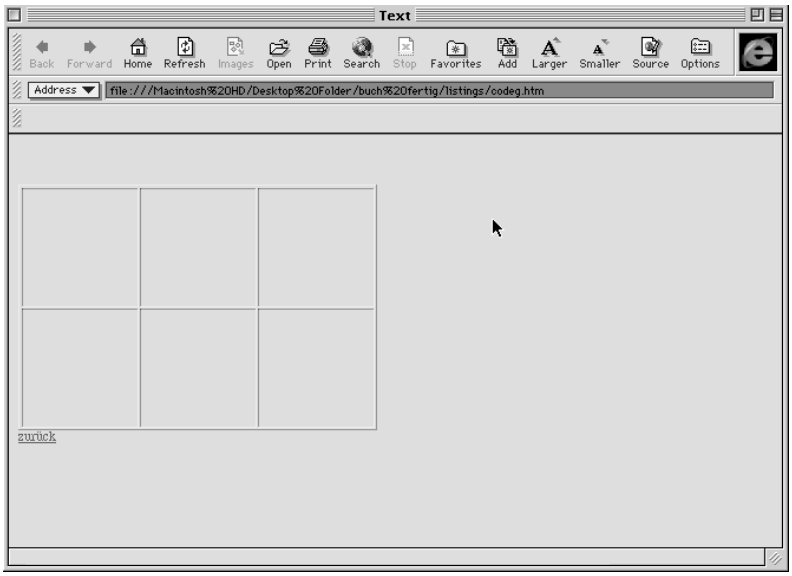

Bild 3.17: Durch die BORDER-Werte sieht man, wie fix die Tabellen sind.

Es ist schon einmal gesagt worden: Die meisten Webdesigner nutzen eine pixelgenaue Programmierung der Tabellen, um auch bei kleinen Bildschirmen eine möglichst gute Ausnutzung zu bekommen, den geforderten Satzstand auch perfekt einzuhalten und vor allem waagerechte Scrollbalken dadurch zu verhindern, dass die Tabelle möglichst in der Breite nicht über ein voll aufgezogenes Fenster auf einem 15"-Bildschirm reicht (hier stehen 640 x 480 Pixel zur Verfügung). Da auch hier Netscape und der Explorer nicht immer pixelgenau reagieren, empfehlen sich in der Breite etwa 580 Pixel. Damit ist noch genug Platz für den Fensterrand auf Windows und MAC/OS sowie einen senkrechten Scrollbalken, der sich vor allem bei textlastigen Seiten so gut wie nie verhindern lässt. Um dabei nicht alles am linken oberen Eck des Bildschirms kleben zu haben, zentrieren viele Designer die komplette Tabelle, die den Spaltensatz enthält. So kann das Layout sich bei größeren Bildschirmen von der Mitte her entfalten.

Die Summe der Pixelangaben pro Zelle muss einfach am Schluss exakt die Zahl der Pixel ergeben, die Sie im *<TABLE>*-Tag eintragen. Das ist es. Da der Browser jetzt zwei identische Summen für die Tabelle erhält, baut er sie auch exakt auf. Aber seien Sie gewarnt: Auch darauf können Sie nicht immer bauen, denn gerade Netscape ist hier in den Versionen 2.0/3.0 eher unsauber in der Programmierung.

Und außerdem können Sie sicher sein, dass auch nur ein kleiner Rechenfehler in der Summe Ihre Programmierung ordentlich durcheinander wirft.

Sollten alle genauen Positionierungen nicht helfen, dann hat HTML noch einen Tag in der Trickkiste, mit dem Sie sehr vorsichtig umgehen sollten. Er lautet:

```
<PRE></PRE>
```

Und er bedeutet, dass alle Eingaben aus dem Source-Code exakt so wiedergegeben werden, wie Sie dort stehen. Das hört sich zuerst ein wenig absurd an. Wenn Sie sich aber daran erinnern, dass Zeilenumbrüche aus dem Source-Code normalerweise nur dann angezeigt werden, wenn Sie dafür den entsprechenden Tag eingeben, und nicht mehr als ein Leerzeichen transformiert wird, dann ahnen Sie schon, was dieser Tag nun plötzlich ermöglicht. Sie können innerhalb des *<PRE>*-Tags auch mehrere Leerzeichen und einen festen Zeilenumbruch angeben. Allerdings warne ich Sie aus zwei Gründen vor einer allzu euphorischen Anwendung:

Zum einen ändert sich die Font-Art in diesem Modus so, wie Sie das schon vom *<TT>*-Tag her kennen. Der User sieht also sehr schnell, dass Sie hier in einem anderen Modus arbeiten, und das kann stören.

Des Weiteren, und das ist viel wichtiger, mag das so gewonnene Layout auf Ihrem Computer perfekt aussehen. Allerdings zerstört es sich auf anderen Computern mit anderen Schriften und anderen Fontgrößen sehr schnell. Zu allem Übel laufen die Schriften dann oft noch über den Zellenrand hinaus oder zerstören durch den unbedingten Umbruch den Textfluss. Sie werden also nur bedingt mit diesem Tag glücklich. Ich kann Ihnen nur empfehlen, *<PRE>*-Tags nur für etwas ausgeflipptere Layouts zu verwenden, in denen sich ein »Fehler« sogar blendend macht. Aber diese Fälle sind selten, und deshalb haben wir diesen Tag auch ganz am Schluss dieses Kapitels behandelt.

Lassen Sie mich am Ende dieses sicher nicht leichten Kapitels noch ein wenig aus dem Nähkästchen plaudern. Zum einen haben Sie durch die vorherigen Seiten eine Menge an Denksport hinter sich gebracht, zum anderen werden Sie vielleicht wissen wollen, wie man bei Tabellen am besten die Übersicht behält und am schnellsten arbeitet.

Manche Webdesigner sind stolz darauf, durch paradoxe Programmierung Dinge zu erreichen, die eigentlich so nicht darstellbar sind. Schön und gut. Bedenken Sie aber, dass im Moment alle vier Monate eine neue Browserversion von den Hauptkonkurrenten Microsoft und Netscape auf den Markt kommt, die diese genutzten Fehler oft mit einem Bugfix in ihrer Wirkung stoppt. Das berühmteste Beispiel für schlechte Trickkarten ist »http://

www.bild.de«, das in der Version Netscape 2.0 einen faszinierenden Trick nutzte, um einen bestimmten Randeffekt zu erzeugen. In Netscape 3.0 löste derselbe HTML-Code allerdings ein hässliches Randgerüst aus, das eher wie eine Baustelle daherkam ... also lassen Sie lieber die Finger vom Querfeld-einprogrammieren.

Nun ist es ja kein Geheimnis mehr, dass diverse Editoren auf dem Markt sind, die mit ein paar Klicks Tabellen zaubern können. Sie könnten es sich also sehr einfach machen. Aber ich rate ihnen dennoch davon ab. Wenn ich einen Satzspiegel einer HTML-Seite in Tabellen aufbaue, dann habe ich ein klares Vorgehen:

Ich schalte das Radio ab, vergesse für Stunden alles um mich herum, werfe notfalls die Katze aus dem Zimmer und konzentriere mich sehr, sehr, sehr genau darauf, wie ich die entsprechenden Tabellen aufbaue, wann und wo ich sie beginnen lasse und wann ich sie beende, um eine neue Tabelle zu beginnen. Das ist die heikelste Arbeit im ganzen Programmieren.

Warum tue ich das?

Viele Webdesigner denken zwar höllisch darüber nach, wie sie große Byte-Zahlen mit Bildern verhindern können und welcher Trick Ihnen noch kleinere Bilder etc. ermöglicht, aber sie vergessen die Zeit, die vor allem ältere Browser und Computer brauchen, um eine geladene Tabellendefinition zu berechnen und anzuzeigen.

Dann ist es auch so, dass vor allem die Netscape-Browser durch den Einsatz von kleinen versteckten Bilddateien einen stabileren Tabellenaufbau erreichen, aber das muss ja auch wieder geladen werden.

Um es kurz und schmerzlos zu machen:

Je weniger Tabellenverschachtelungen Sie benötigen, um das gewünschte Layout zu bekommen, desto besser. Sie sparen Lade- und auch Rechenzeit.

Und wenn Sie es sogar schaffen, eine Seite nicht als unzertrennbares Ganzes zu sehen, dann sind Sie auf dem Weg, ein ganz großer Webdesigner zu werden. Ehrlich. Sicher schmunzeln Sie jetzt, aber das tun Sie nur so lange, bis ich Ihnen erklärt habe, dass Sie eine Menge Zeit sparen, wenn Sie eine Seite immer als Ansammlung von Bausteinen definieren können. Und diese Bausteine sind vor allem Tabellen.

Stellen Sie sich eine Webpage wie eine Mauer vor, die Sie aus Klötzchen gebaut haben. Das geht nur, wenn Sie auf bestimmte Klötzchennormen zurückgreifen können. Es gibt vielleicht einen Normklotz, dann gibt es seine

Hälfte oder seine doppelte Größe, aber sicher haben diese Klötzchen alle die gleiche Höhe, damit sie frei untereinander kombiniert werden können.

Wenn Sie diese Metapher nun auf das Erstellen einer Website anwenden, werden Sie schnell merken, wie leicht die Arbeit werden kann. Wenn die Katze aus dem Haus fliegt und neben mir der Kaffe kalt wird, dann tue ich nichts anderes als Tabellenklötzchen definieren, die untereinander frei kompatibel sind. Jedes dieser Tabellenklötzchen teste ich auf Herz und Nieren. Sind alle Browser mit dem Code zufrieden? Baut der Code in allen gängigen Betriebssystemen sauber auf? Gibt es noch eine schnellere Lösung für die notwendigen Zellen? Wenn ich diese Testphase abgeschlossen habe, dann verwende ich nur noch diese Klötzchen, keine anderen; denn auf die ich mich zu 100 Prozent verlassen.

Sie werden im Kapitel 10 noch genauer erfahren, wie man eine Seite aufplanen muss, um zu wissen, welche Klötzchen man sicher braucht und welche nicht. Aber Sie ahnen gewis schon jetzt, dass man mit dieser Art des Vorgehens zum einen beim eigentlichen Bauen der Seiten sehr viel Zeit sparen kann (man muss die einzelnen Seiten eigentlich nicht mehr sehr stark testen) und zum anderen auch sehr einfach Ordnung in die eigene Site bringt.

Übrigens auch optisch: Wenn dem Auge immer wieder ähnliche Tabellenbreiten am Rand und in der Höhe geboten werden, dann erzeugt sich unterschwellig schnell das angenehme Gefühl von Übersichtlichkeit. Wenn Sie natürlich genau diesen Eindruck nicht erzeugen wollen, dann sollten Sie weiterhin kreuz und quer programmieren.

Unter »http://taglinger.de« sollten Sie sich einfach einmal den Code genauer anschauen. Dann werden Sie merken, woher die aufgeräumte Wirkung der Seiten entsteht, obwohl keine Trennlinien oder ähnliche Igitt-Gestaltungsmittel genutzt wurden.

Übrigens verweise ich auf die zahlreichen Layout- und Grafikerbücher im Handel. Auch ein Layouter-Grundkurs bei einer Volkshochschule oder an einer Universität kann nicht schaden. Das so erworbene Wissen können Sie stärker umsetzen, als Sie vielleicht jetzt noch glauben. Sie haben ja in diesem Kapitel gelernt, pixelgenau zu programmieren. Webdesigner haben übrigens intern immer Wetten mit Grafikern und Layoutern laufen, die behaupten, das vorliegende Layout könne man nicht im Web umsetzen. Man kann. Man kann, manchmal nur mit Abstrichen an Ladezeit und Übersichtlichkeit, aber man kann. Aber mal ehrlich: Nicht jedes Design ist im Web sinnvoll, oder fänden Sie es komisch, in ihrem Badezimmer ein Zehnmeterbrett installiert zu bekommen? Eben.

So, genug geplauscht. Jetzt wollen wir uns wieder den Befehlen von HTML zuwenden.

HTML-Code am Schluss des Kapitels:

1. Das Inhaltsverzeichnis

```
<HTML>
   <HEAD>
      <TITLE>Inhaltsverzeichnis</TITLE>
   </HEAD>

<BODY>
Fügen Sie hier ein:    (CODE 1) oder
                (CODE 2) oder
                (CODE 3) oder
                (CODE 4) oder
                (CODE 5) oder
                (CODE 6) oder
                (CODE 7) oder
                (CODE 8) oder
                (CODE 9)
</BODY>
</HTML>
```

2. Die Texte

```
<HTML>
   <HEAD>
      <TITLE>Text</TITLE>
   </HEAD>
<BODY>

<TABLE BORDER=0 WIDTH=95% HEIGHT=10%>
   <TR>
      <TD>
         <BR>
      </TD>
   </TR>
</TABLE>

Fügen Sie hier bitte CODE A oder
                CODE B oder
                CODE C oder
                CODE D oder
                CODE E oder
                CODE F oder
```

```
              CODE G ein.
<A HREF=inhalt.htm>zurück</A>

</BODY>
</HTML>
```

Jetzt können Sie bereits auf drei Arten Ordnung in Ihre HTML-Files bringen:

- ✘ Über Listen, die sich sogar automatisch durchnummerieren
- ✘ Über Tabellen, die Übersicht in die Daten schaffen
- ✘ Über Spaltensatz, der den Text da plaziert, wo Sie ihn haben wollen

Im nächsten Kapitel wollen wir uns aber endlich der Farbgebung widmen. Denn bis jetzt ist alles, was wir gearbeitet haben, noch farblos und grau.