Mit vielen Übungen

Florence Maurice

PHP 5.4 & MySQL 5.5

Der Einstieg in die Programmierung dynamischer Websites

ADDISON-WESLEY

ALWAYS LEARNING

PEARSON

PHP 5.4 & MySQL 5.5

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

10 9 8 7 6 5 4 3 2 1

14 13 12

ISBN 978-3-8273-3164-9 (Buch) ; 978-3-86324-512-2 (pdf) ; 978-3-86324-189-6 (ePub)

ein Imprint der Pearson Deutschland GmbH
Martin-Kollar-Straße 10–12, D-81829 München/Germany

www.pearson.de
Einbandgestaltung: Marco Lindenbeck, webwo GmbH (mlindenbeck@webwo.de)
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Fachlektorat: Frank Bueltge, bueltge.de; Ralph Steyer, rjs.de
Korrektorat: Brigitte Hamerski, Willich
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: Reemers Publishing Services GmbH, Krefeld (www.reemers.de)
Druck: Drukarnia Dimograf, Bielsko-Biala
Printed in Poland

Florence Maurice

PHP 5.4 & MySQL 5.5

Der Einstieg in die Programmierung dynamischer Websites

ADDISON-WESLEY

An imprint of Pearson

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

3 HTML und CSS – Grundlagen

Webbrowser verstehen kein PHP. Webbrowser verstehen nur HTML und CSS. PHP setzen Sie ein, um HTML-Dokumente zu erstellen – mit dynamischen Inhalten. Deswegen sind auch für die Arbeit mit PHP grundlegende HTML-Kenntnisse notwendig. Diese vermittelt das Kapitel in Kurzform und verrät Ihnen, wo Sie weitere Informationen finden.

3.1 Grundstruktur

HTML steht für Hypertext Markup Language und ist eine Auszeichnungssprache für Webseiten. HTML wird vom World Wide Web-Konsortium – kurz W3C (`http://www.w3.org/`) – betreut. Das W3C ist ein unabhängiges Gremium, dem auch Vertreter von Firmen angehören.

HTML dient dazu, die Struktur einer Seite zu definieren und festzulegen, worum es sich bei den einzelnen Bestandteilen handelt, ob etwas ein Absatz, eine Aufzählungsliste oder eine Überschrift ist. Diese Bestandteile werden dann standardmäßig vom Browser auf eine bestimmte Art formatiert angezeigt. Um diese Default-Formatierungen zu ändern, kommt eine andere Technik ins Spiel: die Cascading Stylesheets (CSS), die am Schluss dieses Kapitels besprochen werden.

HTML gibt es in mehreren Versionen und Varianten, so gibt es HTML 4.01, XHTML 1.0 und HTML5. Wir verwenden hier HTML5. Was die Unterschiede zwischen den unterschiedlichen Varianten sind, erfahren Sie in Abschnitt 3.6.

Jede HTML-Seite besteht aus Tags und normalem Text. Tags werden in spitzen Klammern geschrieben, und man unterscheidet zwischen Start- und Endtags. `<html>` ist ein Starttag, das besagt, dass jetzt HTML folgt. Dieses wird durch ein Endtag `</html>` geschlossen. Es gibt verschiedene Elemente, die auf bestimmte Art ineinander verschachtelt sein können, jedoch basieren alle HTML-Dokumente auf demselben Grundgerüst:

Listing 3.1: **Die Basis aller HTML-Dokumente: das HTML-Grundgerüst (*grundgeruest.html*)**

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04   <meta charset="UTF-8" />
05   <title>Grundgerüst</title>
06 </head>
07 <body>
08   <p>Erste Webseite!</p>
09 </body>
10 </html>
```

TIPP

Zu besserer Orientierung sind im Folgenden längere Listings, d. h. im Allgemeinen Listings ab 10 Zeilen, mit Zeilennummern versehen. Diese gehören natürlich nicht zum Code und Sie dürfen sie nicht mit abschreiben.

Ganz am Anfang des Dokuments in Zeile 1 steht die Dokumenttypangabe. Diese Zeile verrät dem Browser, welche Version von HTML in welcher Geschmacksrichtung eingesetzt wird. Im Listing ist es HTML5. Diese Zeile sorgt dafür, dass der Browser die Webseite so gut wie möglich – nämlich in seinem Standardmodus darstellt. Darauf folgt das `html`-Starttag.

Alle Dokumente bestehen aus einem `head`- und einem `body`-Bereich. Der `head`-Bereich beinhaltet Informationen über das Dokument wie beispielsweise den Zeichensatz. Außerdem steht innerhalb des `title`-Elements der Titel des Dokuments. Dieser wird in der Titelzeile ganz oben im Browser angezeigt.

Innerhalb von `<body>` und `</body>` platzieren Sie die eigentlichen Inhalte, die im Browser dargestellt werden. Im Beispiel ist es ein `p`-Element, das zur Kennzeichnung eines Absatzes steht. Darin steht der Text, der angezeigt wird.

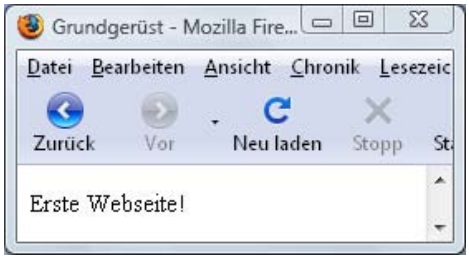

Abbildung 3.1: **Ein erstes Dokument im Browser**

Dieses Beispiel können Sie mit einem einfachen Texteditor erstellen. Geben Sie der Datei beim Speichern die Endung *.html*. Zum Betrachten mit dem Browser können Sie sie direkt durch Doppelklick auf den Dateinamen öffnen – etwas, das ja bei PHP-Dokumenten nicht geht: Diese müssen Sie immer beim lokalen Test über den Server `localhost` aufrufen.

TIPP

Bei den Dateinamen haben Sie an sich freie Hand: Verwenden Sie aber keine Sonderzeichen wie ü, ä, ö oder ß und auch keine Leerzeichen. Da die Groß- und Kleinschreibung bei Betriebssystemen wie Linux relevant ist und viele Server im Internet auf Linux laufen, sollten Sie sich am besten angewöhnen, alles kleinzuschreiben.

3.2 Inhalte strukturieren mit Überschriften, Absätzen und Listen

Das erste Beispiel war inhaltlich noch nicht sehr aufregend. Nun folgen weitere Elemente, die Sie zur Auszeichnung Ihrer Inhalte verwenden können. Das Element für Absätze `p` haben Sie bereits kennen gelernt. Möchten Sie nur einen Zeilenumbruch innerhalb eines Absatzes machen, so dient dafür `<br />`.

Zusätzlich gibt es vordefinierte Elemente für Überschriften. Es stehen sechs zur Verfügung: `h1` ist die Überschrift erster Ordnung, `h2` die Überschrift zweiter Ordnung etc. bis `h6`. Das folgende Listing führt die Überschriften `h1` bis `h3` vor. Es beinhaltet außerdem zwei Absätze und zeigt die Verwendung von `br` für Zeilenumbrüche.

Listing 3.2: **Überschriften und Absätze dienen zur Strukturierung von Dokumenten (*ueberschriften.html*).**

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04   <meta charset="UTF-8" />
05   <title>Strukturierungen</title>
06 </head>
07 <body>
08   <h1>Überschrift der ersten Ebene</h1>
09   <h2>Überschrift der zweiten Ebene</h2>
10   <h3>Überschrift der dritten Ebene</h3>
11   <p>Ein normaler Absatz<br />mit Zeilenumbruch</p>
12   <p>Ein normaler Absatz ohne Zeilenumbruch</p>
13 </body>
14 </html>
```

`<br />` wird verwendet, um einen Zeilenumbruch im Browser darzustellen. Leerzeichen und Zeilenumbrüche im HTML-Quellcode selbst hingegen werden vom Browser ignoriert. Sie sollten diese aber einsetzen, um Ihren Code übersichtlich zu gestalten.

Wie Sie sehen, werden Überschriften automatisch fett dargestellt und – zumindest die Überschriften der Ebenen eins bis drei – auch größer als der Text in Absätzen. Das sind die Defaultvorgaben des Browsers. Per CSS können Sie das ganz nach Belieben anders gestalten.

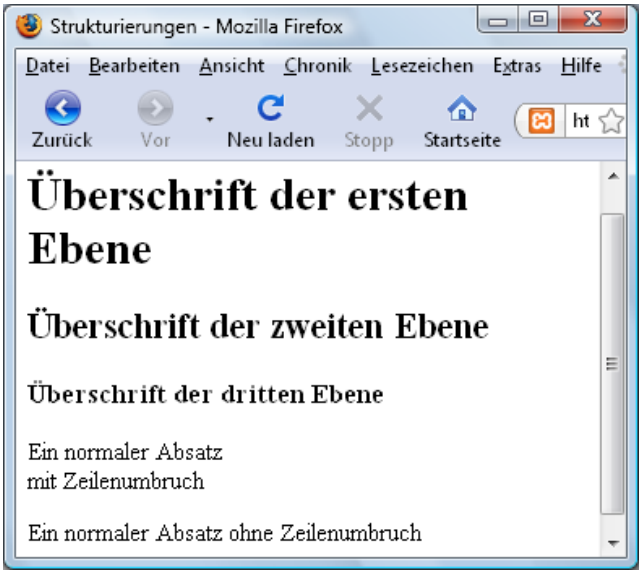

Abbildung 3.2: **Überschriften und Absätze in einem Dokument**

Suchen Sie eine Möglichkeit, einzelne Wörter oder Satzteile hervorzuheben, so sieht HTML hier zwei Elemente vor: `strong` und `em` (*emphasized*).

```
<em>betont</em>
<strong>stärker betont</strong>
```

ÜBUNG

Übung 1

Nehmen Sie das Dokument *ueberschriften.html* als Basis und ergänzen Sie weitere Überschriften und Absätze!

Die Lösungen zu den Übungen finden Sie im Anhang. Die Lösungsdateien stehen außerdem im Unterordner *loesungen* bei den Listings!

3.2.1 Aufzählungen

Weitere praktische Elemente sind Aufzählungen. Eine ungeordnete Liste erstellen Sie mithilfe des Elements `ul` (*unordered list*). Innerhalb von `<ul>` und `</ul>` stehen die einzelnen Aufzählungspunkte wiederum innerhalb von `<li>` und `</li>` (`li` = *list item*).

Für eine geordnete Liste, die automatisch vom Browser durchnummeriert wird, verwenden Sie anstelle von `ul` das Element `ol` (*ordered list*). Auch hier werden die einzelnen Punkte innerhalb von `<li>` und `</li>` notiert.

Listing 3.3: **Listen im Einsatz (*listen.html*)**

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="UTF-8" />
  <title>Listen</title>
</head>
<body>
<ol>
  <li>Telefonate führen</li>
  <li><em>E-Mails</em> beantworten</li>
  <li>Stadtbibliothek: bestelltes <strong>Buch</strong> abholen</li>
  <li>Einkäufe</li>
</ol>
<ul>
  <li>Brot</li>
  <li>Weichkäse</li>
  <li>Buttermilch</li>
</ul>
</body>
</html>
```

Das Beispiel zeigt außerdem den Einsatz von `em` und `strong` für Hervorhebungen. Normalerweise wird ein mit `em` markierter Teil kursiv, ein mit `strong` ausgezeichneter Textteil fett dargestellt. Aber auch das können Sie selbstverständlich per CSS ändern.

Bei komplexeren HTML-Dokumenten ist es wichtig, Hinweise unterzubringen, die den HTML-Code erläutern und einem helfen, später Modifikationen oder Ergänzungen vorzunehmen. Hierfür sind die Kommentare gedacht. HTML-Kommentare beginnen Sie mit `<!--` und beenden Sie mit `-->`. Alles, was zwischen `<!--` und `-->` steht, wird vom Browser ignoriert.

```
<!-- Beginn Kopfbereich -->
```

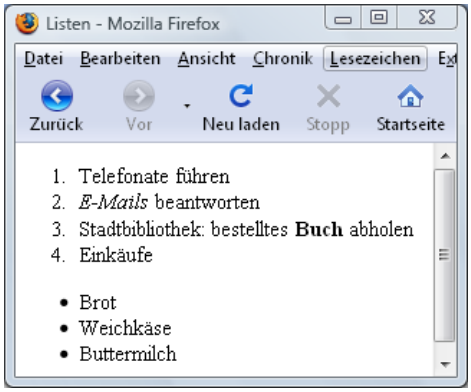

Abbildung 3.3: **Eine nummerierte Aufzählung und eine Liste**

3.3 Sonderzeichen und Zeichenkodierung

Wir haben bisher direkt Umlaute wie ü's und ä's geschrieben – geht das denn überhaupt gut, wenn jemand aus dem Ausland mit anderen Browser-Defaulteinstellungen auf unsere Seite geht? Ja, sofern Sie den Zeichensatz angeben, wie hier über eine `meta`-Angabe (Zeile 05 in Listing 3.3): Damit wählt der Browser automatisch die richtige Kodierung aus und auch die Sonderzeichen werden richtig dargestellt.

INFO

Die möglichen Zeichensätze werden später noch einmal bei der UTF-8-Unterstützung und den diesbezüglichen Besonderheiten von PHP besprochen (*Kapitel 6*).

Um ganz sicherzugehen, dass die Sonderzeichen unter allen Umständen korrekt angezeigt werden, können Sie aber auch die Entity-Schreibweise wählen. Entities beginnen immer mit einem `&`-Zeichen und enden mit einem Strichpunkt. Sie können den Wert des Zeichens dezimal oder hexadezimal angeben. Komfortabler und besser zu merken sind hingegen die benannten Entities. `&Uuml;` steht beispielsweise für das große Ü. `&Uuml;berschrift` gibt der Browser als »Überschrift« aus. Die wichtigsten Entities sehen Sie in Tabelle 3.1.

ZEICHEN	BENANNTE ENTITY	ZEICHEN	BENANNTE ENTITY
ä	`&auml;`	©	`&copy;`
Ä	`&Auml;`	(geschütztes Leerzeichen)	` `
ö	`&ouml;`	<	`<`
Ö	`&Ouml;`	>	`>`
ß	`&szlig;`	&	`&`
ü	`&uuml;`	'	`&apos;`
Ü	`&Uuml;`	"	`&quot;`
€	`&euro;`	©	`&copy;`

Tabelle 3.1: **Die wichtigsten Entities**

Wichtig sind die Entitities für die Zeichen < und &, da diese beiden eine Sonderbedeutung in HTML haben: < leitet ein Tag ein, & steht am Beginn eines Entities. Wenn Sie diese Zeichen im normalen Text verwenden möchten, müssen Sie die entsprechenden Entities verwenden.

» `4 < 5` sollten Sie also schreiben als `4 < 5`.

» Für `Panter, Tiger & Co` schreiben Sie `Panter, Tiger & Co`.

3.4 Verknüpfungen – Links und Bilder

Das Internet wäre nicht das Internet ohne Verlinkungen und Bilder.

3.4.1 Links

Für Links gibt es das Element `a`, abgekürzt von englisch *anchor*. Wichtig ist beim `a`-Element das Attribut `href` (*hypertext reference*). Als Wert von `href` geben Sie den Dateinamen an, auf den Sie verlinken wollen.

INFO

Attribute dienen zur weiteren Kennzeichnung von HTML-Elementen. In XHTML müssen die Werte von Attributen immer in Anführungszeichen stehen.

Der Text, den der Surfer sieht und auf den er klicken kann, um zum angegebenen Ziel zu kommen, steht innerhalb von `<a href="pfad">` und `</a>`:

```
<a href="ueberschriften.html">Dokument Überschriften aufrufen</a>
```

Befindet sich die Datei, auf die Sie verlinken wollen, in demselben Verzeichnis wie Ihr Dokument, geben Sie direkt den Dateinamen an. Steht die Datei hingegen in einem Unterverzeichnis, so schreiben Sie dieses zuerst und den Dateinamen nach einem Slash (`/`):

```
<a href="dateien/dokument.html">Dokument im Unterverzeichnis </a>
```

Möchten Sie hingegen von einem Unterordner in den übergeordneten Ordner wechseln, notieren Sie zwei Punkte:

```
<a href="../ueberschriften.html">einen Ordner höher</a>
```

Sie können selbstverständlich auch auf eine andere Website im Internet verweisen. Hierfür müssen Sie die vollständige URL mit Protokoll (`http://`) angeben:

```
<a href="http://www.addison-wesley.de/">Addison-Wesley</a>
```

Standardmäßig werden Links im selben Fenster geöffnet. Soll die ursprüngliche Seite erhalten bleiben und die neue sich in einem neuen Fenster/Tab öffnen, so ergänzen Sie beim Link ein `target="_blank"`:

```
<a href="http://www.addison-wesley.de/" target="_blank">Addison-Wesley</a>
```

Übrigens hängt es dann von den Browser-Einstellungen ab, ob die neue Seite in einem neuen Fenster oder einem neuen Tab geöffnet wird.

Schließlich gibt es noch Links auf E-Mail-Adressen. Falls auf dem Rechner ein Mailprogramm installiert ist, öffnet sich dieses dann per Mausklick und die Adresse des Adressaten ist schon voreingetragen. Hierfür notieren Sie `mailto:` vor der E-Mail-Adresse:

```
<a href="mailto:ich@mir.de">E-Mail an ich@mir.de</a>
```

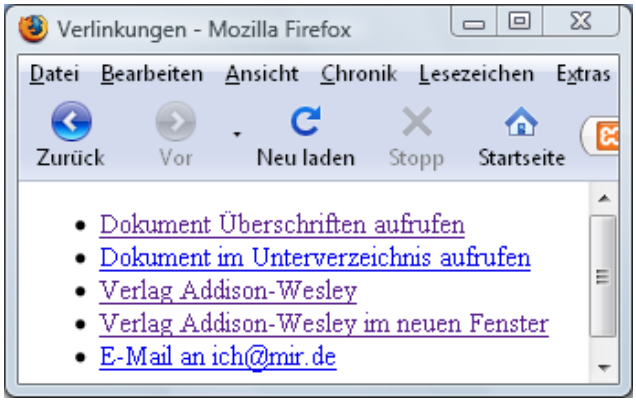

Abbildung 3.4: **Eine Liste von Links**

Listing 3.4 führt die einzelnen Linkarten noch einmal vor. Die Links werden – damit Sie sehen, wie Elemente verschachtelt werden können – innerhalb einer ungeordneten Liste dargestellt:

Listing 3.4: **Eine Liste mit Links (*verlinkungen.html*)**

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8" />
<title>Verlinkungen</title>
</head>
<body>
  <ul>
    <li><a href="ueberschriften.html">Dokument Überschriften aufrufen</a></li>
    <li><a href="dateien/weiteresdokument.html">Dokument im Unterverzeichnis
        aufrufen</a></li>
    <li><a href="http://www.addison-wesley.de/">Verlag Addison-Wesley</a></li>
    <li><a href="http://www.addison-wesley.de/" target="_blank">Verlag
        Addison-Wesley im neuen Fenster</a></li>
    <li><a href="mailto:ich@mir.de">E-Mail an ich@mir.de</a></li>
  </ul>
</body>
</html>
```

Links werden standardmäßig blau und unterstrichen dargestellt, die besuchten Links sind lila. All das können Sie per CSS abändern.

3.4.2 ... und Bilder

Das, was Sie gerade über die Pfade bei Links erfahren haben, gilt ebenfalls für Pfade zu Bildern. Wenn Sie in eine HTML-Seite ein Bild einfügen möchten, so binden Sie das Bild nicht ein, sondern schreiben nur einen Verweis auf das Bild und überlassen das Einbinden dem Browser. Hierfür ist das `img`-Element vorgesehen, dem Sie beim Attribut `src` den Pfad zum Bild als Wert zuweisen:

```
<img src="blumen.jpg" />
```

Außerdem sollten Sie noch die Breite (`width`) und die Höhe (`height`) in Pixeln angeben und einen alternativen Text spezifizieren. Dieser alternative Text wird angezeigt, falls das Bild nicht geladen werden kann:

```
<img src="blumen.jpg" width="90" height="120" alt="Blumen" />
```

Im folgenden Listing werden zwei Bilder eingebunden.

Listing 3.5: **Bilder einbinden per HTML (*bilder.html*)**

```
01 <!DOCTYPE html>
02
03 <html>
04 <head>
05 <meta charset="UTF-8" />
06 <title>Bilder</title>
07 </head>
08 <body>
09 <h1>Bilder einbinden</h1>
10 <img src="landschaft.jpg" width="90" height="120" alt="Landschaft" />
11 <img src="blumen.jpg" width="90" height="120" alt="Blumen" />
12 </body>
13 </html>
```

Abbildung 3.5: **Ein Dokument mit zwei Bildern und einer Überschrift**

ÜBUNG

Übung 2

Erstellen Sie ein neues HTML-Dokument mit folgendem Inhalt:

- **einer Überschrift (`h1`)**
- **einem Absatz**
- **einem Bild**
- **und einer ungeordneten Liste**
- **... außerdem soll ein Link dabei sein, der zu `http://php.net/` führt.**

Die Lösung zu dieser Übung finden Sie sowohl im Anhang als auch im *loesungen*-Ordner der Listings unter dem Namen *beispiel.html*.

3.5 Daten übersichtlich über Tabellen darstellen

Mit PHP kann man einfach auf Datenbanken zugreifen und die Inhalte nach bestimmten Kriterien gefiltert darstellen lassen. Die bevorzugte Darstellungsform für die Ausgabe von vielen Daten sind Tabellen. Sehen wir uns einmal an, wie man per HTML eine Tabelle erstellt.

Eine Tabelle setzt sich aus mehreren verschachtelten HTML-Elementen zusammen:

- `table` umfasst die gesamte Tabelle. Hier können Sie mit dem Attribut `border="1"` noch dafür sorgen, dass sichtbare Gitternetzlinien angezeigt werden. Das ist für den Entwurf praktisch, da dann Fehler besser zu erkennen sind. Sie können später bei Bedarf die Gitternetzlinien mit `border="0"` auch wieder ausschalten.
- `tr` (*table row*) umfasst jeweils eine Tabellenzeile.
- `td` steht für *table data*, umschließt den Inhalt der eigentlichen Tabellenzellen und wird innerhalb von `tr` notiert. Wenn Sie drei `td`-Elemente in einer `tr`-Zeile haben, so haben Sie eine dreispaltige Tabelle. Innerhalb von `td` steht der Inhalt der Zelle.
- Bei Spalten- oder Zeilenüberschriften sollten Sie für die einzelnen Zellen anstelle von `td` das Element `th` (*table header*) benutzen.

Das folgende Beispiel zeigt eine zweispaltige Tabelle mit drei Zeilen.

Listing 3.6: **Tabellengrundgerüst (*tabelle_grundgeruest.html*)**

```
01 <!DOCTYPE html>
02
03 <html>
04 <head>
05 <meta charset="UTF-8" />
06 <title>Tabelle - Grundgerüst</title>
07 </head>
08 <body>
09   <table border="1">
10     <tr>
11       <th>Abfahrt Hbf</th><th>Ankunft Katzenreuth</th>
12     </tr>
13     <tr>
14       <td>20:07</td><td>20:27</td>
15     </tr>
```

```
16     <tr>
17       <td>20:27</td><td>20:47</td>
18     </tr>
19   </table>
20 </body>
21 </html>
```

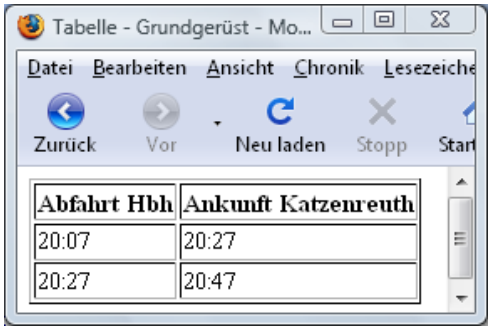

Abbildung 3.6: **So sieht die Tabelle im Browser aus.**

Über `rowspan` und `colspan` können Sie Tabellenzellen verbinden: `colspan` verbindet Spalten und `rowspan` Zellen. Dahinter geben Sie jeweils an, wie viele Spalten/Zellen verbunden werden sollen.

Das folgende Beispiel demonstriert das:

Listing 3.7: **Zellen verbinden (*zellen_verbinden.html*)**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<title>Zellen verbinden</title>
</head>
<body>
  <table border="1">
    <tr>
     <td>eins</td><td>zwei</td><td>drei</td>
    </tr>
    <tr><!--colspan verbindet Spalten -->
      <td colspan="3">vier</td>
    </tr>
    <tr><!--rowspan verbindet Zeilen -->
      <td>fünf</td><td rowspan="2">sechs</td><td>sieben</td>
    </tr>
    <tr>
      <td>acht</td><td>neun</td>
    </tr>
  </table>
</body>
</html>
```

Abbildung 3.7: **Zelle vier erstreckt sich über drei Spalten (colspan="3") und Zelle sechs erstreckt sich über zwei Zeilen (rowspan="2").**

Die Eigenschaften `cellpadding` und `cellspacing` sorgen für Abstand in der Tabelle, ein Beispiel hierzu finden Sie bei den Listings (*cellpadding_cellspacing.html*).

ÜBUNG

Übung 3

Ergänzen Sie beim Beispiel *tabelle_grundgeruest.html* zwei weitere Zeilen mit fiktiven Abfahrtszeiten. Fügen Sie dann zu Beginn noch eine Zeile ein, die sich über die beiden Spalten erstreckt und den Inhalt *Verbindungen* hat! Die Lösung finden Sie im Anhang und unter *loesungen/kapitel03/tabelle_erweitert.html*.

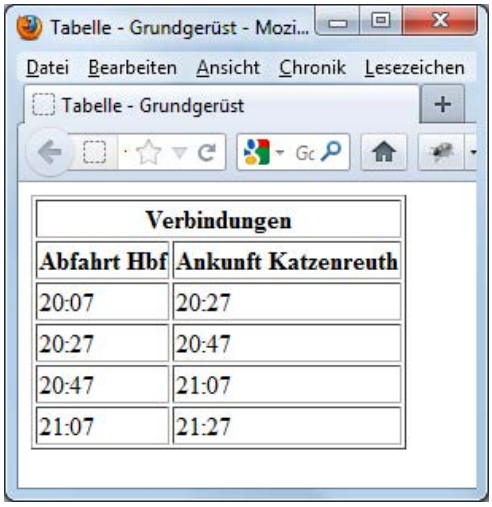

Abbildung 3.8: **Die erweiterte Tabelle**

3.6 HTML 4.01, XHTML 1.0 und HTML5

Einmal kam das Thema schon darauf, dass es unterschiedliche Versionen von HTML gibt, genau genommen sind es derer drei: HTML 4.01, XHTML 1.0 und HTML5. Und diese gibt es teilweise noch in unterschiedlichen »Geschmacksrichtungen«. Nun ist der richtige Zeitpunkt gekommen, diese genauer zu betrachten.

3.6.1 Der Klassiker – HTML 4.01

HTML 4.01 ist der Klassiker, der schon in die Jahre gekommen ist. HTML 4.01 stammt aus dem Jahr 1999. Auch wenn es schon so alt ist, funktioniert HTML 4.01 noch bestens und es wird bei vielen Dokumenten benutzt.

Das Beispiel mit den Bildern (*bilder.html*) sieht als HTML 4.01 etwa folgendermaßen aus:

Listing 3.8: **Das Beispieldokument als HTML 4.01 (***bilder_html4.html***)**

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/
    strict.dtd">
02 <html>
03  <head>
04    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
05 <title>Bilder</title>
06 </head>
07 <body>
08 <h1>Bilder einbinden</h1>
09 <img src="landschaft.jpg" width="90" height="120" alt="Landschaft" >
10 <img src="blumen.jpg" width="90" height="120" alt="Blumen" >
11 </body>
12 </html>
```

Im Vergleich zur HTML5-Variante gibt es ein paar Änderungen:

Der auffälligste Unterschied zu vorher ist die Dokumenttypangabe am Anfang des Dokuments.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/
    strict.dtd">
```

HINWEIS

Im Beispiel wurde HTML Strict benutzt, daneben gibt es noch den Typ Transitional. Der Unterschied zwischen beiden ist, dass HTML Strict strenger ist, d. h. bestimmte veraltete Elemente sind in der Strict-Variante nicht mehr erlaubt, in der Transitional-Variante hingegen schon.

Dann gibt es noch zwei kleine Besonderheiten – vielleicht sind Sie Ihnen ja aufgefallen. Die eine betrifft die Angabe des Zeichensatzes:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" >
```

In HTML5 ist eine verkürzte Schreibweise für den Zeichensatz möglich, wie wir sie eben verwendet haben. Hier sehen Sie die ausführliche. Und noch etwas ist dabei bemerkenswert: Das `meta`-Element, das hier zur Angabe eines Zeichensatzes dient, ist ein sogenanntes leeres Element.

In HTML 4.01 sehen leere Elemente genauso aus wie ein Starttag. Das heißt, sie haben kein `/>` am Ende. Das sehen Sie auch beim `img`-Element:

```
<img src="landschaft.jpg" width="90" height="120" alt="Landschaft" >
```

HTML hat einige weitere Besonderheiten. Eine davon ist, dass es möglich ist, bei manchen Elementen die Endtags wegzulassen. Eine Liste können Sie in HTML 4.01 beispielsweise so schreiben:

```
<ul>
  <li>Startseite
  <li>Aktuelles
  <li>Informationen
  <li>Impressum
</ul>
```

Sie sehen, bei den einzelnen `li`-Elementen fehlen die schließenden `</li>`. Ein `li`-Element geht dann so weit, bis das nächste kommt. Eine weitere Besonderheit ist, dass Sie bei Attributwerten die Anführungszeichen weglassen dürfen. Sie können also anstelle von `height="90"` auch `height=90` schreiben.

3.6.2 XHTML – weniger Freiheit, mehr Klarheit

XHTML ist eine Neuformulierung von HTML auf der Basis von XML.

TIPP

XML steht für eXtensible Markup Language, das heißt erweiterbare Auszeichnungssprache, und ist eine Metasprache, die beschreibt, wie man Markupsprachen erstellt. Deswegen steht auch das X in XHTML für eXtensible. Wir werden in *Kapitel 12* noch einmal ausführlicher zu XML kommen.

XHTML 1.0, das im Web eingesetzt wird, hat an sich denselben Sprachumfang wie HTML 4.01, das heißt es gibt im Wesentlichen dieselben Elemente und Attribute, die es auch in HTML 4.01 gibt.

Aber es gibt syntaktische Unterschiede. Dort, wo HTML 4.01 lascher ist und Ausnahmen von der Regel »Zu jedem Starttag gibt es ein Endtag« zulässt, gibt es in XHTML keine solchen Ausnahmen.

Hier sehen Sie das Beispieldokument einmal in der XHTML-Variante:

Listing 3.9: **Das Beispieldokument als XHTML (*bilder_xhtml.html*)**

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/
    TR/xhtml1/DTD/xhtml1-strict.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03  <head>
04    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
05 <title>Bilder</title>
06 </head>
07 <body>
08 <h1>Bilder einbinden</h1>
09 <img src="landschaft.jpg" width="90" height="120" alt="Landschaft" />
10 <img src="blumen.jpg" width="90" height="120" alt="Blumen" />
11 </body>
12 </html>
```

Erst einmal fällt die andere Dokumenttypangabe auf:

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

HINWEIS

Auch XHTML 1.0 gibt es in drei verschiedenen Varianten, wobei heute nur noch die Varianten Strict und Transitional relevant sind.

Der Unterschied zwischen Strict und Transitional besteht wieder darin, dass eigentlich als veraltet eingestufte Attribute bei Transitional noch erlaubt sind, in Strict nicht.

Vor der Dokumenttypangabe könnten Sie auch noch die XML-Deklaration schreiben, die so aussieht:

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

Eine weitere Besonderheit gibt es beim Starttag von HTML. Hier ist eine sogenannte Namensraumangabe erforderlich:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Ansonsten gibt es im Beispiel keine großen Besonderheiten. Die leeren Elemente wie das `img`-Element werden mit einem Slash geschlossen:

```
<img src="landschaft.jpg" width="90" height="120" alt="Landschaft" />
```

Das ist obligatorisch. Vieles, was Sie jetzt in HTML dürften, dürfen Sie in XHTML nicht.

» Sie dürfen also nicht die Anführungszeichen um die Attribute weglassen.

» Sie dürfen bei Listen auch nicht die Endtags der `li`-Elemente weglassen.

» Sie müssen alle Elemente und Attribute kleinschreiben.

Für XHTML ist eigentlich ein spezieller MIME-Typ vorgesehen, nämlich `application/xhtml+xml`.

Problematisch ist hieran einerseits, dass ältere Internet Explorer diesen MIME-Typ nicht unterstützen, und andererseits, dass dann das Dokument auch wirklich vollständig den XHTML-Regeln entsprechen muss. Browser verwenden bei diesem MIME-Typ den internen XML-Parser, der beispielsweise bei syntaktischen Fehlern die Verarbeitung einfach abbricht und dann nur eine Fehlermeldung anstelle der Seite ausgibt.

So werden derzeit die meisten XHTML-Dokumente mit dem auch bei HTML verwendeten Standard-MIME-Typ `text/html` ausgeliefert.

Das ist eigentlich ein komischer Mischmasch. Wenn man XHTML schreibt, hält man sich an die strengere Syntax. Und die Browser interpretieren das heimlich dann doch wieder als HTML.

3.6.3 HTML5

Vor ein paar Jahren wäre nicht mehr dazu zu sagen gewesen. Aber jetzt steht neu auf der Bühne: HTML5.

Das W3C hatte ursprünglich eine eindeutige Linie: HTML sollte nicht weitergeführt werden. Version 4.01 sollte die letzte bleiben – so hatte das W3C es beschlossen und sich auf die Weiterentwicklung von XHTML in Form von XHTML 2 konzentriert.

Einer Gruppe von Firmen und Browserherstellern gingen die Entwürfe des W3C in die falsche Richtung: zu weit entfernt von der aktuellen Internet- und Browserrealität. So gründeten im Jahr 2004 Apple, Mozilla und Opera eine neue Arbeitsgruppe WHATWG (Web Hypertext Application

Technology Working Group). Im Gegensatz zum W3C setzten sie weiter auf HTML als Basis, ihr Arbeitsentwurf zu »Web Applications 1.0« wurde bald unter dem Namen HTML5 bekannt.

Im Jahre 2006 vermehrte sich die Kritik am W3C und an dem langsamen Vorankommen von Neuentwicklungen. Im Herbst 2006 kündigte Tim Berners Lee die Gründung einer neuen HTML-Arbeitsgruppe an, die im Frühjahr 2007 zusammentrat. So werden inzwischen die Entwürfe zu HTML5 sowohl vom W3C als auch von der WHATWG betreut. Eine Weile lang gab es parallel dazu noch eine Arbeitsgruppe zu XHTML2. Deren Aus kam im Jahr 2009, als das W3C beschloss, diese Arbeitsgruppe nicht mehr weiterzuführen, und seitdem ist es eindeutig: **Die Zukunft heißt HTML5**.

HTML5 ist mehr als HTML. Es ersetzt sowohl HTML als auch XHTML und die DOM-Spezifikation, die die Programmierung dieser Elemente über JavaScript beschreibt. Dass alles zusammengefasst ist, trägt dem Umstand Rechnung, dass es bei Webseiten heute immer häufiger nicht mehr um einfache Webseiten, sondern um Anwendungen/Applikationen geht. Uns interessiert natürlich hier der Teil von HTML5, bei dem es um die Auszeichnungssprache geht.

HTML5 führt die Zweiteilung von HTML und XHTML weiter. Das heißt, Sie können HTML5 als HTML schreiben oder auch als XHTML. Wenn Sie es als XHTML schreiben, müssen Sie aber den hierfür vorgesehenen MIME-Typ `application/xhtml+xml` nehmen. Da dieser im Internet Explorer nicht funktioniert, sehen wir hier von dieser Variante ab.

Jetzt zu HTML5 in der HTML-Variante. Das waren die Beispiele, die Sie hier gesehen haben, schön erkennbar an der einfachen Dokumenttypangabe, die man auswendig schreiben kann:

```
<!DOCTYPE html>
```

Und dann gibt es noch eine Besonderheit. Wenn Sie sich die strengere XHTML-Syntax angewöhnt haben, also beispielsweise leere Elemente mit einem Slash zu schließen, so dürfen Sie das weiterhin. Und damit sind wir bei den Dokumenten, wie Sie sie im bisherigen Kapitel kennengelernt haben.

Ob Sie HTML 4.01 oder XHTML 1.0 benutzen, ist Geschmackssache. Sie nehmen am besten das, was Sie besser kennen oder was Ihre Tools erzeugen, mit denen Sie sonst schon arbeiten.

Ich habe mich für dieses Buch für HTML5 entschieden, für die Arbeit mit PHP ist es aber nicht relevant, welche Variante von HTML Sie nutzen.

3.7 Formatierung mit CSS

Zur Formatierung von Webseiten dient CSS. CSS steht für Cascading Stylesheets, also so viel wie kaskadierende Formatvorlagen. Es ist die Formatierungssprache für Webseiten und wird ebenfalls vom W3C betreut.

CSS-Regeln folgen immer einem klaren Schema. Zuerst steht der sogenannte Selektor. Dieser wählt aus, für welche HTML-Elemente die angegebene Formatierung gelten soll. Dahinter stehen in geschweiften Klammern die CSS-Anweisungen. Eine CSS-Anweisung besteht aus dem Namen der Eigenschaft, einem Doppelpunkt und einem vorgegebenen Wert. Abgeschlossen wird das mit einem Strichpunkt.

Wenn Sie als Selektor den Namen eines Elements schreiben – also beispielsweise `h1` oder `p` –, so gelten die angegebenen Formatierungen für alle entsprechenden Elemente. Die folgende Regel färbt alle `h1`-Überschriften rot ein und gibt ihnen eine gelbe Hintergrundfarbe.

```
h1 {
  background-color: yellow; color: red;
}
```

CSS-Regeln können Sie an verschiedenen Stellen notieren. Beim Entwurf ist es praktisch, die Angaben im Kopf des Dokuments zu notieren. Dafür schreiben Sie innerhalb des `head`-Bereichs `<style type="text/css"> </style>`.

Listing 3.10: **Ein erstes CSS-Beispiel mit den CSS-Angaben im Dokumentkopf (*css_anfang.html*)**

```
01 <!DOCTYPE html>
02
03 <html>
04 <head>
05 <meta charset="UTF-8" />
06 <title>Erste Formatierungen</title>
07 <style type="text/css">
08 h1 { background-color: yellow; color: red; }
09 p { background-color: gray; color: white; }
10 </style>
11 </head>
12 <body>
13 <h1>Eine Überschrift</h2>
14 <p>Und ein Absatz - dieses Mal bunt!</p>
15 <p>Noch ein Absatz</p>
16 </body>
17 </html>
```

Abbildung 3.9: **Überschrift und Absatz in verschiedenen Farben**

3.7.1 Farbangaben

Oben haben Sie gesehen, dass für die Farbangabe ein englischer Farbname benutzt wurde. Offiziell erlaubt sind folgende sechzehn: `black` (schwarz), `green` (grün), `navy` (dunkelblau), `gray`

(grau), `lime` (hellgrün), `blue` (blau), `maroon` (dunkelrot), `olive` (olivgrün), `purple` (violett), `red` (rot), `yellow` (gelb), `fuchsia` (magenta), `silver` (hellgrau), `aqua` (cyan), `teal` (blaugrün), `white` (weiß).

Das ist gut zum Testen, aber für den richtigen Einsatz braucht man ein ausgefeilteres System. Im Web werden RGB-Farben (Rot-Grün-Blau) eingesetzt, die üblicherweise über zweistellige Hexadezimalzahlen angegeben werden. Hexadezimale Ziffern reichen von 0 über 9 und A bis F. Der größtmögliche Wert ist damit FF, der kleinstmögliche 00.

Die Farbangaben bestehen aus sechs Stellen und werden direkt nach einem Gatterzeichen (#) notiert. Die beiden ersten Stellen geben den Rotwert, die beiden nächsten den Grün- und die beiden letzten den Blauanteil an. Damit lässt sich die Farbe Weiß als `#FFFFFF` schreiben, `#FF0000` beschreibt einen Rotton etc.

```
h1 { color: #FF0000; }
```

3.7.2 Mehr Freiheit durch Klassen

Bisher haben Sie gesehen, wie man Formatierungen für alle Elemente einer bestimmten Art vornimmt, beispielsweise für alle Absätze. Nicht immer möchte man aber alle gleichartigen Elemente gleich formatieren, sondern auch Ausnahmen definieren. Das lässt sich über Klassen realisieren.

Sehen wir uns das am Beispiel einer Zebratabelle an. Bei dieser ist jede zweite Zeile anders eingefärbt. Um das zu realisieren, vergibt man an einzelne Elemente, nämlich die, die anders formatiert werden sollen, **eine Klasse**. Das heißt: Man ergänzt, wo gewünscht, im HTML-Teil das Attribut `class` mit einem frei wählbaren Namen:

```
<tr class="gerade">
```

Diese kann man dann bei der CSS-Definition ansprechen, indem man als Selektor den gerade vergebenen Namen mit einem Punkt davor notiert:

```
.gerade { background-color: #FFDFBF; }
```

Das folgende Beispiel zeigt die Realisierung einer Zebratabelle:

Listing 3.11: **Bei dieser Tabelle erhält jede zweite Zeile eine Klasse und darüber eine andere Hintergrundfarbe (*zebratabelle.html*).**

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8" />
<title>Zebratabelle</title>
<style type="text/css">
table { background-color: #FFC080; }
.gerade { background-color: #FFDFBF; }
</style>
</head>
<body>
  <table border="1">
    <tr>
      <th>Abfahrt Hbf</th><th>Ankunft Katzenreuth</th>
    </tr>
    <tr class="gerade">
```

```
18        <td>20:07</td><td>20:27</td>
19      </tr>
20      <tr>
21        <td>20:27</td><td>20:47</td>
22      </tr>
23      <tr class="gerade">
24        <td>20:47</td><td>21:07</td>
25      </tr>
26      <tr>
27        <td>21:07</td><td>21:27</td>
28      </tr>
29    </table>
30 </body>
31 </html>
```

Im Beispiel werden zwei CSS-Regeln definiert: Zum einen erhält die ganze Tabelle eine Hintergrundfarbe (Zeile 8) und außerdem wird für die Klasse `.gerade` eine andere Hintergrundfarbe bestimmt (Zeile 9). Die Klasse wird dann bei ausgewählten `tr`-Elementen eingesetzt (Zeile 17 und Zeile 23).

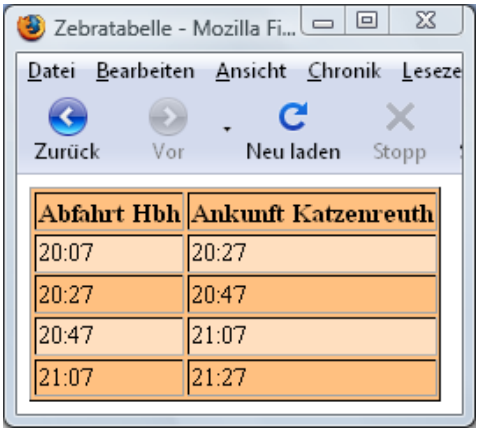

Abbildung 3.10: **Zebratabelle – jede zweite Zeile ist in einer anderen Farbe eingefärbt.**

TIPP

Es gibt eine Reihe von sogenannten Selektoren, um Elemente in CSS für die Formatierung auszuwählen. Ein weiterer ist der id-Selektor. Wenn Sie ein Element mit einer id kennzeichnen, etwa `<div id="beispiel">`, so können Sie dieses Element über den Selektor `#beispiel` (am Anfang steht ein #) für die Formatierung auswählen.

3.7.3 Weitere häufig benötigte Formatierungen

Bisher wurden in den Beispielen für CSS-Formatierungen nur Farben eingesetzt. Aber es gibt natürlich mehr. Hier eine Auflistung häufig benötigter Formatierungen.

Schriftart: Die Schriftart geben Sie über `font-family` an. Als Eigenschaft notieren Sie eine Liste von Schriften. Ist die erste auf dem Computer des Surfers nicht vorhanden, nimmt der Browser die nächste usw. usf.

```
p { font-family: Verdana, sans-serif; }
```

Schriftgröße: Zur Angabe der Schriftgröße benutzen Sie `font-size`. Notieren Sie dabei die Einheit direkt an den Wert. Einsetzen können Sie `px` für Pixel oder `em`. Der genaue Wert eines em orientiert sich jeweils an der gewählten Schriftgröße und entspricht im Normalfall 16px.

```
h2 { font-size: 1.2em; }
```

Kursiv: Um etwas kursiv zu machen, notieren Sie `font-style: italic`.

Fett: Durch die Anweisung `font-weight: bold` wird etwas fett.

Ausrichtung: Über `text-align: center` zentrieren Sie Inhalte, `text-align: right` würde sie hingegen rechts anordnen.

Textausschmückungen: Mit `text-decoration` können Sie eine Unterstreichung (`text-decoration: underline`) bestimmen oder diese auch per `text-decoration: none` entfernen.

Ausmaße von Elementen: Mit `width` lässt sich die Breite von Elementen bestimmen, mögliche Einheiten sind `px`, `em` oder `%`: `width: 300px` macht ein Element 300px breit, `height` macht dasselbe für die Höhe. Mit `padding` schaffen Sie Abstand zwischen Elementen und ihrem Rand. `border` können Sie für einen Rahmen einsetzen.

Das folgende Listing zeigt die Formatierungen im Einsatz.

Listing 3.12: **CSS-Formatierungen am Beispiel (*formatierungen.html*)**

```
01 <!DOCTYPE html>
02
03 <html>
04 <head>
05   <meta charset="UTF-8" />
06   <title>Formatierungen</title>
07   <style type="text/css">
08   body { font-family: Verdana, sans-serif; }
09   .stil1 { font-style: italic; }
10   .stil2 { color: red; background-color: yellow;font-size:1.5em;}
11   .stil3 { font-weight: bold; text-align: center; }
12   .stil4 { border: 2px dotted blue; padding: 20px; }
13   .stil5 { width: 400px; height: 80px; background-color: orange;}
14   p { background-color: #DDDDDD; }
15   </style>
16 </head>
17 <body>
18 <p class="stil1">Hier steht kursiver Text (font-style: italic)</p>
19 <p class="stil2">Groß (font-size: 1.5em) und bunt (color: red; background-
   color: yellow)</p>
20 <p class="stil3">Fett (font-weight: bold) und zentriert (text-align:
   center).</p>
21 <p class="stil4">Ein Absatz mit Rahmen und Innenabstand: border: 2px dotted
   blue; und padding: 200px</p>
22 <p class="stil5">Ein Absatz mit festgelegter Breite und Höhe (width: 400px;
   height: 80px; background-color: orange)
23 </body>
24 </html>
```

Im Beispiel wird für `body` die Schriftart Verdana oder eine andere serifenlose Schrift festgelegt (Zeile 8). Da `body` das umfassende Element ist, in dem alle anderen Elemente stehen, erben diese die Schriftart. Ab Zeile 10 beginnt die Definition von fünf Klassen, die immer unterschiedliche

Formatierungen haben. Diese Klassen werden weiter unten innerhalb von `body` dann den einzelnen `p`-Elementen zugewiesen (Zeile 19–23). Außerdem wird allgemein für `p` eine graue Hintergrundfarbe definiert (Zeile 14).

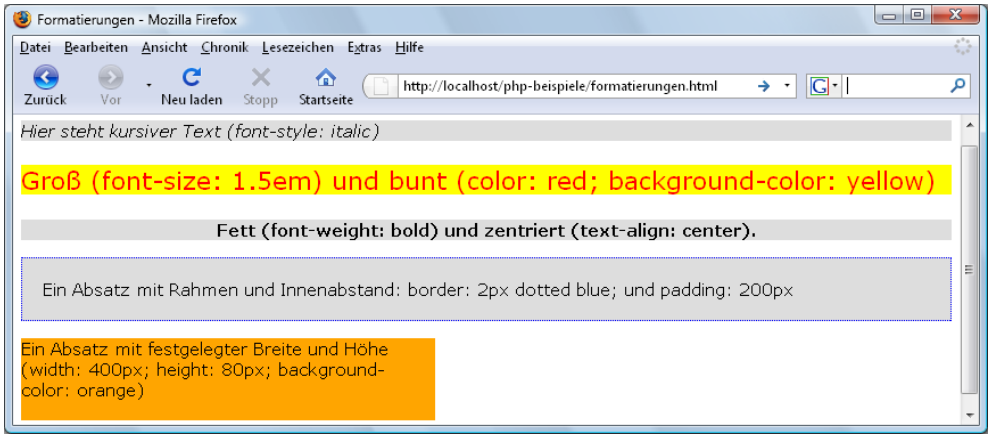

Abbildung 3.11: **Das Formatierungsbeispiel**

Dieser kurze Überblick über HTML/CSS hat aus Platzgründen natürlich nicht alle möglichen Elemente und Attribute behandelt. Suchen Sie weitere Informationen, so ist die Online-Dokumentation Selfhtml eine äußerst lohnenswerte Quelle. Sie finden sie unter `http://de.selfhtml.org/`. Sie können sich diese Dokumentation auch zum Offline-Lesen herunterladen. Neuer ist Selfhtml in Wiki-Form unter `http://wiki.selfhtml.org/wiki/Startseite`.