
jetzt lerne ich
Start ohne Vorwissen
HTML5
CHRISTOPH PREVEZANOS
Markt+Technik

Kapitel 3

jetzt lerne ich

Besondere Auszeichnungen für Absätze, Wörter & Elemente

Haben Sie die Grundstruktur Ihres HTML-Dokuments mit Überschriften, Sektionen und Bereichen definiert, füllen Sie die Seiten mit Ihren Inhalten. Tippen Sie Ihren Text einfach in den Editor innerhalb des gewünschten Abschnitts. Allerdings besteht ein Artikel meist nicht einfach nur aus einem langen Fließtext. Häufig müssen Sie Absätze oder Zeilenumbrüche einfügen, Zitate hervorheben oder andere Zusatzinformationen kennzeichnen. Hierzu kennt HTML eine ganze Reihe von Textauszeichnungen, die Sie verwenden können. Beachten Sie dabei unbedingt, dass diese Befehle ausschließlich der Deklaration und Strukturierung der Webseite dienen. Die Befehle dürfen nicht zweckentfremdet werden, um z. B. ein bestimmtes Aussehen oder Layout des Texts zu erzielen.

3.1 Auszeichnungen für Absätze

Absätze stellen typischerweise einen größeren Textblock innerhalb eines längeren Dokuments dar. Das ist grundsätzlich auch das Verständnis in HTML. Allerdings wird der Begriff »Absatz« hier ein wenig weiter gefasst. Als Absatz gelten auch größere Elemente, die für sich alleine in einem Dokument stehen dürfen. Das können z. B. größere Zitatblöcke sein, Illustrationen mit Legende, Kommentare oder sogar begrenzende Linien. HTML kennt verschiedene Befehle, um solche Sinnabschnitte und Absätze zu erzeugen bzw. im Text zu deklarieren. Dieser Absatz zeigt Ihnen, welche Befehle Ihnen dabei zur Verfügung stehen.

3.1.1 P – Absätze erzeugen

Den Haupttext Ihrer Seite können Sie mit Ihrem Texteditor direkt in das HTML-Dokument tippen. Am besten setzen Sie vor das Ganze eine Überschrift und legen dann mit Ihren Inhalten los. Um den Text ein wenig übersichtlicher zu gestalten, können Sie dabei im Texteditor mit Zeilenumbrüchen, Absätzen und Leerzeilen arbeiten. Wie bereits erwähnt, hat das aber keinerlei Auswirkungen auf die Anzeige im Browser. Ihre Absätze werden wie ein langer Fließtext angezeigt.

Um Ihren Text sinnvoll zu strukturieren, müssen Sie Absätze einfügen. Hierzu dient der ***Paragraph***-Befehl, der einfach nur *P* lautet. Öffnen und schließen Sie einen Absatz immer mit dem *P*-Befehl, damit Ihr Text als eine Einheit angesehen wird. Das ist vor allem für die spätere Formatierung mit Stylesheets sehr wichtig. Am besten schreiben Sie reinen Fließtext ausschließlich innerhalb des *P*-Befehls und vermeiden es, einen Text ohne jede Auszeichnung zu erzeugen.

Richtig:

```
<h1>Überschrift</h1>
<p>
Dies ist ein eigener Absatz. Er wird immer als geschlossene Einheit
interpretiert und automatisch durch einen Zeilenumbruch am Anfang und
Ende dargestellt.
</p>
```

Falsch:

```
<h1>Überschrift</h1>
Dies ist ein eigener Absatz. Er wird immer als geschlossene Einheit
interpretiert und automatisch durch einen Zeilenumbruch am Anfang und
Ende dargestellt.
```

Oft wird der *P*-Befehl alleine verwendet, um an einer bestimmten Stelle einen Zeilenumbruch zu erzeugen. Tun Sie das niemals, denn dafür ist der P-Befehl überhaupt nicht gedacht. Außerdem können so weder der Browser noch ein Text-Management-System erkennen, wo der Abschnitt anfängt und aufhört.

Falsch:

```
Text, Text, Text
<p>
Text, Text, Text
```

Richtig:

```
<p>Text, Text, Text</p>

<p>Text, Text, Text</p>
```

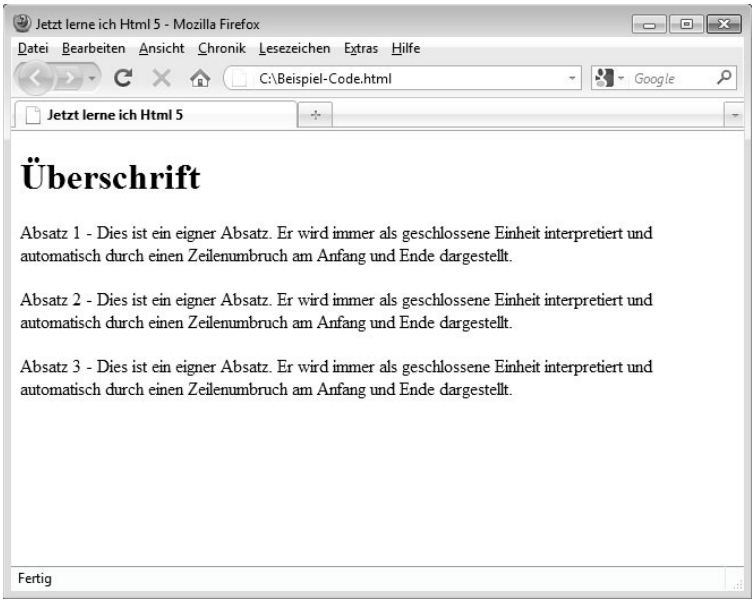

Abb. 3.1: Ein Artikel mit Kopf- und Fußbereich

Textumbruch und Silbentrennung im Browser

Beachten Sie beim Tippen Ihrer Texte immer, dass es in HTML keinerlei Silbentrennung gibt. Ein Zeilenumbruch kann immer nur vor oder nach einem Wort stattfinden. Das führt dazu, dass sehr lange Wörter manchmal das Erscheinungsbild negativ beeinflussen, z. B. weil am Ende einer Zeile eine sehr lange Lücke entsteht. Das lässt sich derzeit in keiner Weise umgehen. Sie haben zwar die Möglichkeit, manuell Trennstriche zu setzen, und diese werden auch richtig übernommen, aber davon ist dringend abzuraten. Ändert sich das Layout Ihres Textes, die Schriftgröße, oder zieht der Besucher einfach sein Browser-Fenster größer, fließt der Text völlig anders. Die Folge wären zusammenhanglose Striche irgendwo im Text.

3.1.2 BR – Zeilenumbrüche einfügen

Möchten Sie in einem Fließtext einen Zeilenumbruch erzeugen, sollten Sie hierfür den Break-Befehl verwenden. Er lautet schlicht *BR*. Der *BR*-Tag stellt eine Besonderheit dar, weil er nicht geöffnet oder geschlossen werden muss. Er steht für sich alleine. Damit fügen Sie einen Zeilenumbruch ein, ohne dabei die logische Struktur des Texts zu verändern. Im Gegensatz zum *P*-Befehl wird ein längerer Text also weiterhin als eine Einheit interpretiert. Der *P*-Befehl steht immer für einen neuen Absatz und somit meist für einen inhaltlichen Wechsel.

```
Text, Text, Text
<br>
Text, Text, Text
<br>
Text, Text, Text
```

Verwenden Sie den Break-Befehl nicht, um mit mehreren Leerzeilen einen größeren Abstand und somit ein Layout zu erzwingen. Das ist unpraktisch, sieht nicht gut aus und geht mit CSS viel besser. Ebenso sollten Sie mit dem Befehl keine künstlichen Listen oder Aufzählungen erzeugen. Auch hierfür gibt es eigene Befehle. Stattdessen verwenden Sie den Break-Befehl ausschließlich, um sinnvolle Zeilenumbrüche innerhalb eines Absatzes oder anderer Textelemente zu erzeugen.

```
<h1>Jetzt lerne ich HTML 5</h1>

<p>
Text, Text, Text
<br>
Text, Text, Text
</p>

<h2>Erste Schritte in HTML</h2>

<p>
Text, Text, Text
<br>
Text, Text, Text
</p>
```

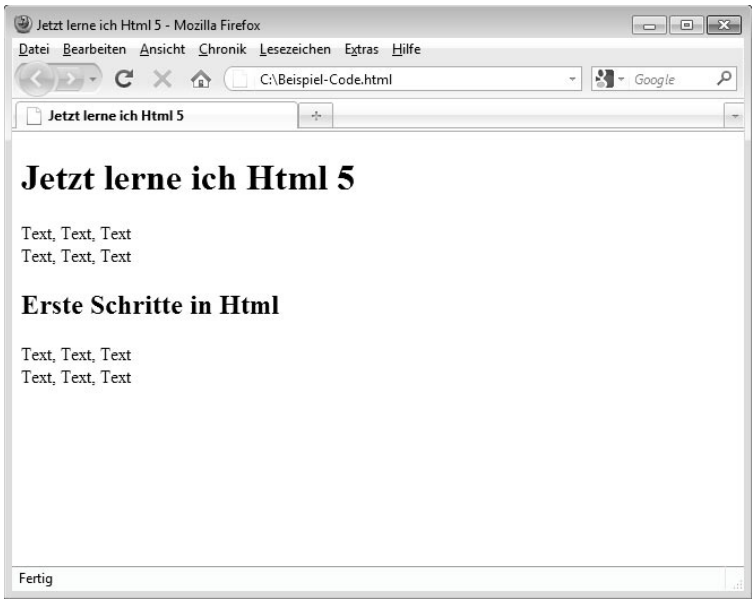

Abb. 3.2: Den Text mit Zeilenumbrüchen gestalten

3.1.3 NOBR & WBR – Zeilenumbrüche steuern

Mit dem *BR*-Befehl erzwingen Sie in HTML einen Zeilenumbruch. Das Gegenstück dazu heißt *NOBR* und verhindert, dass der Text durch den Browser umbrochen wird. Die Einsatzgebiete für diesen Befehl sind allerdings sehr eingeschränkt. In einem Fließtext sollten Sie ihn niemals einsetzen, denn der gesamte Text würde in einer langen Zeile dargestellt werden. Die Folge wäre, dass der Besucher im Browser sehr weit nach rechts scrollen müsste, um den Text zu lesen. Vielmehr ist dieser Befehl für einzelne lange Elemente gedacht, wie z. B. Zitate oder auch Internetadressen. Vor und nach dem *NOBR*-Befehl wird automatisch eine neue Zeile angefangen.

```
<nobr>
Dieser Text wird im Browser nicht umbrochen. Der Besucher muss also
nach rechts scrollen, um ihn lesen zu können.
</nobr>

<nobr>
http://www.prevezanos.com/
index.php?option=com_content&view=frontpage&Itemid=1
</nobr>
```

Abb. 3.3: Zeilenumbrüche gezielt unterdrücken.

Normale Worte ohne Bindestriche oder sonstige Unterteilungen werden vom Browser in der Regel niemals umbrochen, egal wie lang sie sind. Das kann auch bei langen Adressen oder Formeln passieren. Das sieht überhaupt nicht gut aus, weil sich dadurch sehr merkwürdige Zeilenumbrüche und Absätze ergeben. Um das zu verhindern, lassen sich mit dem *WBR*-Befehl in langen Worten Umbruch-Optionen deklarieren. Sie geben dem Browser also an, wo er das Wort umbre-

chen darf, wenn es notwendig ist. Ist ein Zeilenumbruch nicht erforderlich, hat der Befehl keine Auswirkungen. Der *WBR*-Befehl steht übrigens für sich alleine – er muss also nicht geöffnet und geschlossen werden.

```
<p>
Mary Poppins findet das
super<wbr>kali<wbr>fragi<wbr>listisch<wbr>expi<wbr>alle<wbr>gorisch.
</p>
```

3.1.4 CITE – Titel, Sachnamen und Bezeichnungen

HTML kennt einen eigenen Befehl, mit dem sich Titel und Bezeichnungen deklarieren lassen. Gemeint sind hiermit Titel und Namen von Büchern, Zeitschriften, Filmen, TV-Sendungen, Kunstwerken, Spielen usw. Man könnte das unter dem Begriff Sachnamen zusammenfassen. Leider sorgt der Befehl aufgrund seines Namens *CITE* immer wieder für Missverständnisse – mit Zitaten hat er nämlich nichts zu tun.

```
<p>
Wenn Sie eine eigene Webseite erstellen möchten, sollten Sie das Buch
<cite>Jetzt lerne ich HTML 5</cite> lesen.
</p>

<p>
Das ist ein Zitat aus dem Film <cite>Casablanca</cite>.
</p>
```

Abb. 3.4: Titel markieren und hervorheben

Die meisten Browser heben mit *CITE* markierte Wörter hervor, z. B. durch Fettdruck oder kursive Darstellung. Verwenden Sie diesen Befehl aber nicht, um Eigennamen von Personen, Markennamen oder ähnliche herkömmliche Bezeichnungen hervorzuheben. Dafür gibt es spezielle Befehle, und *CITE* ist für Werke und Produktionen aus dem schöpferischen Bereich definiert.

3.1.5 Q & BLOCKQUOTE – Zitate hervorheben

Beim Schreiben Ihres Texts werden Sie sich bestimmt gelegentlich auf andere Quellen beziehen oder andere Autoren zitieren. Das ist nicht nur bei wissenschaftlichen Texten als Beleg wichtig, sondern im Sinne des Urheberrechts auch bei allen anderen Artikeln. Hierzu gibt es in HTML extra ein paar Tags, die genau für diese Aufgabe geschaffen wurden.

Einfache Zitate lassen sich mit dem Befehl *Q* (to quote = zitieren) markieren. Alle Browser setzen den Text automatisch in Anführungszeichen, einige heben ihn zusätzlich durch kursive Darstellung hervor. *Q*-Zitate eignen sich bestens für Fließtext, weil keinerlei Absatz oder Zeilenumbruch erzeugt wird.

```
<p>
Zum Thema einer konstruktiven Diskussion fällt mir spontan folgendes
Zitat ein: <q>Der eine sucht einen Geburtshelfer für seine Gedanken,
der andre einen, dem er helfen kann: So entsteht ein gutes
Gespräch.</q> Auf diese Weise werden alle Beteiligten neue
Gedankenanstöße mitnehmen.
</p>
```

Möchten Sie Ihr Zitat deutlich vom Text absetzen, sollten Sie das Blockzitat nutzen. Es wird mit dem Befehl *BLOCKQUOTE* deklariert und erzeugt automatisch einen eigenen Textblock. Das Zitat selbst muss aber weiterhin in einem eigenen *P*-Absatz verpackt werden, um als eine Einheit verstanden zu werden. Auf welche Weise das Blockzitat im Text dargestellt wird, ist leider nicht definiert und hängt somit vom Browser ab. Einige erzeugen große Absätze, andere rücken den Text ein, oder der Text wird kursiv gedruckt.

```
<blockquote>
<p>"Der eine sucht einen Geburtshelfer für seine Gedanken, der andre
einen, dem er helfen kann: So entsteht ein gutes Gespräch."</p>
</blockquote>
```

Blockzitate bieten die Möglichkeit, die Quelle mit anzugeben. Hierzu dient das Attribut *CITE*, das aber nicht mit dem gleichlautenden Befehl verwechselt werden darf. Bei der Quelle muss es sich um eine gültige Internetadresse handeln, die vollständig und in Anführungszeichen angegeben wird. Leider ist diese Zusatzinformation derzeit noch nutzlos, weil sie weder angezeigt noch in irgendeiner Weise ausgewertet wird.

```
<blockquote cite="http://www.zitate.de/zitat.html">
<p>"Der eine sucht einen Geburtshelfer für seine Gedanken, der andre
einen, dem er helfen kann: So entsteht ein gutes Gespräch."</p>
</blockquote>
```

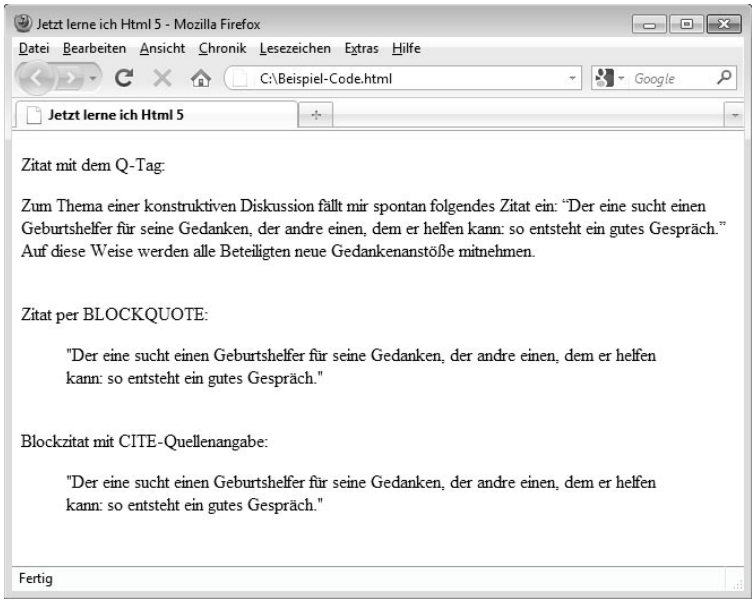

Abb. 3.5: Zitate und Blockzitate in einem HTML-Text

3.1.6 ADDRESS – Kontaktinformationen

Möchten Sie in Ihrem Text eine Adresse als eigenständiges Element deklarieren, können Sie das mit dem *ADDRESS*-Tag tun. Es eignet sich für E-Mail-Adressen, Postanschriften oder auch Verweise auf Kontaktseiten. Die meisten Browser zeigen den so deklarierten Text leicht hervorgehoben oder kursiv an, eine Verlinkung ähnlich wie bei richtigen Verweisen findet aber nicht statt. Seien Sie mit diesem Befehl etwas vorsichtig, denn damit zeigen Sie Werbefirmen genau, wohin sie die Spam-Mails schicken sollen.

```
<p>E-Mail-Adresse:</p>
<address>Daffy Duck, daffy@warnerbros.com</address>

<p>Autor des Artikels:</p>
 <address>
   <a href="../autor.html">Ch. Prevezanos</a>
 </address>
```

Abb. 3.6: Adressen im Text hervorheben

3.1.7 FIGURE – Illustrierende Elemente deklarieren

Mit dem Befehl *FIGURE* lassen sich in einem Text spezielle Elemente und Inhalte deklarieren und mit einer Beschreibung versehen. Das können Fotos oder Grafiken, Beispielcode, Hörproben oder ähnliche Inhalte sein. Es ist vergleichsweise schwierig, den Unterschied zu herkömmlichen Elementen zu verdeutlichen. Als *FIGURE* werden meist Elemente deklariert, die als Beispiel für den Haupttext dienen oder seine Aussage verdeutlichen sollen. Sie sind nicht zwingend zum Verständnis notwendig und könnten auch auf einer anderen Seite oder in seiner seitlichen Spalte stehen.

```
<figure>
<img src="pusteblume.jpg" alt="Pusteblume">
<figcaption>Das Bild zeigt eine typische Pusteblume.</figcaption>
</figure>
```

Im Beispiel befasst sich ein Artikel mit Pusteblumen. Als Beispiel wird das Foto einer typischen Pusteblume hinzugefügt. Mit dem Befehl *FIGCAPTION* lässt sich dem Foto zusätzlich eine kurze Beschreibung geben. Wichtig ist hierbei, dass sich der Befehl *FIGCAPTION* innerhalb des *FIGURE*-Befehls befinden muss. Außerdem muss sich die Beschreibung per *FIGCAPTION* ganz am Anfang oder ganz am Ende des *FIGURE*-Bereichs befinden, aber niemals in der Mitte.

```
<figure>
<figcaption>Code-Beispiel</figcaption>
<p><code>if ($day=="Fr") echo "Schönes Wochenende!";</code></p>
</figure>
```

```
<figure>
<p>Wer hat an der Uhr gedreht?<br>
Ist es wirklich schon so spät? <br>
Soll das heißen, ja ihr Leut<br>
Mit dem Paul ist Schluss für heut!</p>
<figcaption>Titelmelodie aus der Zeichentrickserie <cite>Der rosarote
Panther</cite>.</figcaption>
</figure>
```

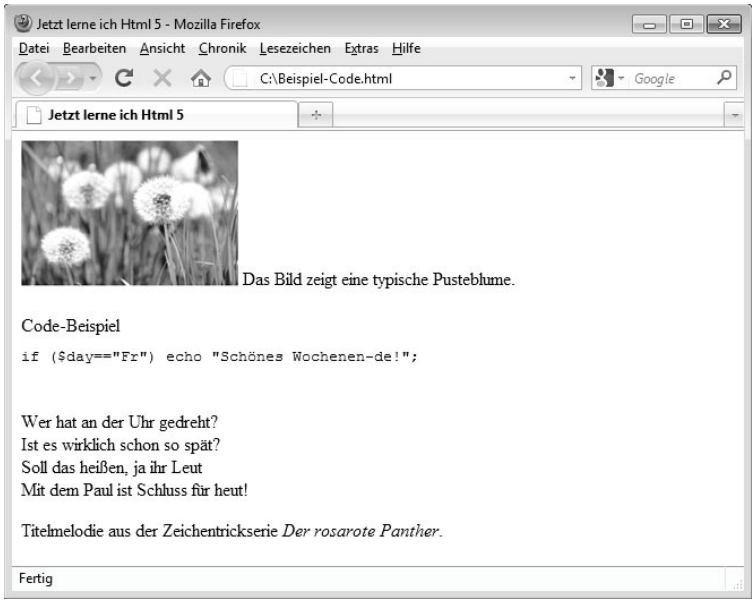

Abb. 3.7: Fotos, Code oder Zitate als FIGURE-Tag

3.1.8 PRE – Textpassagen vorformatieren

In HTML gibt es die Möglichkeit, einen Text genau so anzeigen zu lassen, wie Sie ihn im Editor eintippen. Es werden also keine Formatierungen, Einrückungen oder Ähnliches vorgenommen. Selbst Leerstellen und Absätze werden dabei genau so angezeigt, wie sie im Texteditor eingegeben wurden. Man spricht dabei von »präformatierten Textblöcken«. Der englische Begriff dafür lautet »preformat«, und entsprechend lautet der HTML-Tag schlicht *PRE*. Innerhalb des *PRE*-Blocks wird Ihr Text von den meisten Browsern mit der Schriftart Courier dargestellt.

```
<pre>
Hier beginnt die Zeile          und hier endet sie.

Ein Zeilenumbruch wird auch
als solcher dargestellt.

          Zeilen können mit Leerstellen eingerückt werden.
</pre>
```

Der *PRE*-Befehl wird häufig verwendet, um Quelltexte, Tastatureingaben oder Programmiercode darzustellen. Das ist besonders praktisch, weil Sie innerhalb des vorformatierten Bereichs auch weitere HTML-Befehle nutzen können. Hier bieten sich vor allem die Textauszeichnungen für Code an.

```
<p>Der Code für das Programm lautet:</p>

<pre><code>
#include <stdio.h>
     int main()
     {
         printf("hello, world");
         return 0;
 }
</code></pre>
```

Obwohl alle Browser die Zeichen innerhalb des *PRE*-Blocks sozusagen roh anzeigen, gelten dabei natürlich dieselben Einschränkungen beim Zeichensatz, bei den Sonderzeichen und den HTML-Tags. Sie können also keine Zeichen abbilden, die nicht Teil des ASCII-Zeichencodes sind. Sämtliche Sonderzeichen müssen Sie wie zuvor mit speziellen Codes abbilden. Achten Sie auch auf die spitzen Klammern (< >). Sie sind nach wie vor verboten und bleiben den HTML-Tags vorbehalten.

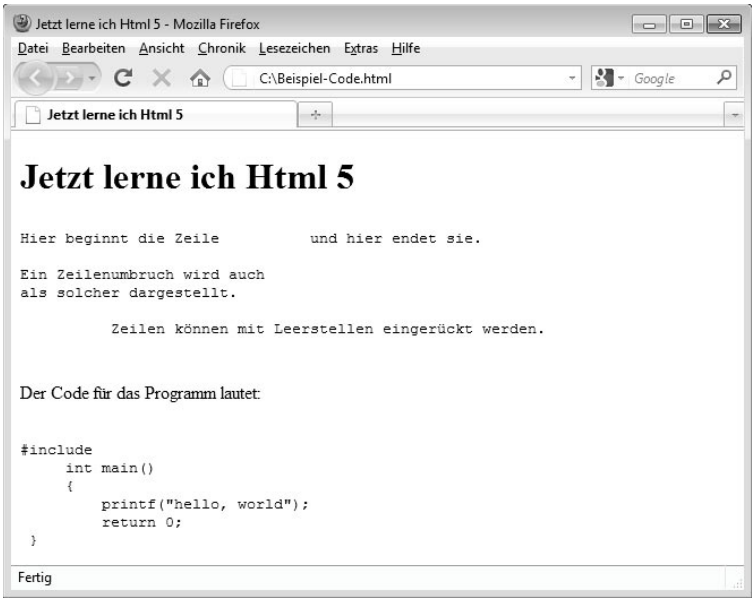

Abb. 3.8: Vorformatierter Text und Programmcode

3.1.9 HR – Linien für die Texttrennung

Manchmal ist es ganz praktisch, den Text mit ein paar Begrenzungen auszustatten. Hierzu bietet sich der Befehl *HR* an, mit dem eine schlichte Linie erzeugt wird. Anders als die meisten HTML-Befehle steht HR für sich alleine – der Befehl muss also nicht geöffnet und geschlossen werden. Setzen Sie den Befehl einfach an die gewünschte Stelle. Vor und hinter der Linie wird automatisch ein Absatz eingefügt, sodass die Linie immer in einer eigenen Zeile steht.

```
... Text ...
<hr>
... Text ...
```

Beachten Sie hierbei, dass der HR-Befehl kein Gestaltungsmittel darstellt, z. B. um den Text mit einem hübschen Rahmen auszustatten. Vielmehr soll er einen inhaltlichen oder thematischen Wechsel im Text optisch deutlich hervorheben. Der HR-Befehl eignet sich sehr gut in Fließtext zwischen Absätzen. Er sollte nicht zwischen Sektionen und Artikeln verwendet werden, weil diese automatisch einen inhaltlichen und thematischen Wechsel beinhalten.

Abb. 3.9: Den Text mit einer Linie trennen

3.1.10 Kommentare in den HTML-Code einfügen

Bei großen und aufwendigen Webseiten müssen Sie den HTML-Code und die Anzeige im Browser sicherlich mehrfach überprüfen und vergleichen. Da kann man schon mal den Überblick verlieren, besonders wenn Sie später das Layout mit Menüs, Grafiken und viel CSS erstellen. Um ein wenig Ordnung zu schaffen, lassen sich in HTML auch Kommentare einfügen. Diese werden vom Browser

nicht ausgewertet oder angezeigt. Sie erscheinen aber im Quellcode, sodass Sie sich ein bisschen besser zurechtfinden.

Ein Kommentar wird in HTML im Grunde wie ein Pseudo-Befehl eingefügt. Er steht also in spitzen Klammern, mit jeweils zwei Bindestrichen am Anfang und am Ende und einem Ausrufezeichen am Anfang. Diese Syntax hat den Vorteil, dass alle aktuellen Browser den Befehl erkennen und ältere Browser darin einen unbekannten HTML-Befehl sehen und ihn ignorieren. Der Text wird also in keinem Fall angezeigt.

```
<!--  Hier steht ein Kommentar  -->
```

Mit solchen Kommentaren lassen sich nun z. B. die Bereiche einer Webseite markieren. Geben Sie an, wo Ihr Menü beginnt und aufhört, wo der Haupttext losgeht usw. Auf diese Weise ist der HTML-Code überschaubarer, und mögliche Fehler lassen sich besser finden.

```
<body>

   <!--  Hier beginnt das Menü -->
<a href="startseite.html">Startseite</a>
<a href="kontakt.html">Kontakt</a>
   <!--  Hier endet das Menü -->

   <!--  Hier beginnt der Haupttext -->
<h1>Jetzt lerne ich HTML 5</h1>
Text ...
Text ...
   <!--  Hier endet der Haupttext -->

</body>
```

Kommentare per COMMENT-Tag

Früher haben viele HTML-Autoren den *COMMENT*-Befehl für Kommentare im Quelltext verwendet. Er wurde wie ein herkömmlicher HTML-Befehl geöffnet und geschlossen, dazwischen stand der Kommentartext. Die meisten Browser verarbeiten den Befehl immer noch problemlos, allerdings war er nie offizieller Teil des HTML-Standards. Deshalb sollten Sie auf diesen Befehl besser verzichten und die oben genannte, standardkonforme Methode verwenden.

3.1.11 ü – Sonderzeichen im Text verwenden

Sonderzeichen waren in HTML bisher immer ein Problem. Mit der Einführung von UTF-8 als Standardzeichensatz hat sich das deutlich gebessert. Die meisten alltäglichen Zeichen fremder Sprachen werden von den Browsern selbstständig erkannt und verarbeitet. Sie dürfen in Ihren Dokumenten also direkt deutsche Umlaute eintippen. Trotzdem gibt es noch eine ganze Reihe von Sonderzeichen, die sich nicht auf diese Weise verarbeiten lassen. Oft liegt das am Webbrowser,

häufig aber auch am Texteditor, der diese Zeichen ebenfalls nicht darstellen kann.

Damit solche Sonderzeichen trotzdem möglich sind, gibt es eine eigene HTML-Codierung. Es handelt sich dabei um einen eigenen Befehl bzw. ein Tag, mit dem sich Sonderzeichen in das Dokument einfügen lassen, sodass sie jeder Browser versteht. Die Sonderzeichen werden also unabhängig von der Sprachcodierung richtig verarbeitet. Die Codierung ist bei allen Sonderzeichen ähnlich: Eingeschlossen von einem kaufmännischen Et (= &) und einem Semikolon (= ;) geben Sie den Code für das Sonderzeichen ein.

```
&CODE1;   &CODE2;   &CODE3;
```

Bei der Wahl des Codes hat man sich an den tatsächlichen Namen der Zeichen orientiert, wodurch man sich diesen deutlich besser merken kann. Ein deutsches »ä« wird auch als »a mit Umlaut« bezeichnet. Und genau so lautet auch der Code in verkürzter Form.

```
&auml;   =  ä
&uuml;   =  ü
&ouml;   =  ö
&szlig;  =  ß
```

Der Code unterscheidet hierbei Groß- und Kleinschreibung. Wird der erste Buchstabe des Codes großgeschrieben, erscheint auch das Sonderzeichen groß.

```
&Auml;   =  Ä
&Uuml;   =  Ü
&Uuml;   =  Ö
&Szlig;  =  ß (identisch, da es kein großes ß gibt)
```

Sie setzen diese Sonderzeichen einfach ohne Leerzeichen direkt in Ihren Satz bzw. in das betreffende Wort. Das mag im Quellcode am Anfang verwirrend und unübersichtlich wirken, funktioniert aber problemlos.

```
<p>
Demn&auml;chst erfahren Sie mehr &uuml;ber HTML, z.B. wie Sie
Aufz&auml;hlungen erstellen, Abs&auml;tze formatieren usw.
</p>
```

Verwenden Sie als Zeichencodierung ISO-8859-1, ist das spezielle Codieren von deutschen Sonderzeichen überflüssig. Trotzdem wurden sie hier erläutert, weil sie sich sehr gut als Beispiel eignen. Im Alltag müssen Sie das nicht tun, wenn Sie den Zeichensatz mit angeben. Es gibt aber eine ganze Reihe von Sonderzeichen, die sich nicht direkt verarbeiten lassen. Oft gibt es dafür gar keine Taste auf der Tastatur. Typische Beispiele sind mathematische Formeln, internationale Währungszeichen, Wirtschaftszeichen wie ®, ™, © usw. Ebenso dürfen Sie keine spitzen Klammern verwenden, weil sie ein HTML-eigenes Kommando sind. Alle diese Zeichen müssen Sie zwangsläufig wie hier beschrieben codieren. Um Ihnen die Arbeit zu erleichtern und damit Sie sich die vielen Zeichen nicht merken müssen, finden Sie im Anhang ab Seite 407 eine Tabelle mit allen Sonderzeichen und deren Codierung.

3.2 Auszeichnungen für interaktive Elemente

HTML war ursprünglich als reine Auszeichnungssprache für die Elemente eines Dokuments gedacht. Im Laufe der Jahre hat sich HTML immer weiter entwickelt, sodass nun auch Fotos, Videos, Musik, Flash und viele andere Multimedia-Elemente eine Selbstverständlichkeit sind. Zu diesen neuen Elementen gehören auch interaktive Inhalte. Darunter kann man im weitesten Sinne alle Inhalte und Objekte verstehen, die sich vom Besucher verändern lassen, die sich auf sein Handeln einstellen oder auf eine andere Weise mit ihm oder für ihn interagieren und verändern. Das können im einfachsten Fall Menüs sein, die sich beim Anklicken verändern, kleine Spiele in JavaScript oder auch größere Anwendungen. Viele dieser Elemente wurden in HTML 5 komplett überarbeitet oder erst neu eingeführt.

Das klingt gut und vielversprechend, doch den meisten HTML-Autoren nützen sie wenig. Die meisten HTML-Befehle deklarieren lediglich, wo und wie eines dieser interaktiven Elemente platziert werden soll – sie bilden aber nicht das interaktive Element und sie lassen sich damit auch nicht erzeugen. Dazu ist nach wie vor immer ein externes Programm oder eine externe Programmiersprache notwendig. Meistens kommt dabei JavaScript, Flash, ein Java-Applet oder Ähnliches zum Einsatz. Und diese Elemente haben nichts mit HTML zu tun, sondern bilden eine völlig eigene Programmiersprache, die gesondert erlernt werden muss.

3.2.1 SCRIPT – Java-Skripte in Dokumente einbetten

Zu den bekanntesten interaktiven Elementen gehört sicherlich JavaScript. Von einer Skriptsprache redet man im Allgemeinen dann, wenn der Programmcode nicht zu Exe- oder Com-Dateien kompiliert wird, sondern direkt verarbeitet werden kann. So ist es auch bei JavaScript. Sie schreiben kleine Programme und platzieren den Code direkt in Ihr HTML-Dokument. Alle modernen Browser können mit JavaScript umgehen und führen die Anweisungen direkt aus. Dadurch ist es möglich, ein HTML-Dokument interaktiv zu gestalten.

Streng genommen stellt JavaScript aber weiterhin ein externes Objekt dar. Es hat mit HTML nichts zu tun und wird zusätzlich in den Text eingebunden. Aus diesem Grund gibt es spezielle Befehle, die diese Einbindung ermöglichen. In der Regel ist es so, dass das Programm abseits vom Inhalt irgendwo in das Dokument geschrieben wird. An der gewünschten Stelle im Dokument befindet sich dann nur noch der Auslöser für das Programm.

Diesen getrennten Bereich erzeugen Sie mit dem Befehl *SCRIPT*. Er erzeugt einen gesonderten Bereich innerhalb des Dokuments, der ausschließlich für das Skript reserviert ist. Der Browser weiß sofort, dass er hier keine herkömmlichen Inhalte finden wird. Deshalb müssen Sie immer darauf achten, dass der Bereich sauber wieder geschlossen wird.

```
... HTML-Inhalte ...

<script>
   JavaScript-Programm
</script>

 ... HTML-Inhalte ...
```

Früher war es üblich, den *SCRIPT*-Bereich innerhalb des Dokumenten-Headers zu setzen. Das sollte das Dokument übersichtlicher machen. Gegen diese Vorgehensweise spricht auch heute nichts, aber sie wird vom HTML-Standard nicht mehr zwingend vorgeschrieben. Sie dürfen den *SCRIPT*-Bereich an jede beliebige Stelle des Dokuments setzen.

Für die meisten Browser und Programme ist es vollkommen ausreichend, den reinen *SCRIPT*-Bereich zu deklarieren. Es gibt aber optional ein paar Attribute, die verwendet werden können:

- **TYPE** – Mit dem Attribut *TYPE* geben Sie die Art des Skripts an. Auch wenn die meisten Browser sofort von JavaScript ausgehen, muss das nicht immer so sein. Deshalb kann es nicht schaden, wenn Sie den Typ mit angeben.
- **CHARSET** – Dieses Attribut gibt an, mit welchem Zeichensatz ein Skript geschrieben ist. In der Regel sollten dabei nur Zeichen aus dem ASCII- oder UTF-8-Satz verwendet werden. Immerhin ist nicht gewährleistet, dass Browser am anderen Ende der Welt mit europäischen Zeichen umgehen können. Falls es aber doch mal notwendig sein sollte, geben Sie hiermit den Zeichensatz an. Ansonsten lassen Sie das Attribut einfach weg.
- **SRC** – Falls der Code für das JavaScript nicht direkt in den *SCRIPT*-Bereich geschrieben werden soll, definieren Sie hiermit eine externe Datei. Dabei handelt es sich in der Regel um reine Textdateien mit dem Skript, welche die Endung ***.js*** tragen.

```
<script type="text/javascript">
   JavaScript-Programm
</script>

<script charset="ISO-8859-1">
   JavaScript-Programm
</script>

<script src="programm.js">
   JavaScript-Programm
</script>
```

JavaScript ist heute überhaupt nicht mehr aus dem Internet wegzudenken. Sie finden es auf sehr vielen Seiten. Trotzdem mag nicht jeder Nutzer JavaScript und deaktiviert es im Browser. Das kann die durchaus begründete Angst vor bösartigen Skripten sein oder der übermäßige Einsatz nerviger Skripte, die den Nutzer abschrecken. Es kann hierfür viele Gründe geben, und jeder Browser erlaubt das Deaktivieren von JavaScript mit wenigen Mausklicks. Die Folge ist oft, dass eine Seite nicht mehr richtig funktioniert und vielleicht sogar gar nichts anzeigt.

Um das zu verhindern, sieht HTML einen alternativen Befehl vor. Mit *NOSCRIPT* deklarieren Sie einen Bereich, den der Browser nur verarbeitet und anzeigt,

wenn das JavaScript nicht ausgeführt wird. Das schließt auch einen Fehler im Skript ein, der die Ausführung verhindert. Platzieren Sie in diesen Bereich einen sinnvollen Text oder einen Hinweis für den Besucher. Auch dieser Bereich muss korrekt geöffnet und geschlossen werden.

```
<script>
   JavaScript-Programm
</script>

<noscript>
   Diese Seite benötigt JavaScript!
</noscript>
```

3.2.2 DETAILS – Zusatzinformationen per Widget

In vielen Programmen und auch in Windows selbst gibt es häufig Zusatzinformationen, die in einem Extra-Fenster, einer Seitenleiste oder einem Fähnchentext untergebracht sind. Manchmal muss man dafür auch eine Schaltfläche wie ***Mehr*** oder ***Weiter*** betätigen. Eine ganz ähnliche Funktion möchte man nun auch in HTML 5 umsetzen. Sie soll es erlauben, Zusatzinformationen zu einem Inhalt oder einem Objekt als sogenanntes Widget einzublenden. Darunter kann man sich eine Art Fahne, aufklappbare Leiste oder Ähnliches vorstellen.

Ein solches Widget erstellen Sie mit dem Befehl *DETAILS*. Er muss in jedem Fall sauber geöffnet und geschlossen werden, weil nur so klar ist, was Teil des Widgets sein soll. Jedes Widget per *DETAILS*-Befehl muss einen Namen bzw. eine Überschrift besitzen. Hierzu dient der Befehl *SUMMARY* – er muss ebenfalls immer geöffnet und geschlossen werden.

```
<details>

   <summary>Widget-Name</summary>
   Der Inhalt des Widgets

</details>
```

In der Praxis sieht das dann so aus, dass der Name des *SUMMARY*-Elements in der Webseite zu sehen und anklickbar ist. Klickt der Besucher mit der Maus darauf, öffnet sich das Widget und zeigt den im *DETAILS*-Bereich erzeugten Inhalt an. Klickt man erneut auf das Widget, schließt es sich wieder. Das ist sicherlich eine tolle Idee, aber leider bisher nur Theorie. Kein Browser unterstützt diese Funktion oder versteht auch nur die Befehle. Aus diesem Grund kann hier als Beispiel nur eine Illustration vom W3C-Konsortium gezeigt werden.

```
<details>

<summary><label for="fn">Name & Extension:</label></summary>

<p><input type="text" id="fn" name="fn" value="Pillar Magazine.pdf">
<p><label><input type="checkbox" name="ext" checked> Hide extension</
label>

</details>
```

Abb. 3.10: Widgets per DETAILS-Befehl erstellen (Abbildung: W3C)

3.2.3 MENU – eigene Menüs in Webseiten erstellen

Jede Webseite benötigt ein Menü bzw. eine Navigation. Damit nun nicht jeder HTML-Autor aufs Neue in JavaScript, Flash oder einer anderen Sprache ein interaktives Menü programmieren muss, soll es hierfür in Zukunft einen eigenen Befehl geben. Er heißt schlicht *MENU*. Mit diesem Befehl deklarieren Sie einen Menübereich, in dem wiederum die einzelnen Elemente des Menüs stehen – also die Befehle. Der *MENU*-Befehl muss immer sauber geschlossen werden.

Ihnen stehen dabei drei Arten von Menütyp zur Verfügung, die Sie mit dem Attribut *TYPE* festlegen.

- **TOOLBAR** – Hiermit erstellen Sie eine Menüleiste, wie man sie von den meisten Windows-Programmen her kennt.
- **CONTEXT** – Ein Kontextmenü ist ein Menü, das sich öffnet, wenn man ein bestimmtes Objekt anklickt. Ein typisches Beispiel ist der Rechtsklick unter Windows.
- **LIST** – Dieses Menü stellt eine einfache Liste dar, die beim Anklicken geöffnet wird.

```
<menu type="toolbar">

   Die Inhalte des Menüs

</menu>
```

Jetzt muss das Menü natürlich noch mit Inhalten gefüllt werden. Hier kommt wieder der Befehl *MENU* zum Einsatz. Durch die Verschachtelung werden innerhalb der zuvor angelegten Menüumgebung die Oberpunkte festgelegt. Wichtig ist dabei, dass jedem Oberpunkt mit dem Attribut *LABEL* ein Name gegeben wird. Dieser erscheint dann auch im Browser.

```
<menu type="toolbar">

   <menu label="Datei"></menu>
   <menu label="Bearbeiten"></menu>
   <menu label="Hilfe"></menu>

</menu>
```

In diesem Beispiel mit den Oberpunkten ***Datei***, ***Bearbeiten***, ***Hilfe*** wird das Menü einer typischen Windows-Anwendung nachempfunden. Auch wenn das für eine Homepage eher nicht infrage kommt, wurden diese Punkte bewusst gewählt, damit der Aufbau des Menüs sofort klar ist. Innerhalb dieser drei Oberpunkte müssen nun natürlich Unterpunkte bzw. Funktionen integriert werden. In Windows wären das z. B. ***Öffnen***, ***Schließen***, ***Einfügen***, ***Optionen*** usw.

Das Problem ist hierbei, dass der HTML 5-Standard noch keine konkreten Befehle bereitstellt, wie das auszusehen hat. Derzeit wird vorgeschlagen, die Menüs über Listen-Elemente zu erzeugen. Sie stellen ein zuverlässiges Mittel zur Strukturierung dar und werden von allen Browsern unterstützt. In Zukunft könnten die Browser also Listen-Elemente innerhalb eines *MENU*-Bereichs wie klassische Menüs öffnen und schließen. Das allein reicht aber nicht aus, denn HTML-Befehle sind nicht geeignet, um tatsächliche Aktionen auszulösen. Hierzu müsste wieder JavaScript integriert werden. Zukünftige HTML 5-Menüs stellen also eine Mischung aus HTML, dem *MENU*-Befehl, Listen-Elementen und JavaScript dar.

Ein möglicher HTML-Code könnte wie folgt aussehen:

```
<menu type="toolbar">

<li>
<menu label="File">
<button type="button" onclick="fnew()">New...</button>
<button type="button" onclick="fopen()">Open...</button>
<button type="button" onclick="fsave()">Save</button>
<button type="button" onclick="fsaveas()">Save as...</button>
</menu>
</li>

<li>
<menu label="Edit">
<button type="button" onclick="ecopy()">Copy</button>
<button type="button" onclick="ecut()">Cut</button>
<button type="button" onclick="epaste()">Paste</button>
</menu>
</li>

<li>
<menu label="Help">
<li><a href="help.html">Help</a></li>
<li><a href="about.html">About</a></li>
</menu>
</li>

</menu>
```

Leider kann mit dieser Idee derzeit noch kein Browser umgehen. Sie ignorieren den *MENU*-Befehl einfach und versuchen, aus den Listen- und Button-Elementen eine herkömmliche Auflistung zu erstellen. Diese sieht recht merkwürdig aus und ist natürlich völlig funktionslos. Aus diesem Grund stellen dieser Beispielcode und die folgende Abbildung derzeit nur Illustrationen dar. Sie wurden der Dokumentation des W3C-Konsortiums entnommen.

Abb. 3.11: Richtige Menüs mit HTML 5 erstellen (Abbildung: W3C)

3.2.4 COMMAND – Befehle in Menüs erstellen

Mit dem Befehl *MENU* soll es in Zukunft möglich sein, richtige Menüs in HTML zu erstellen. Doch bisher gibt es keinerlei Möglichkeit, wirklich ein Kommando oder einen Befehl mit solchen Menüs abzuschicken. Dazu ist bis jetzt unbedingt JavaScript notwendig, das die einzige Möglichkeit ist, um HTML irgendwie interaktiv zu machen. Daran ändert auch der *COMMAND*-Befehl nur bedingt etwas, weil auch er weiterhin auf JavaScript angewiesen ist. Aber er erleichtert die Handhabung und zeigt bereits deutlich konkreter, wie ein Menü in HTML 5 einmal aussehen und funktionieren könnte.

Der *COMMAND*-Befehl kann nur innerhalb einer *MENU*-Umgebung vorkommen. Setzen Sie ihn in Ihrem Dokument an eine andere Stelle, z. B. im Fließtext, soll er von den Browsern ignoriert werden. Der *COMMAND*-Befehl steht für sich alleine und muss nicht extra geschlossen werden.

Folgende Attribute stehen Ihnen dabei zur Verfügung:

- **TYPE** – Legen Sie die Art des Kommandos bzw. des Menüpunktes fest. Mit dem Wert *COMMAND* geben Sie an, dass ein »normales« Kommando folgt, das direkt ausgeführt werden kann. Mit dem Wert *CHECKBOX* erzeugen Sie einen Kasten, der durch einen Haken aktiviert oder deaktiviert werden kann. Der Wert *RADIO* erzeugt eine Auswahlliste, in welcher der gewünschte Punkt durch Anklicken ausgewählt werden kann.
- **LABEL** – Geben Sie diesem Befehl bzw. Kommando einen Namen. Dieser Name erscheint auch im Menü als Anzeige.
- **ICON** – Möchten Sie ein grafisches Menü erstellen, legen Sie hiermit das Symbol für diesen Befehl fest. Sie können direkt den Namen einer Grafikdatei angeben, z. B. im JPG-, GIF- oder PNG-Format.
- **RADIOGROUP** – Haben Sie sich beim TYPE für den Wert *RADIO* entschieden, legen Sie hiermit den Namen für diese Gruppe von Radio-Buttons fest. So werden sie als eine Einheit erkannt und verarbeitet.

Leider hat der *COMMAND*-Befehl dasselbe Problem wie auch der ihm übergeordnete *MENU*-Befehl – er wird derzeit von keinem Browser verstanden oder gar verarbeitet. Deshalb ist das im Moment leider nur Theorie, aber eine schöne Aussicht für die Zukunft mit angepassten Browsern. Um Ihnen die mögliche Funktionsweise zu verdeutlichen, zeigt der folgende Code ein Menü mit allen beschriebenen Attributen. Es handelt sich um ein Beispiel des W3C-Konsortiums.

```
<menu type="toolbar">

<command type="radio" radiogroup="alignment"
checked="checked" label="Left" icon="icons/alL.png"
onclick="setAlign('left')" >

<command type="radio" radiogroup="alignment"
label="Center" icon="icons/alC.png"
onclick="setAlign('center')" >

<command type="radio" radiogroup="alignment"
label="Right" icon="icons/alR.png"
onclick="setAlign('right')" >

<command type="command" disabled
label="Publish" icon="icons/pub.png"
onclick="publish()" >

</menu>
```

3.2.5 CANVAS – der interaktive Multimedia-Container

HTML ist eine Seitenbeschreibungssprache. Damit lassen sich die Inhalte eines Dokuments strukturieren. Multimediale Inhalte, interaktive Objekte und richtige Programme sind damit nicht möglich. Wer solche Inhalte erzeugen möchte, ist zwingend auf Adobe Flash oder Java-Applets angewiesen. Das war der unabänderbare Zustand der letzten Jahre. Das hat dazu geführt, dass Adobe Flash ein Quasi-Monopol im Internet erlangt hat. Überall werden Spiele, Menüs, Anwendungen, Videos und viele andere Inhalte per Flash erzeugt und dann mit ein paar Befehlen eingebunden.

Das möchte man nun mit HTML 5 ändern. Es soll nun endlich eine Schnittstelle geben, mit der sich solche Inhalte direkt im HTML-Dokument erzeugen lassen. Der neue Befehl dafür lautet *CANVAS*, was so viel wie »Leinwand« oder »Projektionsfläche« bedeutet. Der Name verdeutlicht also sofort, dass hiermit ein Bereich innerhalb eines HTML-Dokuments geschaffen werden soll, der völlig neuartige Anwendungen erlaubt.

Die Handhabung ist überraschend einfach. Öffnen und schließen Sie mit dem Befehl *CANVAS* einen Multimedia-Bereich.

```
<canvas>

 ... Multimedia ...

</canvas>
```

Damit der Browser weiß, wie groß die »Leinwand« sein soll, müssen Sie mit den Attributen *WIDTH* und *HEIGHT* die Größe in Pixeln angeben.

```
<canvas width="400" height="300">

 ... Multimedia ...

</canvas>
```

Damit auf den *CANVAS*-Bereich besser zugegriffen werden kann, sollten Sie ihm einen eindeutigen Namen geben. Hierzu dient das Attribut *ID*. Wählen Sie einen eindeutigen Namen in Kleinbuchstaben. Verwenden Sie keine Leerzeichen, Sonderzeichen oder deutsche Umlaute.

```
<canvas id="browser_game" width="400" height="300">

 ... Multimedia ...

</canvas>
```

So einfach und unkompliziert ist die neue Multimedia-Umgebung erstellt. Doch jetzt stellt sich sofort die Frage, wie es denn nun weitergeht. Wie bekommt man die schönen neuen Inhalte in den *CANVAS*-Bereich? Das ist zum allergrößten Teil Ihnen überlassen. HTML selbst stellt nach wie vor keine Befehle bereit, um interaktive Inhalte zu erzeugen. Natürlich nicht, denn HTML ist keine Programmiersprache, um damit z. B. ein Spiel zu erstellen. Hierfür benötigen Sie weiterhin Programmiersprachen wie z. B. JavaScript.

Ein ganz simples Beispiel könnte wie folgt aussehen: Zunächst wird der *CANVAS*-Bereich deklariert. Innerhalb des Bereichs wird ein *SCRIPT*-Bereich erzeugt. Dieser weist den Browser an, zwei farbige Rechtecke zu zeichnen.

```
<canvas width="400" height="300" id="zeichnen">

<script>

var canvas = document.getElementById('zeichnen');
   if (canvas.getContext) {
        var ctx = canvas.getContext('2d');
        ctx.fillStyle = 'rgba(155, 111, 111, 0.5)';
        ctx.fillRect(40, 40, 160, 120);
        ctx.fillStyle = 'rgba(120, 126, 320, 0.5)';
        ctx.fillRect(140, 120, 160, 120);
}
</script>

</canvas>
```

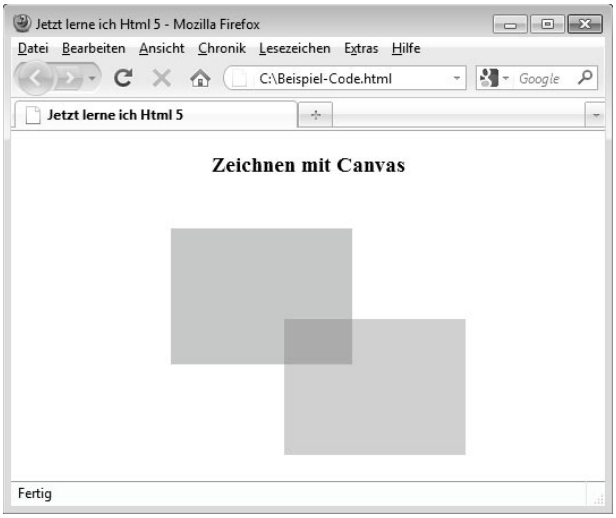

Abb. 3.12: Eigene Objekte im CANVAS-Bereich erstellen

Natürlich sieht das noch ziemlich unspektakulär aus und hat nicht den Anschein von Multimedia. Es stellt sich auch die Frage, wo hier der Vorteil bestehen soll. Der liegt eindeutig in der Unabhängigkeit der Programmierer und der Anwendungen. HTML und JavaScript sind Standards, die jedem kostenfrei zur Verfügung stehen. Sie müssen nicht extra eine teure Flash-Entwicklungsumgebung von Adobe kaufen. Außerdem werden diese Anwendungen in Zukunft in jedem Browser laufen, ebenso auf jedem Betriebssystem und jedem Gerät – egal ob PC, Notebook, Handy oder Tablet. Sie benötigen auch kein Plug-In oder sonstige Software. Sie müssen niemanden um Erlaubnis fragen, Lizenzen bezahlen oder die Vorgaben eines Shop-Betreibers befolgen. Programmieren Sie mit freien Standards alles, was Sie wollen. Das ist durchaus eine sehr interessante Perspektive für Programmierer.

Wenn man bedenkt, dass HTML 5 noch gar nicht verabschiedet ist und die ersten Programmierer gerade mit Canvas und JavaScript herumspielen, kann man für die Zukunft noch so einiges erwarten. Bereits jetzt gibt es die ersten Übungen, z. B. in Form von kleinen Spielen. Diese reichen bereits locker an Flash-Spiele heran – und das ist erst der Anfang.

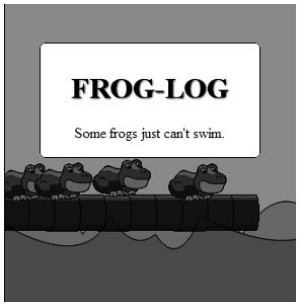

Abb. 3.13: Frogger im Canvas-Container (Jimmy D. http://mix10k.visitmix.com)

3.3 Auszeichnungen für Einzelelemente

HTML bietet viele Möglichkeiten, um einzelne Wörter oder ganze Sätze gezielt hervorzuheben. Dabei geht es aber nicht darum, diese mit einer bestimmten Formatierung oder einem Layout auszustatten, sondern sie als wichtig oder als andersartig zu markieren. Wie diese Wörter beim Leser dargestellt werden, hängt sehr vom verwendeten Browser ab. Zusätzlich haben Sie später natürlich die Möglichkeit, bestimmte Markierungen per CSS nach Ihren Wünschen zu formatieren.

Wo sind die ganzen Formatierungen geblieben?

Haben Sie vorher mit HTML 4 gearbeitet, wird Ihnen in diesem Abschnitt auffallen, dass viele Auszeichnungen und Formatierungen in HTML 5 nicht mehr vorhanden sind. Sie wurden ersatzlos gestrichen. Ein paar haben sich auch in ihrer Bedeutung verändert, z. B. das *B*- und das *I*-Tag. Das liegt einfach daran, dass man in HTML 5 vieles einfacher machen wollte. Deshalb wurden viele ähnliche oder fast identische Befehle verworfen. Sämtliche Befehle, die lediglich eine optische Formatierung mit sich brachten, ohne dabei eine logische Auszeichnung darzustellen, wurden ersatzlos gestrichen. Formatierungen finden jetzt nur noch in CSS statt und nicht mehr im HTML-Code selbst.

3.3.1 EM & STRONG – wichtige Worte betonen

In einem Text ist es oft notwendig, einzelne Worte oder einen Satz besonders zu betonen. Um das zu tun, steht Ihnen das Tag *EM* (Emphasis = Betonung) zur Verfügung. Alle Browser heben die so markierten Wörter deutlich hervor, meist durch einen Fettdruck oder kursive Darstellung.

```
<p>
Warten Sie, bis die Ampel auf <em>Grün</em> springt.
</p>
```

Soll ein Wort oder ein Satz noch stärker betont werden, wählen Sie das Tag *STRONG*. Es stellt eine Steigerung zum *EM*-Tag. Die meisten Browser unterscheiden hier allerdings nicht und stellen diesen Text ebenfalls einfach nur fett oder kursiv dar.

```
<p>
Tun Sie das bitte <strong>auf keinen Fall</strong>.
</p>
```

Mit den Befehlen *EM* und *STRONG* soll einem Wort also eine mehr oder weniger große Wichtigkeit gegeben werden. Zeichnen Sie niemals längere Sätze mit diesen Befehlen aus, denn das wird im Internet als Schreien angesehen. Verwenden Sie die Befehle auch nicht, um ein Wort aus rein optischen Gründen fett darstellen zu lassen. Der Fettdruck ist im Standard nicht vorgeschrieben, sodass ein Browser auch eine andere Methode zur Darstellung wählen kann.

Abb. 3.14: Wichtige Wörter hervorheben

3.3.2 B – Wörter hervorheben

Möchten Sie ein Wort im Text hervorheben, ohne ihm eine besondere Wichtigkeit zu geben, steht Ihnen der *B*-Befehl zur Verfügung. Damit lassen sich z. B. Stichwörter, Eigennamen, Produkte oder andere Elemente markieren. Beachten Sie hierbei, dass der B-Befehl in HTML 4 für den Fettdruck (bold = fett) zuständig war. Das ist in HTML 5 nicht mehr so. Es sollen lediglich Worte hervorgehoben werden – das kann jeder Browser machen, wie er will. Allerdings behalten die meisten Browser den Fettdruck aus reiner Gewohnheit bei.

```
<p>
Viele Browser unterstützen <b>HTML 5</b> noch nicht vollständig.
</p>
```

```
<p>
Bitte rufen Sie <b>Sally</b> an und bestätigen Sie den <b>15 Uhr
Termin</b>.
</p>
```

3.3.3 I – Stimmungen, Dialoge & Eigennamen

Soll in einem Text eine andere Stimmung, ein Dialog oder eine andere Abgrenzung vom Haupttext stattfinden, steht Ihnen der *I*-Befehl zur Verfügung. In HTML 4 wurde mit dem *I*-Befehl die kursive Darstellung herbeigeführt (italic = kursiv). In HTML 5 ist das nicht mehr so, und der I-Befehl ist für die genannten Textarten reserviert. Die meisten Browser setzen den Text zwar weiterhin kursiv, aber das ist vom HTML-Standard nicht mehr vorgeschrieben. Eine rein optische kursive Darstellung sollte per CSS erzeugt werden.

```
<p>
Darauf sagte Fred <i>"Das ist eine gute Idee!"</i> und ging aus dem
Raum.
</p>

<p>
Die besten Computerbücher gibt es bei <i>Markt + Technik</i>.
</p>
```

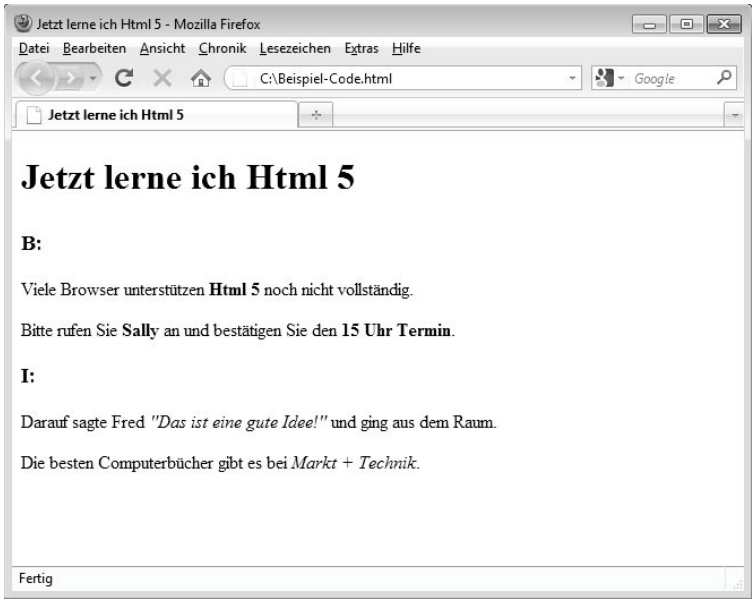

Abb. 3.15: Mit B oder I Worte und Dialoge hervorheben

3.3.4 MARK – wichtige Wörter farblich markieren

HTML 5 kennt noch eine besonders auffällige Methode, um wichtige Wörter oder ganze Sätze hervorzuheben. Der *MARK*-Befehl funktioniert ganz ähnlich wie ein echter Textmarker. Die Stellen werden also farblich markiert. Obwohl der Standard es nicht vorschreibt, verwenden die meisten Browser hierfür Gelb – vermutlich weil es die typische Textmarker-Farbe ist. Der HTML-Standard empfiehlt, den *MARK*-Befehl für wiederkehrende Worte zu verwenden oder für Begriffe, die im späteren Verlauf des Texts interessant sind. Rein gestalterische Gründe sollte man auch hier vermeiden.

```
<p>
Verwenden Sie für die Formatierung ausschließlich <mark>Cascading
Stylesheets</mark>. So bleibt der Code sauber und <mark>CSS</mark>
bietet viel bessere Gestaltungsmöglichkeiten. Mehr zum Thema
<mark>CSS</mark> finden Sie ab Seite...
</p>
```

Abb. 3.16: Dialoge und Eigennamen

3.3.5 SUP & SUB – Wörter hoch- oder tiefstellen

In HTML ist es möglich, einzelne Buchstaben, Ziffern oder auch ganze Wörter hochgestellt zu formatieren. Hierzu dient der *SUP*-Tag (superscript = hochgestellt). Soll hingegen ein Buchstabe oder ein Wort tiefgestellt werden, verwenden Sie das Tag *SUB* (subscript = tiefgestellt). Beide Befehle sind im Moment etwas schwammig in der Anwendung, weil der HTML-Standard verlangt, dass sie für die Lesbarkeit des Texts zwingend notwendig sein müssen. Für mathematische Formeln sollten die beiden Befehle ebenfalls nicht verwendet werden. Da bleibt nicht mehr viel übrig, sodass es Ihrem Urteil überlassen bleibt, wann der Einsatz sinnvoll ist und wann nicht.

```
<p>
Dieser Text ist <sup>hochgestellt</sup> und wird etwas kleiner
angezeigt.
</p>
```

```
<p>
Dieser Text ist <sub>tiefgestellt</sub> und wird etwas kleiner
angezeigt.
</p>
```

3.3.6 SMALL – Wörter kleiner schreiben

Manchmal ist es praktisch, wenn nicht so wichtige Elemente eines Texts kleiner dargestellt werden. Markieren Sie ein Wort oder einen Satz mit dem *SMALL*-Tag (= klein), werden diese ein wenig kleiner dargestellt als der Rest des Texts. Der HTML-Standard empfiehlt den Befehl für kurze Anmerkungen oder Randinfor-

mationen. Das kann ein Urheberrechtshinweis sein, eine Fußnote oder eine Randnotiz. Für längere Texte, die Teil des Hauptdokuments sind, sollte der *SMALL*-Befehl hingegen nicht verwendet werden. Weil die meisten Browser ihn eher lieblos und unschön verarbeiten, macht er in der Praxis nur wenig Sinn.

```
... Text ...
... Text ...
<p><small>Copyright 2010 Autor und Verlag</small></p>

Dieser Tag ist <small>unwichtig</small> und kommt kaum noch zur
Anwendung.
```

Abb. 3.17: Hoch- oder Tiefstellen sowie Kleinschrift

3.3.7 TIME – Zeit und Datum markieren

Sucht man im Internet nach Informationen, sollen diese natürlich möglichst aktuell sein. Damit der Leser, die Suchmaschine oder der Webserver nun nicht mehr raten müssen, von wann genau ein Dokument ist, gibt es nun einen eigenen *TIME*-Befehl.

Er ist nicht darauf begrenzt, das Erstellungsdatum eines Dokuments zu markieren. Sie können damit auch einen Stundenplan markieren, das Datum auf einer Einladung, die Uhrzeit für eine Sitzung oder Ähnliches. Wichtig ist dabei, dass es sich um eine konkrete Angabe handeln muss. Markieren Sie ein Datum oder eine Uhrzeit, aber nicht eine Angabe wie »in 5 Tagen« oder »nächste Woche«.

Zusätzlich haben Sie Möglichkeit, mit dem Attribut *DATETIME* ein standardisiertes Datumsformat mit zu codieren. Das erleichtert vor allem Suchmaschinen und Content-Management-Systemen den Umgang mit diesen Daten.

```
<p>
Am <time datetime="2010-10-31">31.10.2010</time> ist Halloween, die
Nacht der Geister und Vampire!
</p>
```

```
<p>
Die diesjährige Konferenz findet am kommenden <time datetime="2010-
08-13">Freitag, 13. August 2010, in Park Hotel statt. Beginn ist um
<time>11:00</time> Uhr.
</p>
```

3.3.8 ABBR – Abkürzungen markieren

Mit dem Befehl *ABBR* lassen sich Abkürzungen (= abbreviation) im Text markieren. Hierfür stehen Ihnen zwei Möglichkeiten zur Verfügung. Im einfachsten Fall markieren Sie nur die Abkürzung an sich. Möchten Sie auch gleich das vollständige Wort mit angeben, nutzen Sie zusätzlich das Attribut *TITLE*. Das kann sinnvoll sein, wenn die Seite durch Suchprogramme gründlich bearbeitet werden soll.

```
<p>
Der Begriff <abbr>HTML</abbr> steht für Hypertext Markup Language.
</p>
```

```
<p>
Sie benötigen <abbr title="Hypertext Markup Language">HTML</abbr>, um
eine eigene Webseite zu erstellen.
</p>
```

3.3.9 DFN – Definitionen deklarieren

Mit dem Befehl *DFN* lassen sich in einem HTML-Dokument Definitionen markieren. Das ist in wissenschaftlichen Texten sehr sinnvoll, wenn Sie einen sehr technischen Artikel schreiben oder sich mit einem anderen Fachthema befassen. Auf diese Weise lassen sich wichtige Formulierungen und Fachausdrücke markieren und beschreiben. Der *DFN*-Befehl kann sehr gut zusammen mit Abkürzungen per *ABBR*-Befehl verwendet werden.

```
<p>
Bei einer <dfn>Spiegelreflexkamera</dfn> lenkt ein Spiegel das
Objektivbild in den Sucher um. Beim Auslösen klappt er hoch und lässt
die Projektion auf den <dfn>Bildsensor</dfn> oder Film fallen.

Bei einer <dfn><abbr title="Digital Single Lens Reflex">DSLR-Kamera</
abbr></dfn> wurde dieses Prinzip vollständig übernommen und lediglich
in die digitale Welt übertragen.
</p>
```

Abb. 3.18: Abkürzungen und Definitionen

3.3.10 CODE, VAR, SAMP & KBD – Computertexte

Es kommt immer mal wieder vor, dass auf einer HTML-Seite Computertexte beschrieben werden. Das kann eine Anleitung zum Programmieren sein, ein HTML-Workshop oder Hilfe zu einem Windows-Problem. Um solche Informationen besser sortieren zu können, gibt es dafür eigene Befehle.

Programmcode – HTML kennt einen eigenen Befehl für das Deklarieren von Quellcode. Er lautet schlicht *CODE*. Dabei kann es sich um jede beliebige Art von Code handeln, z. B. HTML, C++, Java, PHP oder jede andere Sprache. Die meisten Browser zeigen diesen Text etwas anders an, meist in der Schriftart Courier.

```
<p>Der Code lautet:</p>
<code>
 {
   printf ("hello, world");
   return 0;
 }
</code>
```

Variablen – Möchten Sie in einem Text einzelne Variablen markieren, verwenden Sie den Befehl *VAR*. Er kommt vor allem beim Beschreiben von Programmcode zum Einsatz.

```
<p>
Verwenden Sie die Variablen <var>$a</var> und falls nötig noch
<var>$b</var>.
</p>
```

Beispieltexte – Möchten Sie einen Text als Beispiel deklarieren, können Sie das mit Befehl *SAMP* tun. Das kann praktisch sein, wenn Sie über Programmierung oder Kommandozeilen schreiben und dabei eine Ein- oder Ausgabe als Beispiel deklarieren möchten.

```
<p>
Unter DOS geben Sie den Befehl wie folgt ein:
<samp>chkdsk C: /F /R</samp>
</p>
```

Tastatureingaben – Mit dem Befehl KBD lassen sich Tastatureingaben deklarieren (keyboard = Tastatur). Wie bei den anderen Befehlen lassen sich damit Tasten oder Texteingaben vom anderen Text abgrenzen, was vor allem bei technischen Texten Sinn macht. Wie dies im Browser angezeigt wird, hängt vom jeweiligen Browser ab.

```
<p>
Betätigen Sie die Taste <kbd>Enter</kbd> nach Ihrer Eingabe.
</p>
```

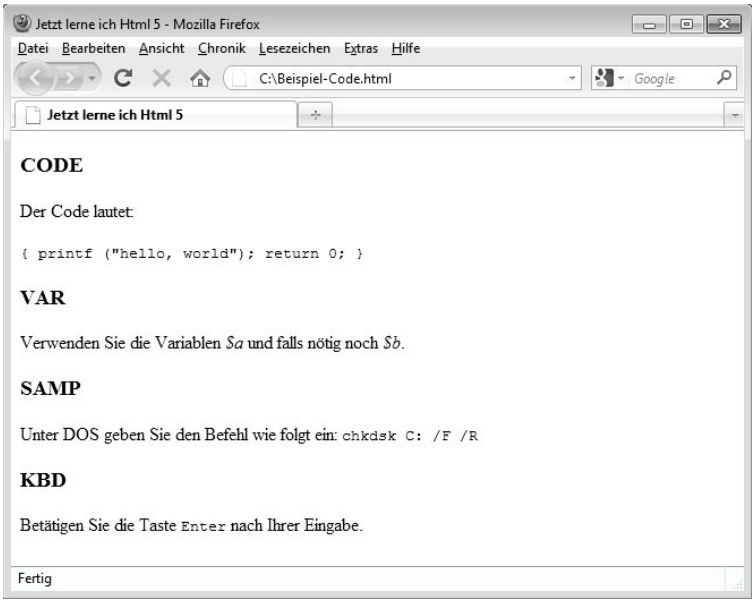

Abb. 3.19: Heben Sie Programmiercode hervor.

3.3.11 RUBY – Aussprache für asiatische Texte

Auch wenn der Gedanke beim Thema HTML naheliegt, so hat der Befehl *RUBY* nichts mit der Skriptsprache Ruby zu tun. Stattdessen besitzt HTML 5 spezielle Befehle, die eine Art Ausspracheanleitung für chinesische oder japanische Schriftzeichen darstellen. Mit dem Befehl *RUBY* öffnen und schließen Sie den Bereich. Mit dem Tag *RT* markieren Sie die Anmerkungen zur Aussprache, und mit *RP* fügen Sie optionale Klammern ein, falls ein Browser mit den Zeichen/Befehlen nicht umgehen kann.

```
<ruby>
<rp> ( </rp>  <rt>ZEICHEN</rt>  <rp> ) </rp>
<rp> ( </rp>  <rt>ZEICHEN</rt>  <rp> ) </rp>
</ruby>
```

Auch wenn es nichts mit chinesischen oder japanischen Schriftzeichen zu tun hat, soll an dieser Stelle noch ein weiterer Befehl für fremde Sprachen erwähnt werden. Mit *BDO* lässt sich die Leserichtung des Texts definieren. Hierzu besitzt der Befehl das Attribut *DIR* (direction = Richtung). Geben Sie ihm den Wert *LTR* (left to right = links nach rechts) oder *RTL* (right to left = rechts nach links).[1]

```
<bdo dir="rtl">بسم الله الرحمن الرحيم</bdo>
<bdo dir="ltr">Jetzt lerne ich HTML 5 – von Anfang an!</bdo>
```

3.3.12 INS & DEL – Überarbeitungen markieren

Möchten Sie den Text auf Ihrer Webseite verändern, überschreiben Sie in der Regel einfach den alten. Möchten Sie aber bei wichtigen Texten die Überarbeitung festhalten oder als solche markieren, helfen Ihnen zwei Überarbeitungsbefehle.

Mit dem Befehl *INS* deklarieren Sie Wörter oder Sätze als nachträglich eingefügt. Mit dem Befehl *DEL* lassen sich Textabschnitte oder einzelne Worte als gelöscht markieren. Für den Browser sind sie dann nicht mehr Teil des Textes, werden aber in durchgestrichener Form weiterhin angezeigt.

```
Das Treffen findet am Montag um 8 Uhr <ins>in Saal 3</ins> statt.

Bitte bringen Sie Ihre Unterlagen, das Ausbildungsbuch,
<del>Gummibärchen,</del> Papier und Bleistift mit.
```

Abb. 3.20: Überarbeitungen markieren

1. Das arabische rechtsläufige Textbeispiel lautet in etwa: »Im Namen des barmherzigen und gnädigen Gottes«.

Textauszeichnungen, die es nicht mehr gibt

Zu den größten Entwicklungen in HTML 5 gehört sicherlich, dass der HTML-Code keinerlei Layout- und Formatanweisungen mehr besitzt. Alle gestalterischen Eigenschaften müssen in CSS ausgelagert werden. In HTML 4 gab es viele wichtige Befehle für die Formatierung, die nun alle ersatzlos gestrichen wurden. Auf ein paar besonders häufig genutzte Befehle werden Sie sicherlich weiterhin treffen – entweder weil noch nicht jeder in HTML 5 schreibt oder weil sich viele nicht mit dem Wegfall dieser Befehle abfinden wollen. Damit Ihnen diese Befehle nicht völlig fremd sind, sollen sie hier kurz aufgeführt werden. Verwenden Sie diese auf keinen Fall, sondern löschen Sie die Befehle aus dem Text.

Seiten-Layout – Innerhalb des *BODY*-Tags ließen sich mit den Befehlen *BGCOLOR* und *BACKGROUND* für die Seite Hintergrundfarben und Hintergrundbilder festlegen. Mit *MARGINWIDTH*, *MARGINHEIGHT*, *TOPMARGIN* und *MARGINLEFT* wurde der Seitenrand eingestellt.

Ausrichtung – Es gab früher eine Menge Tags, mit denen sich Texte und alle anderen Elemente direkt ausrichten ließen. Dazu dienten die Befehle *ALIGN*, *LEFT*, *RIGHT*, *JUSTIFY*, *CENTER* und einige mehr. Diese sind jetzt innerhalb von Elementen absolut verboten und müssen aus dem HTML-Code gelöscht werden. Sie dürfen nur noch in CSS verwendet werden.

Schriftformatierung – In HTML 4 gab es den *FONT*-Befehl. Er war sehr mächtig und erlaubte mit Attributen wie *FACE*, *SIZE*, *COLOR* oder *BASEFONT* eine sehr individuelle Formatierung von einzelnen Absätzen, Sätzen oder auch Wörtern. Das führte schnell zu einem HTML-Code voller Formatierungsbefehle, sodass der *FONT*-Befehl und alle damit verbundenen Tags ersatzlos gestrichen wurden. Für Umsteiger ist es gewöhnungsbedürftig, Schriften gar nicht mehr im HTML-Code formatieren zu können. Es geht mit CSS aber wirklich besser. Treffen Sie noch auf *FONT*-Befehle, übertragen Sie deren Eigenschaften in CSS, und löschen Sie diese aus dem HTML-Code.

3.4 Übungen

1. Erstellen Sie ein Feld für Ihre Kontaktinformationen, das Sie unter Ihre Artikel setzen können.
2. Erstellen Sie einen Textabschnitt, der nicht durch den Browser formatiert wird. Stattdessen soll dieser im Editor vorformatiert sein, z. B. als Programmcode.
3. Fügen Sie in Ihren HTML-Code zwei Kommentarzeilen ein.
4. Schreiben Sie ein kurzes Zitat in Ihr HTML-Dokument, das Sie als illustrierendes Element mit einer Beschreibung deklarieren.