2 Webseiten-Gestaltung mit HTML und CSS

HTML Hypertext Markup Language und CSS Cascading Style Sheets bilden die fundamentalen Technologien zur Entwicklung von Webseiten. HTML ist für die Struktur der Seite zuständig, CSS für das Layout bzw. das Erscheinungsbild. Die Erstellung von Webseiten fordert eine strikte Trennung von Inhalt und Gestaltung sowie eine klare Strukturierung der Inhalte und durchgehend einheitliche Verknüpfung der Seiten. Die weitere Aufgabe besteht in der Definition von Formatvorlagen mit denen die Elemente auf der Seite positioniert und gestaltet werden.

HTML und CSS gehören zusammen. Ohne HTML kein CSS und umgekehrt. Aus diesem Grund wird die Auszeichnungssprache HTML in einem Kapitel mit der deklarativen Stilsprache CSS zusammen behandelt.

Zunächst liegt das Augenmerk auf der Strukturierung der Inhalte. Anschließend erfolgt die Betrachtung der technischen Möglichkeiten zur Gestaltung von Internetseiten. Zum Abschluß werden Aspekte der Publikation einer Website mit multimedialen Inhalten aufgegriffen. Dem Anwender stehen am Ende dieses Kapitels drei fertige Webseiten-Templates mit Navigation und Stilvorlagen zur Verfügung.

Wenn Sie erstmalig Kontakt mit einer Programmiersprache haben, ist es zunächst empfehlenswert, die Grundlagen mit einer einfachen Entwicklungsumgebung zu erarbeiten. Erst wenn man die Zusammenhänge kennt, sollte man sich komplexerer Entwicklungswerkzeuge bedienen. Die Werkzeugempfehlung für dieses Kapitel ist Notepad++ und Firefox mit den Funktionen des Web-Entwickler Menüs. Zur Validierung der Dokumente benutzen wir die Fehlerkonsole von Mozilla Firefox und das Entwickler-Plug-in Firebug von Mozilla. Mittlerweile sind aber auch die anderen gängigen Web-Broser mit Entwickler-Tools ausgestattet, so dass Sie alternativ den Browser Ihrer Wahl einsetzen können.

Internetprovider bieten die Möglichkeit der Online-Generierung einer Webseite mit einem Homepage-Baukasten. Sie melden sich auf der Seite an, geben etwas Text ein, klicken ein paar Bilder, suchen Farben und Zeichensätze für die Schrift aus und schon ist die Homepage fertig. Die Werbung verspricht Seitenerstellung ohne HTML-Kenntnisse. Auch viele HTML-Editoren bieten Gleichartiges. Der Anwender stellt nur noch seine Daten zusammen und wählt ein Layout. Ebenfalls führen Blogsysteme unmittelbar zu einer professionell aussehenden Website.

Noch weiter geht es bei der Nutzung von *Content Management Systemen* (CMS). Man setzt sich an einen Internetarbeitsplatz, loggt sich ein und publiziert einen Artikel. Alle Webseiten, die zusammen die Web-Site bilden, haben immer das gleiche Layout. Doch auch diese hochkomplexen Systeme bieten die Möglichkeit der unmittelbaren Einbindung von HTML-Tags. Spätestens hier wird dann deutlich,

G. Pomaska, *Webseiten-Programmierung*, DOI 10.1007/978-3-8348-2485-1_2,

wie sinnvoll Basiswissen sein kann. Wer auf soliden Grundkenntnissen aufbaut, ist in der Lage, komplexe Systeme schnell zu erlernen und anzuwenden und kann notwendige Modifikationen im Quelltext selbst vornehmen.

HTML ist mit wenig Zeitaufwand leicht zu erlernen. Bevor wir mit dem ersten Hello-World-Dokument beginnen, noch ein paar Worte zu den Versionen. HTML 4.01 sollte die letzte Version von HTML sein und durch XHTML, der XML konformen Formulierung ersetzt werden. Entgegen den Vorstellungen des W3C wurden aus der Industrie heraus die Standards Web Applications 1.0 und Web Forms 2.0, mit der Zielsetzung, das Web als Anwendungsplattform zu nutzen, entworfen. Daraus entwickelte sich der Begriff HTML5. Es sind noch nicht alle Entwürfe umgesetzt und alle Standards in die Browser integriert. HTML5 wächst kontinuierlich, der Begriff vom *Living Standard* wurde propagiert. Mit HTML5 werden Webseiten semantischer, interaktiver und von Plug-ins befreit.

2.1 HTML Hypertext Markup Language

2.1.1 Dokumentenstruktur und Metatags

Ein HTML-Dokument gliedert sich in die Bereiche

- Dokumententyp-Deklaration
- HTML-Root-Tag
- Abschnitt Head
- Abschnitt Body

Mit der Dokumententyp-Deklaration wird angegeben, nach welchem Standard die Webseite erstellt wurde. Der Root-Tag umschließt das Dokument.

Im Head-Abschnitt befinden sich der Seitentitel, Metainformationen, Stilvorlagen, Skripte und Referenzen auf externe Dateien. Die Informationen im Abschnitt Head werden nicht vom Browser dargestellt.

Der Körper eines HTML-Dokuments enthält alle Elemente des Inhalts wie Text, Hyperlinks, Bilder, Tabellen, Listen u.a., die vom Browser entsprechend der Stilvorgaben gerendert werden.

Ein HTML-Element besteht aus Auszeichnungen, den sog. Tags, Attributen (Eigenschaften) und dem Inhalt. Alle HTML-Auszeichnungen befinden sich zwischen spitzen Klammern. Jeder Tag hat einen Start-Tag und einen Ende-Tag. Der Ende-Tag entspricht dem Anfangstag mit einem vorgestellten / bzw. in der Kurzform als Ende des Starttags />. Zwischen Groß- und Kleinschreibung wird bei HTML nicht unterschieden. Wir wählen hier aber immer die Kleinschreibung der Tags.

Alle HTML-Auszeichnungen müssen sich zwischen `<html>` und `</html>` befinden. Innerhalb von `<head></head>` ist ein Title-Tag vorgeschrieben, dessen Inhalt im Titelbalken des Anwendungsfensters erscheint.

```
<title>Text der Titelzeile</title>
```

Die Metatags liefern u.a. Informationen über Inhalt und Autor der Webseite und bieten Hinweise für Suchmaschinen an.

```
<meta charset="utf-8">
<meta name="description" content="Struktur einer Webseite" />
<meta name="keywords"  content="HTML, Metatags" />
<meta name="author"   content="gp" />
```

Zwischen `<body></body>` befinden sich die Anweisungen für den Browser. Ein Tag kann Attribute enthalten, den Attributen wird über den Zuweisungsoperator (Gleichheitszeichen) ein Wert zugewiesen. Mehrere Attribute sind durch Leerzeichen zu trennen. Die Attributwerte befinden sich immer in doppelten Hochkommata, seit HTML5 ist diese Schreibweise aber nicht mehr zwingend vorgeschrieben. Die Angabe der Landessprache ist ein Attribut im HTML-Tag und gehört zu den Universalattributen, ebenso wie eine Identität (`id`), die als eindeutiger Bezeichner für ein Element benutzt wird.

```
<html lang="de">
```

Ein Kommentar in HTML befindet sich zwischen den Auszeichnungen

```
<!--
 Kommentar für den Benutzer des Quelltextes, wird vom Browser ignoriert
 -->
```

und kann sich über mehrere Zeilen erstrecken.

Innerhalb des Body-Tags können die Elemente gruppiert werden. Bisher hatte man lediglich mit den Tags `<div></div>` und `<span></span>` die Möglichkeit sog. Container einzurichten. Mit dem Class-Attribut und dem ID-Attribut können diese Container eindeutig referenziert werden. Mit dem Entwurf von HTML5 wurden weitere Elemente zur Strukturierung eingeführt, die ein semantisches Markup unterstützen und Class-Attribute verzichtbar machen. Hierzu gehören die logischen Bereiche einer Seite wie Kopfbereich, Navigation, Fußbereich und Abschnitte für Artikel. Im weiteren Verlauf dieses Kapitels werden wir die Strukturelemente kennen lernen und durch die Stilvorlagen positionieren und gestalten.

Hier kommt nun die erste HTML-Seite mit den Einteilungen *Kopfzeile*, *Inhalt*, und *Fußzeile*. Wo diese Bereiche letztlich auf der Seite zu finden sein werden, steuert später die Stilvorlage. Ohne Positionierungsanweisungen rendert der Browser von oben links nach unten rechts im zur Verfügung stehenden Platz des Windows. Das Attribut `charset` im Metatag gibt den benutzten Zeichensatz an. Die Leerzeichen im Dokument dienen nur der Übersicht im Quellcode und werden vom Browser als White-Spaces behandelt, überflüssige Leerzeichen werden entfernt. Wenn Sie mit dem Notepad++ Editor arbeiten, stellen Sie unter Encoding > Encode UTF-8 ohne BOM (byte order mark) ein.

```
<!DOCTYPE html>
<!-- Dokument htmlcss/index0.html -->
<html lang="de">
  <head>
    <meta charset="utf-8">
    <meta name="description" content="Struktur einer Webseite" />
    <meta name="keywords"  content="HTML, Metatags" />
```

```
    <meta name="author"   content="gp" />
    <title>HTML Living Standard</title>
  </head>
  <body>
    <header> Der Kopfbereich einer Seite </header>
    <section> Der Inhaltsbereich     </section>
    <footer> Die Fusszeile       </footer>
  </body>
</html>
```

Der Kopfbereich einer Seite wird mit dem Header-Tag eingeleitet. Der Kopfbereich kann mehrfach auf einer Seite in unterschiedlichen Abschnitten vorkommen, gleiches gilt für den Fußbereich. Unterschieden werden die Bereiche durch Class-Attribute oder Selektoren, wie später im Kapitel CSS erläutert wird. Mit dem Section-Tag wird ein logischer Abschnitt einer Seite eingeleitet. Das kann der Inhaltsbereich sein, der die zu veröffentlichen Themen aufnimmt oder auch ein Navigationsbereich mit eigenem Header und Footer. In die Strukturelemente ist auch der Iframe-Tag einzubeziehen. Der Inline-Frame reserviert einen Bereich im HTML-Dokument zur Einfügung eines kompletten externen Dokuments.

```
<iframe src="http://www.seiten-programmierung.de"
    width="400" height="300" name="meinFrame">
</iframe>
```

Die Seiteneinteilung mit den neuen HTML-Tags wie Section- oder Artikel ist nicht zwingend vorgeschrieben. Sie dürfen auch weiterhin den Div-Tag benutzen und können die Container nach eigenem Bedarf einrichten.

2.1.2 Textstrukturierung

Text ist gegliedert durch Überschriften und Abschnitte. Innerhalb des Textes gibt es Hervorhebungen, Streichungen und anderweitig zu gestaltende Bereiche wie Listen, Adressen, Programm-Quelltext oder Einrückungen.

Überschriften werden als Headings `<h1></h1>` bezeichnet. Für die Wertigkeit der Überschrift benutzt man die Zahlen von 1 bis 6. Überschriften mehrerer Ebenen kann man mit Tag `<hgroup>` gruppieren. Der eigentliche Text wird in Abschnitte (Paragraphen) `<p></p>` unterteilt

Zur grafischen Unterteilung kann mit horizontalen Linien gearbeitet werden. Der Tag wird mit `<hr />` ausgezeichnet und besitzt keinen Ende-Tag. Ein Zeilenumbruch wird mit dem Tag `<br>` erzwungen.

Wir können jetzt die erste eigene HTML-Datei erstellen. Dazu besorgen wir etwas Blindtext von *www.all2e.com*, der im Quelltext hier aber nur reduziert wiedergegeben ist. Die Datei wird gespeichert unter *index1.html* und durch Doppelklick im Dateiexplorer oder im Web-Broser aufgerufen. Wir benutzen das obige Template zur Dokumentenstrukturierung und editieren den Bereich Section des HTML-Dokuments wie folgt:

```
<section>
 <article>
  <header>
```

```
    <h1>Textstrukturierung in HTML</h1><hr />
    <h2>Blindtext</h2>
  </header>
  <p>Weit hinten, hinter den Wortbergen ...
     <del>Vokalien und Konsonantien</del> ...
     <blockquote>Eines Tages aber beschloß eine kleine
                 <del>gehen in die weite</del> ...
     </blockquote>
     Es packte seine sieben Versalien
     <pre>  Als es die ersten
                    Hügel des Kursivgebirges
               erklommen hatte,
     </pre>
     Fett, rund, ein echter Buddha...
     <ul>
        <li>Wehmütig lief ihm eine rethorische Frage</li>
        <li>über die Wange,</li>
        <li> dann setzte es seinen Weg fort.</li>
     </ul
   </p>
</article>
<footer">
 <hr />Blindtext von www.all2e.com</footer>
</section>
```

Beachten Sie bitte, dass der Zeilenumbruch durch den Browser realisiert wird. Je nach Bildschirmauflösung und Fenstergröße wird die Datei mit automatischem Zeilenumbruch dargestellt. Der Web-Autor kann ja nicht wissen, mit welcher Fenstergröße und mit welcher Auflösung der Nutzer die Datei betrachten wird. Bei Bedarf werden vom Browser die Scrollbars eingeblendet. Im Browserfenster erscheinen die Überschriften `<h1>`, `<h2>` der Textabschnitt `<p>` mit den Streichungen `<del>`, der Einrückung `<blockquote>` und einer Vorformatierung `<pre>`. Im vorformatierten Bereich wechselt der Zeichensatz und die Leerzeichen werden nicht mehr entfernt. Der Zeilenumbruch entspricht dem des Originaltextes.

Es sei noch einmal darauf hingewiesen, dass in diesem Lehrbuch nicht alle Tags mit ihren Eigenschaften und Möglichkeiten detailliert erörtert werden können. Wie in allen weiteren Abschnitten auch wird als Nachschlagwerk für Tags und Attribute auf die HTML- und CSS-Referenz bei *www.w3.org* verwiesen. Auch *de.selfhtml.org* ist eine wertvolle Quelle.

HTML stellt für Aufzählungslisten `<ul>`, nummerierte Listen `<ol>` und Definitionslisten <dl> die benötigten Tags bereit. Bei Aufzählungslisten und nummerierten Listen werden die Listenelemente in den Tag `<li>` eingebunden. Definitionslisten unterteilen in den zu definierenden Ausdruck `<dt>` und die Definition selbst `<dd>`. Der Aufzählungsliste im Beispiel der Abbildung 2.1 folgt die Grußformel als Hervorhebung ausgezeichnet mit `<em>` und am Ende wird der Textabschnitt noch geschlossen. Im Bereich `<footer>` befindet sich der Quellverweis mit einem Copyright-Zeichen.

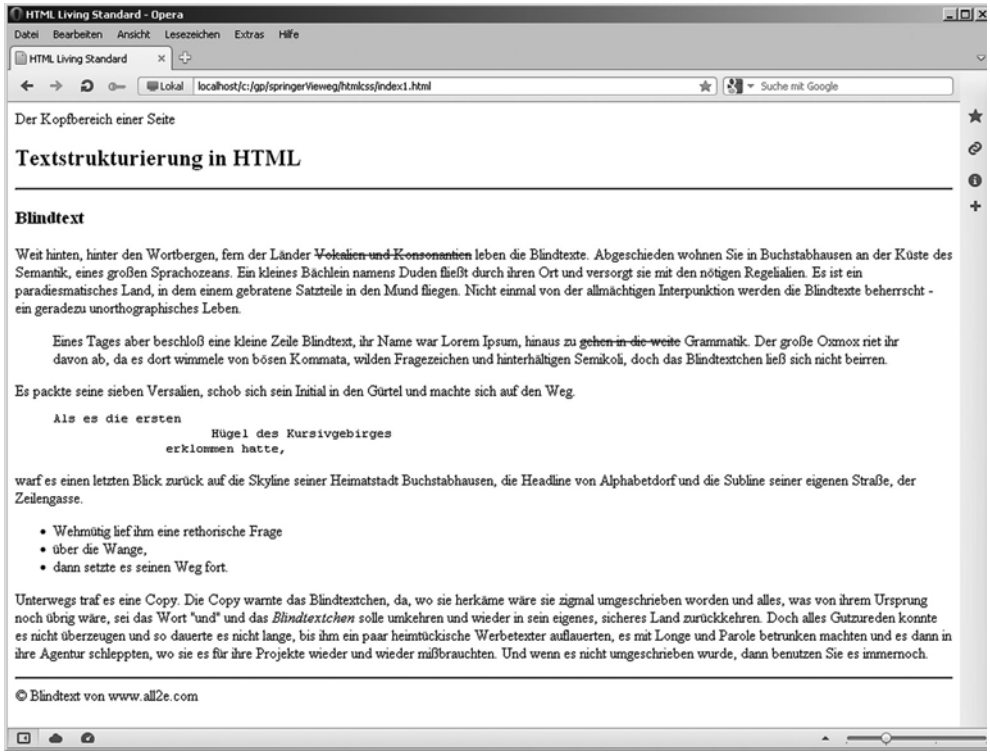

Abbildung 2.1: Seiten- und Textstrukturierung

Die erste HTML-Seite haben Sie jetzt bereits erstellt. Die Seite wurde eingeteilt in einen Seitenheader, eine Inhalts-Section mit einem Artikel. Der Artikel enthält einen Header und Textabschnitt mit verschiedenen Auszeichnungen. Im Anschluss an den Inhaltsbereich folgt noch eine Seitenfußzeile. Auf der Webseite zum Buch gelangen Sie zum Download-Archiv *htmlCSS.zip*. Dort wird die leere HTML-Datei mit *index0.html* bezeichnet, die Textseite ist unter *index1.html* abgelegt.

2.1.3 Tabellen

Tabellen sind ein Beschreibungselement innerhalb von HTML, mit denen Daten in Zellen positioniert werden. Vor der Einführung von CSS wurden Tabellen als universelles Gestaltungsraster benutzt. Heute finden Tabellen nur in der ursprünglichen Bedeutung Anwendung.

Eine Tabelle wird durch den Table-Tag eingeleitet. Bestandteile der Tabelle sind der Tabellenkopf `<thead>` mit den Spaltenüberschriften `<th>`, die Tabellenzeilen `<tr>` und Tabellenzellen `<td>` sowie eine Zusammenfassung im Fußbereich `<tfoot>`. Die Kombination der Tags `<thead>` und `<tfoot>` wird im Zusammenhang mit dem Tag `<tbody>` benutzt.

Tabellen werden immer, von links oben beginnend, zeilenweise mit der Definition von Tabellenzellen aufgefüllt. Unterschieden wird zwischen regelmäßigen und

unregelmäßigen Tabellen. In unregelmäßigen Tabellen können sich Datenzellen über mehrere Spalten, Attribut `colspan`, oder über mehrere Zeilen, Attribut `rowspan`, ausdehnen. Die Attributwerte definieren die Anzahl der belegten Spalten bzw. Zeilen.

Die wichtige Angaben zur Gestaltung einer Tabelle sind der Abstand zwischen den Zellen, Attribut `cellspacing`, und die Polsterung, das ist der Abstand der Zelleninhalte zum Rand, Attribut `cellpadding`. Seit HTML5 sollte man diese Attribute aber nicht mehr verwenden, sondern gegen eine Angabe in der Stilvorlage austauschen. Im Zusammenspiel mit Hintergrundfarbe und Hintergrund der Tabellenzellen lassen sich gut lesbare Tabellen formatieren. HTML5 unterstützt nur noch das Attribut `border` für die Gestaltung der Zellenränder. Alle weiteren Attribute werden der Stilvorlage übertragen.

Nehmen wir die ersten drei Vereine der Bundesligatabelle vom 21. Spieltag der Saison 2011/2012 als Beispiel für eine Tabelle. An dieser Stelle zunächst ohne Gestaltungsattribute, die kommen zur Ergänzung in Abschnitt 2.2 bei der vollständigen Tabelle zur Anwendung.

```
<!-- htmlcss/index2.html -->
<table border ="1">
<thead>
  <tr>
    <th colspan="9">1. Fussball Bundesliga Saison 2011/2012
                    21.Spieltag</th></tr>
  <tr>
    <th> </th><th>Verein</th><th>Sp</th><th>G</th><th>U</th>
    <th>N</th><th>Tore</th><th>+/-</th><th>P</th></tr>
</thead>
<tfoot>
  <tr>
    <td colspan="9">1-3 Champions League, 4 Champions League Quali.,5-6
                    Europa League,16 Relegation,17-18 Abstieg</td></tr>
</tfoot>
<tbody>
  <tr>
    <td>1</td><td>Borussia Dortmund</td><td>21</td><td>14</td>
    <td>4</td><td>3</td><td>46:14</td><td>32</td><td>46</td></tr>
    <tr><td>2</td><td>Bayern München</td><td>21</td><td>14</td><td>2</td>
    <td>5</td><td>49:14</td><td>35</td><td>44</td></tr>
  <tr>
    <td>3 </td><td>Bor. Mönchengladbach</td><td>21</td><td>13</td>
    <td>4</td><td>4</td><td>34:12</td><td>22</td><td>43</td></tr>
</tbody>
</table>
```

Die Tabelle wird eingeleitet mit dem Table-Tag, das Attribut für die Zellenränder ist 1 Pixel. Mit dem Tag `<thead>` wird der Tabellenkopf definiert. Die erste Zeile erstreckt sich über alle neun Spalten, Attribut `colspan="9"`, darin befindet sich die Gesamtüberschrift. In der folgenden Zeile kommen die Spaltenüberschriften. Der Tabellenkopf ist definiert, es folgt der Fußbereich mit `tfoot`, ebenfalls wieder in einer Tabellenzelle für 9 Spalten definiert. Analog zum Attribut `colspan` kann mit `rowspan` ein größerer Bereich Zeilen abgedeckt werden. Im Anschluss an

<thead> und <tfoot> wird der Tabellenkörper in <tbody> eingebettet. Hier folgen die drei gleichmäßigen Tabellenzeilen mit der Definition der Tabellenzellen.

1. Fussball Bundesliga Saison 2011/2012 21. Spieltag

	Verein	Sp	G	U	N	Tore	+/-	P
1	Borussia Dortmund	21	14	4	3	46:14	32	46
2	Bayern München	21	14	2	5	49:14	35	44
3	Bor. Mönchengladbach	21	13	4	4	34:12	22	43

1-3 Champions League, 4 Champions League Quali ,5-6 Europa League,16 Relegation,17-18 Abstieg

Abbildung 2.2: Tabellenspitze der Fussballbundesliga

2.1.4 Grafikeinbindung

Ein Farbmodell bildet die Farben der realen Welt in eine rechnerinterne Darstellung ab. Ein digitales Bild besteht aus einer rechteckigen Anordnung von Farbwerten des zugrunde liegenden Modells. Die horizontalen Felder bilden die Zeilen, die vertikalen Felder bilden die Spalten. Daraus ergibt sich eine Bildmatrix mit n * m Elementen. Die Anzahl der Bildpunkte bestimmt die geometrische Auflösung. Die Anzahl der darstellbaren Farbwerte in jedem Bildpunkt wird als Speichertiefe in Bit angegeben. Ein binäres Bild hat eine Speichertiefe von einem Bit. Das GIF-Format hat eine Speichertiefe von acht Bit und kann somit 256 unterschiedliche Farbwerte darstellen. In diesem Fall wird von indizierten Farben gesprochen, da die Farbwerte auf eine Farbtabelle verweisen können. Das GIF Graphics Interchange Format arbeitet mit verlustfreier Kompression und kann neben Transparenz auch Animationen wiedergeben. Mehrere Einzelbilder werden in einer Datei gespeichert und werden durch schnelle Wiederholung animiert abgespielt. Das GIF-Format eignet sich für Zeichnungen (Cartoons) und Schwarz-Weiß Fotografien.

Bilder mit Echtfarben (true color) werden mit 24 Bit Speichertiefe abgespeichert. Das JPEG-Format, entwickelt von der Joint Photographic Experts Group, speichert Bilder mit verlustbehafteter Komprimierung sequenziell oder progressiv. Ein progressiv gespeichertes Bild besteht aus Koeffizienten, die eine Annäherung an das Originalbild abbilden. Dem Betrachter wird bei großen Bildern der Effekt eines schnellen Vorschaubildes, ähnlich dem Interlacing (Zeilensprung) bei GIF-Bildern geboten.

PNG Portable Network Graphics ist ein Format für Grafiken mit verlustfreier Bildkompression. PNG unterstützt unterschiedliche Farbtiefen und Transparenz. Das PNG-Format wurde aufgrund von Lizenzforderungen für GIF entwickelt und ist eine W3C-Empfehlung, die von allen Browsern unterstützt wird.

Die Abbildung der realen Farbenwelt wird im Internet durch das RGB-alpha-Farbmodell realisiert. Farbwerte werden als Ganzzahlen oder Hexadezimalwerte

für die drei Grundfarben Rot, Grün und Blau angegeben. Alpha ist der Faktor für die Transparenz.

Grafiken können als Füllmuster, Hintergrundbild oder als Seiteninhalt in eine Webseite eingebunden werden. Füllt ein Bild nicht den vorgesehenen Raum aus, wird es wiederholt dargestellt, als Hintergrundbild kann es feststehend unabhängig vom Scrollen der Seite angezeigt werden. Anwendungen als Seitenhintergrund werden im Abschnitt CSS vorgestellt.

Die Einbindung in den Seiteninhalt erfolgt über den Image-Tag. Die Attribute des Image-Tags sind die Quelle (`src`), ein Text (`alt`), der bei Mausberührung gezeigt wird und die Höhe (`height`) und Breite (`width`) des Bildes. Zwei weitere Attribute sind zur Verwendung als Image-Map vorgesehen, vgl. Abschintt 2.1.6 Hyperlinks. Wenn Sie auf die Angabe von Breite und Höhe des Bildes verzichten, wird die Größe des Originalbildes angezeigt. Sinnvoll ist die Anpassung der Bildgröße an die spätere Darstellung. Dann ist die Angabe der Bildgröße verzichtbar und der Browser hat keine Umrechnung vorzunehmen. Die Referenzierung der Bildquelle erfolgt durch relative Pfadangabe ausgehend vom aktuellen Dokument in das es eingebunden ist. Meistens befinden sich die Grafiken in einem eigenen Verzeichnis, unterhalb der Root-Ebene der Website. Wir bauen eine kleine Beispielseite mit alternativer Grafikeinbindung.

```
<!-- htmlcss/index3.html -->
<p> … lorem ipsum …
<img src="img/bauarbeiter.gif" alt="Bauarbeiter" width="64"
   height="96" />
...dolor sit amet
</p>
```

Die alternative Einbindung eines Bildes ist seit HTML5 die Auszeichnung mit dem Figure-Tag und der Möglichkeit durch `<figcaption>` eine Bildunterschrift mit einzufügen.

```
 <figure>
  <img src="img/bauarbeiter.gif" alt="Bauarbeiter" width="64"
  height="96" />
  <figcaption>Der fleissige Bauarbeiter</figcaption>
</figure>
```

Ebenfalls angezeigt ist die Verwendung des Object-Tags oder Embed-Tags. Bis auf den Figure-Tag fügen sich alle Bilder in die Textzeile ein, wirken als sog. Inline-Element.

Abbildung 2.3: Grafiken im Textfluss

ICO ist ein Datenformat, das unter Windows zur Speicherung von Icons verwendet wird. Ein 16x16 oder 32x32 kleines Symbol bezeichnet man auch als Favicon, das in die Adresszeile des Browsers eingebunden werden kann. Das Format können Sie mit der IrfanView-Bildbearbeitung abspeichern. Laden Sie ein beliebiges Bildformat, ändern Sie die Größe und speichern Sie das Bild als ICO-Format. In die Adresszeile des Browsers wird es durch einen Link-Tag im Abschnitt `head` eingebaut:

```
<link rel = "icon" href="img/favicon.ico" type="image/x-icon" />
```

Sofern Sie es in einem anderen Format vorliegen haben geben Sie den entsprechenden Mime-Typ mit an.

```
<link rel = "icon" href="img/favicon.png" type="image/png" />
```

MIME ist die Abkürzung für Multipurpose Internet Mail Extension und klassifiziert die Daten innerhalb einer Nachricht. Dem Browser wird auf diese Weise mitgeteilt, welche Daten der Server übermittelt.Binden Sie doch standardmäßig in die Webseiten immer ihr eigenes Favicon ein, das gehört heute einfach dazu.

Sie sollten noch die Anmerkungen zu den zu den Dateireferenzen beachten. Da Sie nicht immer wissen unter welchem Betriebssystem ihre Website beheimatet ist, sollten Bezeichner von Verzeichnissen, Dateinamen und Erweiterungen grundsätzlich mit kleinen Buchstaben und ohne Sonderzeichen angegeben werden. Dabei ist immer eine relative Pfadangabe zu berücksichtigen. Der Pfad wird von der aktuellen Position des HTML-Dokuments aus verfolgt. Mit der Zeichenkombination ../ gelangen Sie in das Elternverzeichnis, eine Ebene höher. Nehmen Sie als Beispiel folgende Dateistruktur:

```
c:\Web-Server\de\start.html
c:\Web-Server\gb
c:\Web-Server\img\bild.jpg
      index.html
```

In der Datei *index.html* adressieren Sie *de/start.html*. In start.html wollen Sie nun ein Bild einbinden. Der Tag lautet:

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

Mit der relativen Adressierung sind Sie jederzeit zum Umzug mit dem kompletten Verzeichnis *Web-Server* gerüstet. Andernfalls müssten Sie die Pfadzuordnungen im Zielsystem individuell anpassen.

Neben einem Favicon in der Adresszeile gehören auch die Social Media Buttons schon zur Standardausstattung einer Webseite. Das Internet unterstützt Sie beim generieren der Code Snippets. Bei *www.button-einbauen.de* können Sie den Code für Facebook-Like, Google Plus und Twitter generieren. Eine große Auswahl an Buttons gibt es bei *www.socialbookmark.eu*. Den generierten Code bauen Sie in Ihre Webseite ein, vgl. Abbildung 2.4. Aber Vorsicht es gibt dort einige Dinge hinsichtlich des Datenschutzes, die möglicherweise nicht in Ihrem Interesse sind. Der Heise Verlag hat daher ein Plugin für 2 Klick-Buttons mit mehr Datenschutz entwickelt, das Sie über die URL *http://www.heise.de/ct/artikel/2-Klicks-fuer-mehr-Datenschutz-1333879.html* beziehen können.

Abbildung 2.4: Social Bokkmarks

2.1.5 Formulare

Interaktive Benutzereingaben auf Webseiten werden über Formulare vorgenommen. Neben den Bedienelementen wie Checkboxen, Radiobuttons, Eingabefeldern u.a. stellen Formulare auch die Kommunikationsschnittstelle zum Web-Server her.

Formulardaten können clientseitig mit JavaScript ausgewertet werden. Serverseitig werden die übermittelten Daten mittels dort laufender CGI–Programme ausgewertet oder als Umgebungsvariablen den Skripten bzw. Programmen verfügbar gemacht. Sofern der Benutzer einen Email-Client installiert hat, können die Daten auch unmittelbar an den Empfänger per Email weitergeleitet werden.

Ein Formular wird mit dem Tag <form> eingeleitet. Die Attribute des Tags legen die nach der Eingabebestätigung auszuführende Aktion fest. Innerhalb der Formulare gibt es die Tags <input> mit Type-Attribut für unterschiedliche Datentypen, <textarea> und <select>. Die Positionierung der Formularelemente kann geeignet mit Tabellen erfolgen.

Wir wollen ein kleines Kontaktformular aufbauen. Der Besucher einer Webseite erhält die Möglichkeit, eine Anfrage per E-Mail an den Betreiber der Webseite zu stellen. Zunächst betrachten wir das sehr einfache Grundgerüst:

```
<html>
<head>
<title>Formulareingaben</title></head>
<body>
 <form>
  <input name="absender" size="40" maxlength="60"
         value="eMail:">
  <textarea name="anfrage" rows="10" cols="50"
      wrap="virtual">Ihre Anfrage:</textarea>
  <input type="reset" value="Reset">
  <input type="submit" value="Senden">
 </form>
</body>
</html>
```

Innerhalb des Abschnitts <body> befindet sich das Form-Tag. Benutzt werden die Befehle <input> und <textarea>. Es wird ein Texteingabefeld mit dem Bezeichner *absender* für das Attribut `name` eingegeben. Die Größe des angezeigten Feldes beträgt 40 Zeichen, die maximale Anzahl der einzugebenden Zeichen ist auf 60 begrenzt. Das Eingabefeld ist mit dem Text *eMail* vorbesetzt. Durch die Auszeichnung <textarea> wird ein Texteingabebereich von 10 Zeilen und 50 Zeichen aufgebaut. Der automatische Textumbruch wird mit `wrap="`*virtual*`"` ausgeführt. Das Textfeld ist mit dem Kommentar *Ihre Anfrage:* vorbesetzt.

Es folgen noch zwei Buttons. Mit Buttons vom Typ *reset* werden alle Eingaben auf die Anfangswerte zurück gesetzt, bei Klick auf den Button vom Typ *submit* werden die Formulardaten mit der im Form-Tag angegeben Aktion an den Server übermittelt. Das Formular ist zwar schon benutzbar, aber die Behandlung der Daten ist noch nicht angegeben. Zur besseren Übersicht müssen die Elemente auch noch positioniert werden. Nach der im vorhergehenden Abschnitt entwickelten Tabellenstruktur wird in das Formular eine zweispaltige Tabelle mit 640 Pixel Breite eingefügt. Innerhalb der Tabellenzellen finden die Dialogtexte und die Formularelemente ihren zugewiesenen Platz. Der Browser stellt das Listing mit bereits aus-

gefüllten Eingabedaten wie in der Abbildung 2.4 dar. Die Positionierung der Formularelemente ohne Tabelle wird im Abschnitt 2.3.3 gezeigt.

```
<!—Dokument index4.html -->
<form>
  <table width="640" cellpadding ="3" border="1">
  <tr>
    <td>Absender eMail:</td>
    <td><input name="absender" size="40" maxlength="60">
    </td></tr>
  <tr>
    <td>Ihre Anfrage:</td>
    <td>
       <textarea name="anfrage" rows="10" cols="40" wrap="virtual">
       </textarea></td></tr>
  <tr>
    <td><input type=reset value="Reset"></td>
    <td><input type=submit value="Senden"></td>
  </tr></table></form>
```

Wie soll nun nach Absenden mit dem ausgefüllten Formular verfahren werden? Die Auszeichnung <form> benutzt hierfür das Attribut *action*. Der Wert von *action* kann eine URL sein, die eine Webseite mit Anweisungen zur Auswertung der übermittelten Daten angibt.

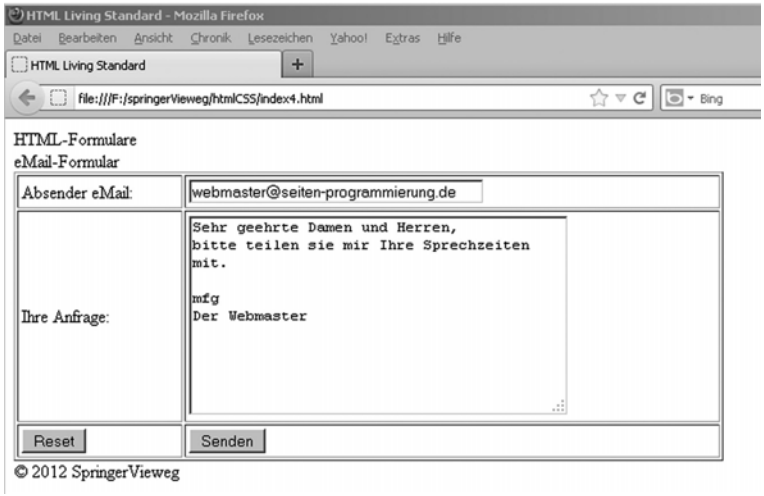

Abbildung 2.5: Einfaches Kontaktformular

Die Art der Datenübermittlung wird im Attribut `method` durch *post* oder *get* festgelegt. Die Get-Methode hängt die Parameter sichtbar an die gesendete URL an und begrenzt die Datenmenge. Vorzuziehen ist die Post-Methode, die Daten unabhängig von der URL übermittelt und größere Informationsinhalte überträgt.

Zwei weitere Parameter von <form> dienen den Auswerteprogrammen zur Identifikation der Formularelemente, die Werte für `name` und `id` erhalten vom Programmierer frei zu vergebende Bezeichner. An dieser Stelle wird nur auf die Existenz der Attribute hingewiesen, in späteren Anwendungen wird auch deren Auswertung behandelt.

In diesem Beispiel werden die Daten per Email übergeben. Es muss daher noch die Festlegung der Inhaltscodierung und eine Angabe über den verwendeten Zeichensatz folgen. Das Form-Tag bekommt hiermit abschließend folgende Notation:

```
<form
 action="mailto:webmaster@seiten-programmierung.de" method="post"
 enctype="text/plain" name="anfrage" id ="anfrage">
```

Nach Absenden wird sich der auf Ihrem Rechner installierte Email-Client einschalten und die Datenübermittlung ausführen. Testen Sie das Programm mit Ihrer eigenen Email-Adresse. Sie erhalten die im Formular benutzten Bezeichner zusammen mit den eingegeben Werten zugesandt.

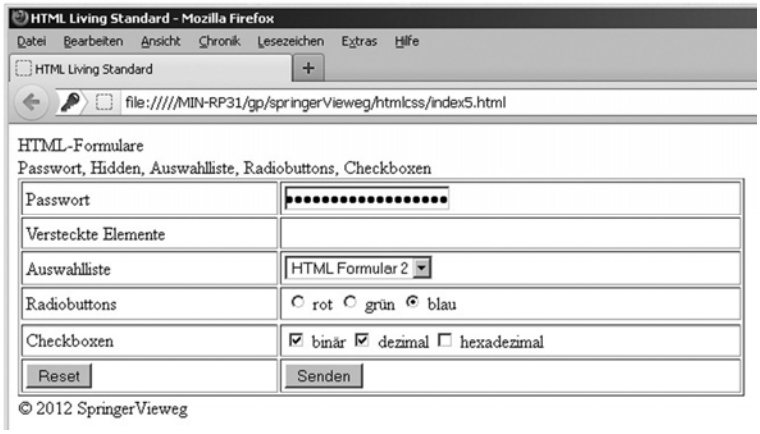

Abbildung 2.6: Gruppierung von Formularelementen

Betrachten wir nun weitere Typen des Input-Tags. Der Typ *password* gibt die eingegeben Zeichen unsichtbar wieder. Mit dem Typ *hidden* können Werte für den Besucher der Webseite unsichtbar bleiben.

```
<input type="password" name="pw" value="">
<input type="hidden" name="datei" value="index5.html">
```

Gruppen von Eingabewerten können durch Auswahllisten, Radiobuttons oder Checkboxen gebildet werden. Eine Auswahlliste wird mit dem Select-Tag aktiviert. Die Liste bekommt das Attribut `name` zur Identifikiation, das Attribut `size` gibt die Anzahl anzuzeigender Elemente an. Das Ereignis `onchange` reagiert auf eine erfolgte Auswahl und ist hier für Testzwecke eingefügt. Wir diskutieren die Anweisung später im Kapitel JavaScript. Die Auswahlmöglichkeiten einer Liste werden im Option-Tag angegeben. Der voreingestellte Wert bekommt das Attribut *selected*.

```
<select name="liste" size="1"
 onchange="alert('Selected Index: '+form.liste.selectedIndex +'\nWert : '
+form.liste.options[form.liste.selectedIndex].value)">
  <option>HTML Text</option>
  <option>HTML Tabelle</option>
  <option selected>HTML Image</option>
  <option >HTML Formular 1</option>
</select>
```

Radiobuttons und Checkboxen werden wieder mit dem Input-Tag ausgezeichnet. Die Gruppierung erfolgt hier durch den gleichen Namen. Bei Radioboxen können Sie nur eine Auswahl treffen, mit Checkboxen sind mehrere Auswahlmöglichkeiten gegeben. Der voreingestellte Wert wird mit dem Attribut *checked* versehen. Das Formular der Abbildung 2.5 wird per eMail übermittelt. Bei Klick auf den Submit-Button *senden* meldet sich der installierte eMail-Client und sendet das Formular mit Bezeichnern und eingegebenen Werten an den Empfänger, wie aus der Abbildung 2.6 zu erkennen ist. Klickt man den Reset-Button *reset*, dann werden alle Benutzereingaben auf die Voreinstellungen zurückgesetzt.

```
<p>Radiobuttons</p>
<input type="radio" name="farbe" value="R" checked /> Rot
<input type="radio" name="farbe" value="G" /> Gruen
<input type="radio" name="farbe" value="B" /> Blau
<p>Checkboxen</p>
<input type="checkbox" name="sys" value="bin" checked /> binär
<input type="checkbox" name="sys" value="dez" /> dezimal
<input type="checkbox" name="sys" value="hex" /> hexadezimal
```

Die Gruppe der Radiobuttons hat die Bezeichnung *farbe* mit einem voreingestellten Wert *R*. Checkboxen sind mit *sys* bezeichnet, der voreingestellte Wert ist *bin*. Im Programm-Archiv *htmlcss* finden Sie die Codierung unter *index5.html*.

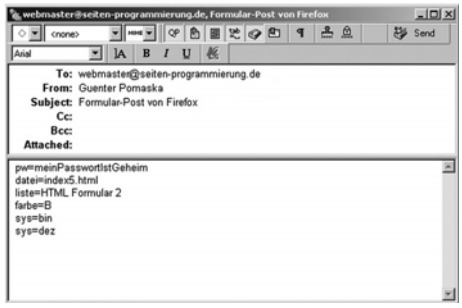

Abbildung 2.7: Absenden des Formulars per eMail

Mit HTML5 hat sich die Handhabung von Formularen praktisch nicht geändert. Es wurden aber neue Typen und Funktionalitäten eingeführt. Die Input-Typen `number`, `telephone`, `url`, `email` und `datetime` sind im folgenden Code-Snippet in einem Fieldset-Tag mit einer Erläuterung, Legend-Tag, zusammengefasst. Abbildung 2.7 zeigt einen Screenshot mit aufgeklapptem Kalender im Browser Opera.

```
<fieldset>
  <legend>Neue HTML5 Input-Typen</legend>
  Zahl <input type="number" name="zahl" value="" required /><br/>
  Telefon <input type="telephone" name="fon" value="" required/><br/>
  URL <input type="url" name="url" value="www.seiten-programmierung.de"
    required /><br />
  eMail<input type="email" name="mail" value="" required /><br />
  Datum <input type="datetime" name="datum" value="" required /><br />
</fieldset>
```

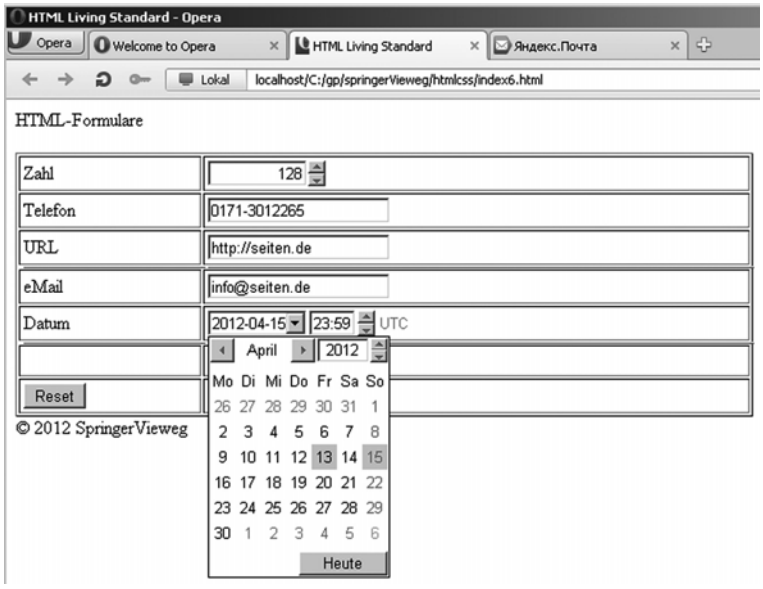

Abbildung 2.8: HTML5 Input-Typen im Browser Opera

Formulare haben in HTML5 eine eigene Validierungs-API, z.B. sind zulässige Wertebereiche bereits im Code definierbar. Formularfelder, Fieldsets oder das ganze Formular können im Browser validiert werden. Die einfachste Form der Validierung erfolgt über das Required-Attribut. Das Element selbst wird keiner Validierung unterzogen, es muss aber vorhanden sein.

Die Browserunterstützung der neuen Typen ist noch nicht umfassend. Aufgrund der *Graceful Degradation*, der angemessenen Reaktion und Behandlung von Fehlern, eines Browsers ist die Verwendung aber bereits jetzt durchaus möglich.

2.1.6 Hyperlinks

Hyperlinks oder kurz Links sind Verweise in einem HTML-Dokument, mit denen bei Aktivierung durch Mausklick

- eine andere Position im Dokument
- eine neue Datei
- oder ein neuer Web-Server

aufgerufen wird. Zusätzlich zu diesen Verweisen existiert auch die Möglichkeit eines Email-Links oder Dateiverweises. Dateiverweise bewirken den Download oder die Ausführung eines externen Programms.

Sensitive Bereiche für Hyperlinks können Texte, Grafiken, Bereiche in Grafiken (image maps) oder Buttons sein. Man bezeichnet die ausgezeichneten Bereiche auch als Schaltflächen. Ein Hyperlink wird mit dem Anchor-Tag `<a>` ausgezeichnet und benötigt als wichtigstes Attribut die Hypertextreferenz `href`. Andernfalls wird das ausgezeichnete Element nicht sensitiv für die Mausberührung. Die Ele-

mente innerhalb des Anchor-Tags stellen die Schaltfläche dar. Standardmäßig wird ein Text, der als Link ausgezeichnet ist, in blauer Farbe und unterstrichen dargestellt. Die angeforderte URL wird bei Mausberührung in der Statuszeile des Browsers angezeigt.

Dem Attribut `href` wird die aufzurufende Referenz zugewiesen. Die Anzeige kann im eigenen Fenster oder in einem neuen Fenster bzw. in einer neuen Registerkarte erfolgen. Wertzuweisungen an das Attribut `target` sind *_blank* bzw. *_self*. Sofern Sie einen Link in einem Iframe angeben möchten, muss der Iframe über seinen Namen identifizierbar sein. Der Name des Iframes wird im Target-Attribut als Ziel angegeben. Sie können das folgende Code-Snippet in Ihrem Browser testen.

```
<body>
  <iframe width="320" height = "240" name ="myFrame_1"
     src="http://imagefact.de/home.html"></iframe><br>
  <a href="http://divide-by-zero.com" target ="_blank">
  Target_blank</a><br>
  <a href="http://divide-by-zero.com" target ="_self">
   Target _self</a><br>
  <a href="http://divide-by-zero.com" target ="myFrame_1">
            Target myFrame_1</a>
</body>
```

Zunächst betrachten wir nur die Verweise im eigenen Dokument. Es wird ein langer Blindtext benutzt, der in Kapitel unterteilt ist. Am Anfang des Textes befindet sich eine Linkzeile wie folgt:

```
<p>
  [ <a name="anfang" href="#k1">Kapitel 1</a> |
          <a href="#k2">Kapitel 2</a> |
          <a href="#k3">Kapitel 3</a> |
          <a href="#k4">Kapitel 4</a> ]
</p>
```

Der erste Anchor-Tag ist gelabelt und hat den Namen *anfang* erhalten. Zu dieser Marke kann innerhalb des Dokuments gesprungen werden. Das sensitive Element für diesen Tag ist der Text *Kapitel 1*. Dieses Element verweist auf die Sprungmarke *k1* im selben Dokument, gekennzeichnet durch das Doppelkreuz #, auch Hash-Zeichen genannt. Die Hyperlinks *Kapitel 2, Kapitel 3* und *Kapitel 4* haben keine eigene Bezeichnung, verweisen aber auf die Marken *k2, k3, k4*.

Vor jedem Kapitel befindet sich ein Anchor-Tag mit der Bezeichnung (`name="`*k1*`"`) gefolgt von der Linkzeile, mit der an den Anfang des Dokumentes, ein Kapitel zurück oder ein Kapitel vorgesprungen werden kann.

```
<a name="k1"><h5>Kapitel 1</h5></a>
 <p>[
  <a href="#anfang">anfang</a> |
  <a href="#k2">vor</a> |
  <a href="#anfang">zurück</a> ]
</p>
```

Die Hyperlinks werden durch Voreinstellung blau und unterstrichen gekennzeichnet. Bei Berührung mit der Maus verändert sich der Cursor, das Ziel wird in

der Statuszeile angezeigt. Bereits besuchte Links werden farblich markiert. Löschen Sie die Chronik besuchter Seiten um zur blauen Markierung zurück zu kehren. Der Quelltext für die obige Anwendung hier noch einmal im Zusammenhang:

```
<!-- htmlCSS/index7.html -->
<div id="content">
  <h4>Gumminbärchen</h4>
  <p>
  [ <a name="anfang" href="#k1">Kapitel 1</a> |
          <a href="#k2">Kapitel 2</a> |
          <a href="#k3">Kapitel 3</a> |
          <a href="#k4"">Kapitel 4</a> ]</p>
  <a name="k1"><h5>Kapitel 1</h5></a>
  <p>[ <a href="#anfang">anfang</a> | <a href="#k2">vor</a> |
     <a href="#anfang">zurück</a> ]</p>
  <p> Freilebende:...Auch das macht </p>
  <a name="k2"><h5>Kapitel 2</h5></a>
  <p>[ <a href="#anfang">anfang</a> | <a href="#k3">vor</a> | <a
     href="#k1">zurück</a> ]</p>
  <p>Und da sie weich sind ... und aromatisiert.</p>
  <a name="k3"><h5>Kapitel 3</h5></a>
  <p>[ <a href="#anfang">anfang</a> | <a href="#k4">vor</a> |
  <a href="#k2">zurück</a> ]</p>
  <p>Mag sein, dass ... Was das schmecken</p>
  <a name="k4"><h5>Kapitel 4</h5></a>
  <p>[ <a href="#anfang">anfang</a> | <a href="#ende">vor</a> | <a
     href="#k3">zurück</a> ]</p>
  <p>Zigaretten auf ... und wieder geht es mir</p>
  <p>[ <a name="ende" href="#anfang">anfang</a> ]</p>
</div>
```

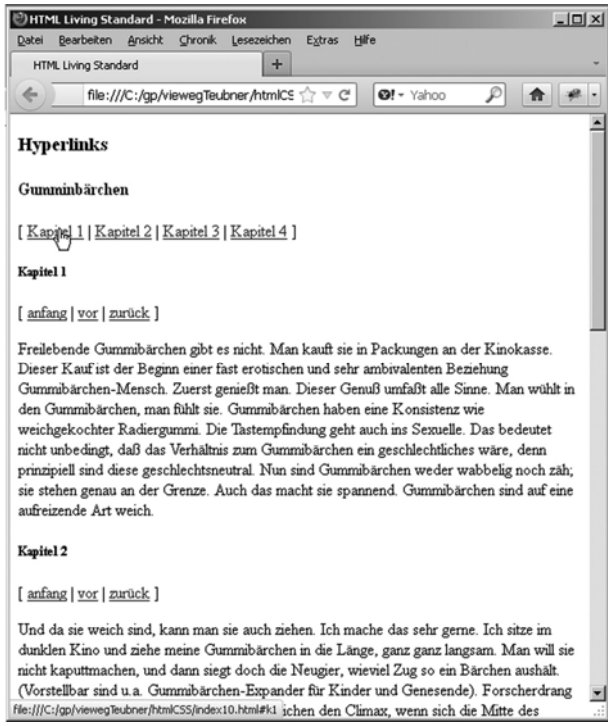

Abbildung 2.9: Verweise im eigenen Dokument

Eine neue Datei wird durch die relative Pfadangabe referenziert, hier mit der Zielangabe eines neuen Fensters. Dabei können Sprungmarken in der Datei unmittelbar angesteuert werden:

```
<a href="../pfad/neuedatei.html#kap1" target ="_blank">
  zur neuen Seite</a>
```

Ruft man einen neuen Web-Server auf, so ist vor die URL das Präfix `http://` zu setzen. Eine elektronische Mail kann über das Präfix `mailto:` abgesetzt werden. Es muss jedoch beim Anwender ein Mail-Client installiert sein. Gleichzeitig können Eintragungen für den Client mit angegeben werden.

```
<a href="http://www.google.de" target="_blank">Google-Suchmaschine</a>
<a href="http://w3.org" target="_blank">WWW Consortiun</a>
<a href="http://de.selfhtml.org" target="_blank">SelfHTML-Referenz</a>
<a href="mailto:webmaster@seiten-programmierung.de
  ?cc=meineEmail@imagefact.de&subject=Anfrage Programmierung
  &body=Ihre Anfrage
  bitte:" target="_blank">an Webmaster<br>
</a>
```

Wird statt des Textes ein Image-Tag zwischen `<a></a>` eingefügt, dann stellt die Grafik die Schaltfläche dar. Es muss aber nicht immer die gesamte Bildfläche als Link definiert sein. Eine Grafik kann mehrere Bereiche enthalten, die auf unterschiedliche Referenzen hinweisen. Man definiert für das Bild eine Map. Dies erfolgt entweder manuell mit Spezifikation der Bereiche (auch mit polygonaler oder kreisförmiger Begrenzung) oder besser durch Unterstützung eines HTML-Editors. Die Einheiten in der Bereichsdefinition sind Pixelkoordinaten.

Nachfolgend ein kleines selbsterklärendes Beispiel. Der Image-Tag mit Verweis auf ein Bild *personen.jpg* ist mit dem Attribut `usemap` versehen, das auf die Verknüpfung mit einer Image-Map mit Namen *map1* in diesem Dokument hinweist. Der Tag `<map>` mit dem Attribut `name = "map1"` definiert die Bereiche durch den Area-Tag. Mit dem Attribut `shape` wird die Form des Bereichs festgelegt, `coords` enthält die Koordinaten bezogen auf das Image. Es sind drei Rechtecke mit der jeweiligen Referenz auf unterschiedliche HTML-Dateien und der Angabe des Zielfensters definiert.

```
<!-- htmlcss/index8.html -->
<h4>Hyperlinks in Image Maps</h4>
<img src="../img/personen.jpg" alt="Klicken Sie auf eine Person"
    width="200" height="150" border="0" usemap="#map1" />
<map name="map1">
<area shape="rect" coords="130,10,176,128"
  href="person_rechts.html" target="_self" />
<area shape="rect" coords="80,15,123,129"
  href="person_mitte.html" target="_self" />
<area shape="rect" coords="28,15,76,129"
  href="person_links.html" target="_self" /></map>
<h4>Wählen Sie eine Person</h4>
```

Der Anchor-Tag hat noch ein weiteres wichtiges Attribut. Mit `target` wird das Ziel eines Links vereinbart. Der Wert *_blank* öffnet ein neues Fenster oder einen neuen Tab, mit *_self* wird in das aktuelle Fenster gelinkt. Sofern Sie einen Link in

einem Iframe angeben möchten, muss der Iframe über seinen Namen identifizierbar sein. Der Name des Iframes wird im Target-Attribut als Ziel angegeben. Sie können das folgende Code-Snippet in Ihrem Browser testen.

```
<body>
 <iframe width="320" height = "240" name ="myFrame_1"
     src="http://imagefact.de/home.html"></iframe><br>
 <a href="http://divide-by-zero.com" target ="_blank">
   Target_blank</a><br>
 <a href="http://divide-by-zero.com" target ="_self">
   Target _self</a><br>
 <a href="http://divide-by-zero.com" target ="myFrame_1">
   Target myFrame_1</a></body>
```

Abbildung 2.10: Hyperlinks mit Image-Maps

2.1.7 Multimedia – Audio, Video und Applets

Bisher galt das Flash-Format von Adobe für die Internetübertragung von Videoinhalten als Standard. Die Daten eines Videos müssen vom Videoplayer decodiert werden. Flash unterstützt u.a. die Codecs MPEG-4 und für das Audio-Format MP3. Die Wiedergabe eines Flash-Videos erfolgt über ein Plug-in im Web-Broser oder Abspielen in einem externen Videoplayer. Das unterschiedliche Verhalten der Web-Broser wurde umgangen, indem ein Embed-Tag in den Object-Tag eingebunden wurde. Das folgende Code-Snippet gibt eine MP3-Audio Datei mit dem QuickTime-Player wieder.

```
<object
 classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
 codebase="http://www.apple.com/qtactivex/qtplugin.cab"
 height="556" width="320" bgcolor="#ff0000">
  <param name="src" value="zukunftstag.mp3" />
  <param name="autoplay" value="true" />
  <param name="target" value="myself" />
  <param name="controller" value="true" />
  <param name="href" value="zukunftstag.mp3" />
```

```
  <param name="type" value="video/quicktime" height="24" width="320" />
   <embed src="zukunftstag.mp3" height="24" width="320"
     autoplay="true" type="video/quicktime"
       pluginspage="http://www.apple.com/quicktime/download/"
       controller="true" href="zukunftstag.mp3" bgcolor="yellow">
   </embed>
</object>
```

Eine Video-Lösung, die auch für HTML5 noch interessant ist, bietet der JW Player von Longtail Video an, *www.longtailvideo.com*. Nachfolgend der Code zur Wiedergabe eines Flash-Formats. Bitte verifizieren Sie den im Code notierten Object-Tag.

Unter der Bezeichnung *movie* wird der Videoplayer *player-viral.swf* zugewiesen, es folgen weitere Playereinstellungen und die Zuordnung des abzuspielenden Videofiles *.flv. SWF steht für ein Shockwave-Flash Object, FLV bezeichnet das von Adobe entwickelte Containerformat für ein Flash-Video. Zusätzlich wird noch *yt.swf,* eine Javascript-Datei *swfobject.js* sowie ein Vorschaubild *preview.jpg* benötigt. Alle Dateien befinden sich auf dem Server im selben Unterverzeichnis mit der Videoquelle *.flv.

Das Beispiel können Sie als Zip-Archiv *htmlcss/audioVideo/multimedia.zip* von der Webseite zum Buch herunterladen.

```
<object id="player" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    name="player" width="480" height="384">
  <param name="movie" value="player-viral.swf" />
  <param name="allowfullscreen" value="true" />
  <param name="allowscriptaccess" value="always" />
  <param name="flashvars"
     value="file=2010_12_10_clipAvi01.flv&image=preview.jpg" />
  <embed
    type="application/x-shockwave-flash" id="player2" name="player2"
    src="player-viral.swf" width="480"   height="384"
    allowscriptaccess="always" allowfullscreen="true"
    flashvars="file=2010_12_10_clipAvi01.flv&image=preview.jpg">
  </embed>
</object>
```

Mit HTML5 gestaltet sich die Einbindung von Multimediaelementen einfacher. Derzeit sind die Video-Codecs H.264, Theora und VP8 aktuell. Bei den Audio-Codecs spielen AAC, MP3 und Vorbis eine Rolle. Video und Audio werden in einem Containerformat gespeichert. Die gängigen Formate sind OGG, MP4 und WebM mit unterschiedlicher Browserunterstützung. Ohne Diskussion weiterer Details kann eine einfache browser-übergreifende Lösung angeboten werden. Konvertieren Sie Ihr Video in die Formate OGG (Theora, Vorbis) und WebM (VP8, Vorbis) und bieten Sie ein alternatives Format zum Download an. Mit Firefogg steht Ihnen ein Konvertierungswerkzeug nach Installation im Firefox-Browser im Menü `Extras > Make Web Video` zur Verfügung. Der HTML5-Code gestaltet sich mit dem Video-Tag sehr einfach.

```
<video poster ="vorschau.jpg" controls>
  <source src="oeynhausen.ogg">
  <source src="oeynhausen.webm">
  <a href="oeynhausen.flv">Video downloaden</a>
</video>
```

Mit der Eigenschaft `poster` wird ein Vorschaubild eingefügt. `Controls` ist ein boolescher Parameter, der die Bedienungselemente zeigt. Kann der Browser keines der beiden Formate darstellen, wird der Download angeboten. Die native Einbindung der Videodarstellung hat den Vorteil, dass man auf das Element wie auf alle anderen HTML-Elemente im Dokument zugreifen kann. Das gilt für JavaScript APIs oder die Stilvorlagen, wie man an den Bordereigenschaften in der Abbildung 2.10 sehen kann. Das Boxmodell der Stilvorlage wurde auf den Video-Tag angewandt.

Abbildung 2.11: Native Videoeinbindung

Die Behandlung des Audio-Tags erfolgt analog zum Video-Tag. Hier ist das Code-Snippet:

```
<audio controls>
  <source src="atommuell.ogv">
  <source src="atommuell.mp3">
  <a href="atommuell.mp3">Audio herunterladen</a>
</audio>
```

2.1.8 Java Applets

Applets sind kleine Java-Programme, die in einem Browser ablauffähig sind. Für den technisch-wissenschaftlichen Bereich existiert eine Vielzahl nützlicher Java-Applets zur interaktiven Visualisierung mathematischer Vorgänge, die mit anderen Werkzeugen nicht derart geeignet zu programmieren wären. Aus Sicherheitsgründen unterliegen Applets Beschränkungen. Ein Zugriff auf das Dateisystem des Clientrechners ist z.B. nicht möglich. Die Plattformunabhängigkeit von Java wird durch Übersetzung des Quellcodes in einen Bytecode für eine virtuelle Maschine realisiert. Die virtuelle Maschine, das Java Runtime Environment (*http://www.java.com/de)* muss auf dem Clientrechner installiert sein.

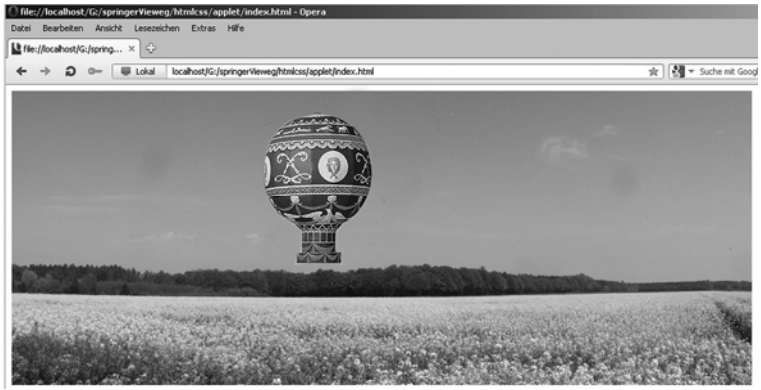

Abbildung 2.12: Java-Applet Animation mit Bitmaps

Seit HTML5 wird der Applet-Tag nicht mehr unterstützt. Man benutzt daher wie oben bereits gezeigt ein Konstrukt aus Object- und Embed-Tag. Dabei ist das Applet von außen über die HTML-Tags steuerbar. Innerhalb der Java-Codierung werden Variablen durch die Funktion *getParameter(name)* Werte zugeweisen. Der Name entspricht dem Namen des Paramters im Object- bzw. Embed-Tag.

```
<body>
<object classid="java:Animation.class" codetype="application/java-vm"
    width="960" height="380">
  <param name="imgBackground" value="hintergrund.png" >
  <param name="imgObject" value="montgolfiere.png" >
  <param name="mySound" value="wind.wav" >
  <param name="speed" value="10">
  <param name="offsetY" value="75">
 <embed type="application/x-java-applet"
     code="Animation.class" width="960" height="380"
     imgBackground ="hintergrund.png"
     imgObject="montgolfiere.png"
     mySound ="wind.wav"
     offsetY="100"
     speed="10">
 </embed>
</object>
</body>
</html>
```

In der vorliegenden Anwendung haben wir es mit einer einfachen, auf Bitmaps basierenden Animation, zu tun. Die Eigenschaften des Object-Tags definieren die Applikation und den Bereich des Applets. Der Param-Tag definiert für die Bezeichner, die im Java-Applet benutzt werden, die Werte. Für die Geschwindigkeit stellt sich die Java-Codierung wie folgt dar:

```
String text = getParameter("speed");
if (text == null){
  // Voreinstellung
  wait = 75;
}else{
  // Umwandlung String > Integer
  wait =Integer.parseInt(text);
}
```

Aus der HTML-Datei wird eine Zeichenkette gelesen und der Variablen zugewiesen. Bei fehlender Angabe wird eine Vorbesetzung gültig. Der Sourcecode des Applets bleibt dem Anwender aber verschlossen, er muss nur die Bezeichner der Parameter und deren Wertebereiche kennen. Hier werden benötigt: Hintergrundbild, das zu bewegende Objekt und eine Sounddatei. Einstellbar sind der Abstand des Objekts vom oberen Rand und die Geschwindigkeit der Animation.

In den bisherigen Abschnitten dieses Kapitels wurden die Elemente eines HTML-Dokumentes behandelt. Es wurde gezeigt wie Texte zu strukturieren sind, Aufzählungslisten und Tabellen behandelt werden und Grafiken sowie Multimedia-Elemente einzubinden sind. Hyperlinks dienen der Navigation auf einer Seite und der Verknüpfung externer Dokumente. Darüber hinaus stellen Formulare ein komfortables Benutzerinterface zu weiterverarbeitenden Programmen dar. Im folgenden Kapitel geht es um die Ausgestaltung der Elemente, der Positionierung dieser Elemente und der Anordnung der Seitenstruktur.

2.2 CSS Cascading Style Sheets

Mit der Trennung von Struktur und Inhalt gegenüber dem Erscheinungsbild einer Webseite verbinden sich etliche Vorteile bei der Administration und Programmierung einer Web-Präsenz. Die Formatierung wird redundanzfrei an nur einer Stelle definiert. Ein einheitlicher Stil für alle Seiten ist gewährleistet. Änderungen können zentral innerhalb einer Datei vorgenommen werden. Verschiedene Ausgabegeräte oder Bildschirmauflösungen können durch Verknüpfung mit den entsprechenden Stilvorlagen bedient werden.

Die Syntax der Formatsprache für das Web ist durch den Standard CSS Cascading Style Sheets vom W3C definiert. Ein erster Vorschlag für Stilvorlagen erfolgte 1993. Bereits 1996 war CSS Level 1 eine Empfehlung des W3C, die schon 1998 als CSS2 fortgeschrieben wurde. Derzeit wird CSS Level 3 von der CSS Working Group in eigenständigen Modulen entwickelt. Unterschiedliche Wiedergaben mit den verschiedenen Browsern sind heute nicht mehr so gravierend. In diesem Kapitel werden die etablierten Standards behandelt und mit den aktuellen Versionen der Browser getestet. Es kann nicht Gegenstand dieser Einführung sein, browserübergreifend alle aktuellen Entwicklungen auch für ältere Browser zugänglich zu machen.

Stylesheet-Dateien können mit einem einfachen Texteditor bearbeitet werden. Heutige HTML-Werkzeuge bieten fast alle Unterstützung bei der Bearbeitung von Stilvorlagen. Spezielle Stylesheet-Werkzeuge liefern den entsprechenden Komfort. Im Anschluss an die Fertigstellung von Inhalt und Struktur einer Seite erstellen Sie die Stildefinition mit Unterstützung eines derartigen Werkzeuges. Da wir die portablen Applikationen im Fokus haben, wird hier der CSS-Editor Oiko von *www.css-editor.info* in unseren Werkzeugkasten aufgenommen. Sie brauchen also in keinem Fall die einzelnen Attribute und Werte parat zu haben. Erwerben Sie sich hier den

notwendigen Umfang an Kenntnissen über die Struktur und Regeln der Sprache. Zur Erstellung der Stilvorlagen setzen Sie das auf Ihrem USB-Stick installierte Werkzeug ein.

Bevor wir in die Thematik einsteigen, noch einige kurze Anmerkungen zur ansprechenden grafischen Gestaltung eines Web-Dokuments. Dieses Buch ist keine Literatur über Web-Gestaltung. Natürlich wollen wir unsere Seiten auch mit einem ansprechenden Layout ausstatten. Versuchen Sie sich aber nicht mit Gewalt in Mediendesign. Vermeiden Sie den Multimedia-Overkill. Beschränken Sie sich auf einen Zeichensatz. Benutzen Sie drei Farben, eine helle Pastellfarbe für den Hintergrund und eine dazu passende dunkle Farbe für den Text. Für Hervorhebungen können Sie noch eine Kontrastfarbe wählen. Die Farben sollen der Thematik der Web-Site entsprechen. Hilfestellung zur Farbwahl finden Sie im Netz, googeln Sie doch einfach nach *Farbgestaltung für das Web*. Bei *www.webmart.de* finden Sie ein Online-Tool Color Schemer Online 2.0. Auch bei *www.visibone.com* gibt es den freien Online-Service zur Farbgestaltung.

2.2.1 CSS Syntax und Einbindung in HTML

CSS Definitionen beginnen mit einem Selektor. Ein Selektor ist ein Muster zum Auffinden von Elementen im HTML-Dokument. In geschweiften Klammern folgen dann eine oder mehrere Deklarationen. Deklarationen sind die Attribute bzw. Eigenschaften und deren Wert, voneinander getrennt durch Doppelpunkt. Mehrere Deklarationen werden durch das Semikolon getrennt. Es gibt unterschiedliche Arten von Selektoren, ein Typselektor bezieht sich auf die Tags eines HTML-Dokuments. Regeln, die gleiche Deklarationen enthalten können durch Angabe mehrerer Selektoren zusammengefasst werden. Die Syntax einer Stilvorlage für die Headings und einen Textabschnitt könnte folgende Gestalt annehmen:

```
/* CSS-Stilvorlage exemplarisch */
h1,h2,h3 {
 font-style: italic;
 font-weight:bold
}
P {
 color: #0000ff;
 text-align:justify
}
```

Ein Kommentar in CSS wird mit `/* */` begrenzt. Typselektoren sind im obigen Beispiel h1, h2, h3 und p. Auszeichnungen im HTML-Code mit h1-h3 werden kursiv und fett dargestellt. Ein Textabschnitt wird in der Farbe Blau als Blocksatz gerendert. Das Semikolon als Trennzeichen ist bei der letzten Regel entbehrlich.

Zur Einbindung von CSS Stilvorgaben in HTML-Dokumente existieren die Alternativen:

- Inline
- Style-Tag im Kopf des HTML-Dokuments
- Exerne Referenzierung

Beim Inline-Stylesheet wird die Regel als Attribut im HTML-Tag mit angegeben

```
<p style="{color:#ff0000; text-align:justify}"> … der Text … </p>
```

Die eingebettete (embedded) Stilvorlage wird im Head-Bereich der Seite notiert

```
<head>
<style type="text/css">
  /* Stilvorgaben wie oben angegeben */
  /* selektor {attribut:wert;} */
</style>
</head>
```

Obige Varianten eignen sich für die Testphase. Der Philosophie einer Stilvorlage kann nur die externe Referenzierung über einen Link-Tag im Kopfbereich der Seite gerecht werden:

```
<link href="http://www.seiten-programmierung.de/css/standard.css"
      rel="stylesheet" type ="text/css" media="screen" />
<link href="../css/drucken.css" rel="stylesheet" type ="text/css"
      media="print" />
```

Es können mehrere Stilvorlagen extern eingebunden werden. Dabei kann das Dokument relativ oder absolut adressiert werden. Mit der Angabe `Media` werden unterschiedliche Stilvorlagen bei Bildschirm und Druckerausgabe berücksichtigt. Die Erweiterungsbezeichnung der Stilvorlage ist `css`, der Style-Tag wird nicht in die Datei mit aufgenommen.

2.2.2 Warum Cascading?

Stilvorlagen können vom Autor einer Webseite festgelegt werden, der Surfer kann ggf. über den Browser eine Stilvorlage auswählen oder der Browser selbst verwendet sein internes Stylesheet. Die Gewichtung der Stilvorlagen wird in der Reihenfolge Browser, Benutzer, Autor vorgenommen. Ausnahmen können durch die Deklaration `important` in der Autoren- oder Benutzer-Stilvorlage vorgenommen werden. Aber gehen wir hier mal davon aus, dass die Regeln des Autors der Webseite nicht vom Surfer abgeschaltet werden.

Mehrere Stilvorlagen können zu einem gemeinsamen Stylesheet zusammengefasst werden. Man könnte auf Bausteine zurückgreifen und hier individuelle Änderungen an ausgesuchten Regeln vornehmen. Überschreibungen werden durch den Import-Befehl erzeugt. Hierdurch werden Stilvorlagen nacheinander importiert. Bei gleichen Regeln überschreiben die zuletzt importierten Regeln die zuvor gültigen. Das gilt dokumentenübergreifend, aber auch bei Auszeichnungen innerhalb einer HTML-Datei dokumentenweit.

```
<style type="text/css">
  @import "http://www.seiten-programmierung.de/css/stilvorlage.css";
  @import "meinestilvorlage.css";
</style>
```

2.2.3 Vererbung der Regeln

HTML-Elemente unterliegen einer Hierarchie. Elemente, die von anderen umschlossen werden, sind untergeordnete Elemente. Im unten aufgeführten Beispiel

ist Kind von <p> und <p> wiederum Kind von <body>. Jedes der Elemente hat ein Elternelement, von dem es die Eigenschaften erbt. Die CSS-Eigenschaften würden von <body> an die Unterelemente vererbt werden. Diese Vererbung kann aber durch Überschreiben der Regeln aufgehoben werden. Werden die Regeln von <p> überschrieben, dann erbt den Stil von <p>.

```
<body>Text im Body
  <p>Absatztext <em> Hervorhebung </em>
    weiter im Absatztext
  </p>
</body>
```

Im Firexox-Browser können Sie als Add-on den DOM-Inspector installieren. Unter dem Menüpunkt `Extras>Web-Entwickler>Dom-Inspector` wird Ihnen die Hierarchie des Dokuments angezeigt.

2.2.4 Selektoren

Einleitend wurde bereits der Typselektor erwähnt, dessen Regeln auf die entsprechenden HTML-Tags angewandt werden. Im Folgenden verschaffen wir uns zunächst einen Überblick über die gängigen Selektoren. Der Universalselektor '*' bezieht sich auf alle HTML-Tags. Wir können z.B. die Abstandsvorgabe und die Polsterung aller Elemente zurücksetzen mit:

```
*{
   margin 0px;
   padding 0px;
}
```

Sollen sich Elementtypen in der Darstellung unterscheiden, so vergibt man Klassen oder IDs. Klassen können im Dokument mehrfach vorkommen, eine ID darf nur einmal vorkommen und bezeichnet genau ein Element. Es wird unterschieden in Klassenselektor und ID-Selektor. Der Klassenselektor beginnt mit einem Punkt '.', der ID-Selektor mit dem Gatterzeichen '#' jeweils gefolgt von der Bezeichnung der Klasse oder der ID. Ein Textabschnitt mag das Attribut *hervorgehoben* haben.

```
<p class="hervorgehoben"> … der Text </p>
```

Diesen Abschnitt wollen wir dann mit Großbuchstaben versehen. Die Stilvorlage bekommt die Regel.

```
.hervorgehoben{
  text-transform: uppercase
}
```

Diese Regel gilt dann für alle Elemente der Klasse. Sollen nur bestimmte Elemente einer Klasse den Regeln folgen, dann folgt auf den Typselektor getrennt durch Punkt der Klassenbezeichner:

```
p {
  font-size : 100% }
p.fussnote {
  font-size : 75%; }
p.anmerkung {
  font-style : italic; }
```

Analog gilt die Regel für eine ID. Trägt z.B. eine Artikel eine ID und nur die Hervorhebungen dieses Artikels sollen in grün formatiert werden dann wird wie folgt notiert:

```
<!-- im HTML-Dokument -->
<article id="2012-04-20">
...Text <em>Hervorhebung</em> ... weiter im Text
</artikel>
```

und in der Stilvorlage:

```
em#2012-04-20 {
  color:#00FF00;
}
```

Ein Selektor mit Nachfahren ist eine Liste von Selektoren getrennt durch Leerzeichen in der Reihenfolge von außen nach innen. Sie haben einen Container `section` der die Klasse *inhalt* in Bereiche `article` unterteilt, die einen `footer` enthalten. Im `article` befindet sich ein weiterer Container `div` mit ebenfalls einem `footer`, der rechtsbündig ausgerichtet werden soll. Der HTML-Code ist wie folgt formuliert:

```
<section class="inhalt">
 <article>
  <div>
   <p>Textabschnitt</p>
   <footer>nicht direkter Nachfahre von Section</footer>
   <div>
   <footer>&copy 2012</footer>
 </article>
</section>
```

Die zugehörige Stilvorlage mit dem Nachfahren-Selektor benötigt die folgende Notation:

```
.inhalt article footer {
 text-align:right;
}
```

Nachfahren-Selektoren berücksichtigen auch die Elemente, die nicht direkte Nachfahren eines Elementes sind. Im obigen Beispiel würde der Nachfahren-Selektor für beide `footer` greifen**.** Dieses Verhalten kann man durch Kind-Selektoren unterbinden. Kind-Selektoren verwenden für die direkten Nachfahren das *größer als* Zeichen.

```
.inhalt>article>footer {
   text-align:right;
}
```

Mit der Schreibweise Kind-Selektor würde nur ein `footer`, der unmittelbare Nachfahre, rechtsbündig angezeigt.

Pseudo-Klassen und Pseudo-Elemente erlauben Zugriff auf Elemente einer Webseite, deren Zustand erst durch Ereignisse entsteht oder die nicht als Element einer Webseite beschrieben sind. Beispiele sind die Mausberührung eines Links oder der erste Buchstabe eines Textabschnitts.

```
a:hover{
  color:#ff0000
```

```
}
P:first-letter{
  font-size:200%;
  font-weight:bold
}
```

Typselektor und Pseudo-Klasse werden durch Doppelpunkt getrennt. Im Beispiel wird die Mausberührung eines Links mit Rot ausgezeichnet, der erste Buchstabe eines Textabschnitts wird mit doppelter Größe und fett gerendert. Wir werden den Pseudo-Selektoren im weiteren Verlauf dieses Kapitels wieder begegnen.

2.2.5 Einheiten

Schriftgrößen können Sie in den folgenden Einheiten:

- Punkt pt
- Pica pc
- Pixel px
- Em em
- Prozentwert %

angeben. Ein Punkt hat eine feste Größe von 1/72 Zoll. Ebenfalls eine feste Größe hat der Pica mit 1/6 Zoll. Für die Bildschirmdarstellung sind diese Einheiten ungeeignet, da sie nicht skalierbar sind. Für Druckausgaben können diese Einheiten Verwendung finden. Pixelangaben sind für Schriftgrößen nur bedingt brauchbar. Skalierbare und zu empfehlende Einheiten sind Em und Prozentwerte. 1 em entspricht ungefähr der Größe des Buchstabens M. In CSS wird 1 em der Schriftgröße des Benutzers gleichgesetzt. Durch diese relative Größenangabe in Dezimalzahlen werden die Größeneinstellungen des Benutzers im Browser berücksichtigt. Ebenfalls geeignet sind Prozentwerte, die sich auf die Standardeinstellungen des Benutzers beziehen. In Prozentwerten entspricht 100% der Angabe 1 em. Schriftgrößen sind auch durch absolute (small, medium, large) und relative (smaller, larger) Schlüsselwerte definierbar. Die Interpretation bleibt dem Browser überlassen. Bei den relativen Größen kann es unerwartete Effekte durch die Schachtelungstiefe von Elementen geben, da die Eigenschaften an Kindelemente von den Elternelementen vererbt werden.

Längenangaben für die Breite und Höhe von Grafiken oder Positionen und Abstände kann man ebenfalls relativ oder mit fester Größe angeben. Für die Abmessungen von Grafiken sind üblicherweise Pixel angezeigt. Sofern Breite (width) und Höhe (height) der Originalgröße des Bildes entsprechen, hat der Browser keine Umrechnung vorzunehmen.

Farbwerte sind hexadezimal oder durch ganze Zahlen für RGB oder durch den vordefinierten Farbnamen anzugeben. Gleiche Bedeutung haben:

```
color: #FF0000;
color. Rgb(255,0,0);
Color: red;
```

2.2.6 Das Boxmodell

Jedes Element in CSS wird durch eine Box umrandet. Das eigentliche Element wird umgeben von einer Polsterung (padding), einem Rand (border) und einem Abstand zum Nachbarelement (margin). Wir betrachten das Beispiel der Abbildung 2.12 und den zugehörigen Sourcecode. In Bezug auf den Rand (border) können wir auch vom äußeren Randabstand (margin) und inneren Randabstand (padding) sprechen. Die Größe des Elements selbst plus Randbreite, Padding und Margin ergeben den benötigten Platz auf der Seite.

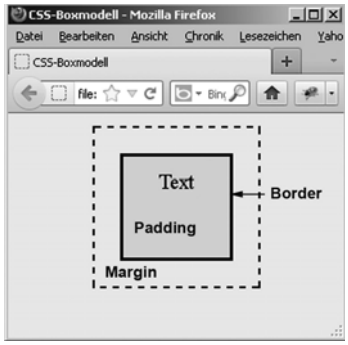

Abbildung 2.13: Boxmodell

```
<html lang="de">
  <head><title>CSS-Boxmodell</title>
    <style type="text/css">
      *{ margin:0px; padding:0px;}
      body {
        background-color:#eeeeee;
        border: 2px dashed #000000;
        margin-left:25%;
        margin-top: 12px;
        argin-right:25%
      }
      P {
        background-color:#dddddd;
        font-size: 125%;
        text-align:center;
        padding: 12px 12px 60px 12px;
        border: 3px solid #000000;
        margin: 24px 24px 24px 24px;
      }
    </style>
  </head>
<body>
  <p>Text</p>
</body>
</html>
```

Ein Textabschnitt enthält lediglich das Wort Text mit einer Schriftgröße von 125%. Die Hintergrundfarbe ist ein helles Grau. Die Abstände des Textes bis zum Rand betragen oben, rechts und links 12px, nach unten 60px. Die Notation `padding: 12px 12px 60px 12px;` läuft im Uhrzeigersinn und beginnt oben. Der Rand um den Text hat eine Stärke von 3px, durchgezogener Strichtyp in der Farbe schwarz. Die Abstände zum Body betragen 24px. Der Body hat einen oberen

Randabstand von 12px. Von der Fenstergröße werden jeweils 25% für den linken und den rechten Rand genutzt, die Begrenzung des Elements Body ist gestrichelt.

2.3 Seitengestaltung mit CSS

Nachdem Sie sich mit den Grundlagen von CSS vertraut gemacht haben und die Begriffe Selektor, Eigenschaft und Wert kennen, sowie auch die Einbindung in HTML kein Problem für Sie ist, beginnen wir mit dem Oiko CSS-Editor. Downloaden Sie die portable Version von *www.css-editor.info* als Zip-Archiv, entpacken Sie die Files in ein eigenes Verzeichnis z.B. *tools/cssEditor*. Mit Doppelclick auf *Oiko.exe* meldet sich der Editor zur Arbeit.

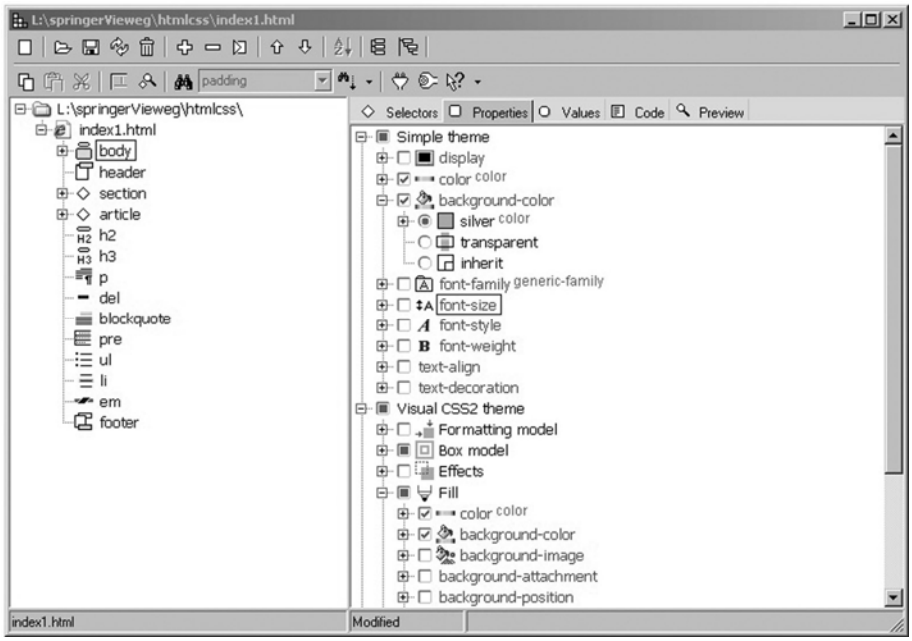

Abbildung 2.14: CSS-Editor Oiko, Selektoren und Attribute

Der Editor hat ein intuitives Interface und ermöglicht den grafischen Zugriff auf die CSS-Struktur. Selektoren, Eigenschaften und Werte werden in einer Baumstruktur angeboten. Die Vorschau eines Dokuments kann im Internetexplorer oder mit Firefox erfolgen. Der Bildschirm ist in einen Projekt-Bereich (links) und den Eingabebereich (rechts) aufgeteilt. Wir rufen unser HTML-Dokument mit den ersten Textbeispielen auf, das war *index1.html*. Benutzen Sie Steuerung + O oder aus der Toolbar Open File. Im Projektbereich wird die Datei angezeigt. Im Eingabe-Fenster können wir jetzt Themes auswählen. Angeboten wird CSS2 , XHTML und unser File *index1.html*. Wir markieren die Checkbox und bekommen automatisch alle im Dokument auftretenden Tags als Selektoren angezeigt. Wir wählen nun im linken Fenster einen Selektor mit Doppelklick aus und klappen rechts bei den Eigenschaften Simple Theme oder Visual CSS2 Theme auf. Bleiben wir zunächst

beim Bereich Simple Theme und setzen die Hintergrundfarbe sowie Margin und Padding für den Selector Body. Danach sieht unser Fenster so aus wie in der Abbildung 2.13, links die Selektoren, rechts die Eigenschaften. Wir können in die Vorschau gehen oder den Source-Code anzeigen lassen. Unsere Stilvorlage ist jetzt embedded. Die 10 % für Margin und Padding, die mit einem Radiobutton markiert wurden, können wir manuell ändern. Nach etwas Übung werden Sie sich mit dem Editor anfreunden und seine Vorzüge schätzen lernen.

2.3.1 Textformatierung

Mit Hilfe des CSS-Editors und manuellen Eingriff in den Quellcode soll nun ein Teil des Textes aus der Datei *index1.html* formatiert werden. Die HTML-Seitenstrukturierung wird reduziert auf `<body>` und `<article>`. Im Article-Bereich werden ein Header mit dem Class-Attribut *article_header*, ein Textabschnitt mit Einzug der ersten Zeile und ein Textabschnitt ohne Einzug eingefügt. Der Hintergrund aller Bereiche ist weiß, die Textfarbe wird in einem nicht zu dunklem Grau gehalten. Die Überschrift kann mit einer zweiten Farbe gestaltet werden, ebenso die Hervorhebung. Ein Auszug aus dem HTML-Code ist wie folgt zu notieren:

```
<body>
 <article>
  <header class="article">
    <h2>Thema Heading 2</h2>
    <h1>Hauptüberschrift Heading 1</h2>
    <h3>Autor</h3>
  </header>
    <p> ...Textbereich 1 ...</p>
    <p class="noindent">Deine ergebene ...</p>
  </article>
</body>
```

Abbildung 2.15: Textformatierung mit CSS

Die eingebettete Stilvorlage setzt im Body-Selector Randabstand und Padding auf null. Die Definition des Schriftsatzes beginnt mit speziellen Fonts, die bei Vorhandensein auf dem Clientrechner Anwendung finden. Sofern keiner der Zeichensätze präsent ist, wird ein generischer Schriftname angewandt. Schriftfarbe ist Grau. Sofern nicht durch Überschreibungen neu definiert, werden diese Deklarationen an die weiteren Selektoren vererbt. Für den Header-Bereich des Artikels werden die Deklarationen für die Überschriften `h1–h3` vorgenommen. Die Selektoren `h1–h3` sind Kindelemente des Headers. Einzelne Werte werden überschrieben. Elternelement für den Textabschnitt `p` ist `body`. Im Textabschnitt wird die Zeilenhöhe auf 150% des Standards gesetzt. Es wird Blocksatz definiert und die erste Zeile um 24px eingezogen. Dieser Wert wird für die Klasse *noident* wieder überschrieben. Zuletzt wird noch die Farbe für Hervorhebungen festgelegt.

```
<style type="text/css">
 body {
   margin: 0;
   padding: 0;
   font-family: Geneva,Arial,Helvetica, sans-serif;
   color: #444444;
 }
 article {
   border-color: #444444;
   border-width: 1px;
   border-style: solid;
   padding: 12px;
   margin: 12px;
 }
 header.article {
   font-style: normal;
   font-weight: normal;
   text-align: left;
   text-decoration: none;
      line-height: 100%;
   color: teal;
 }
 h1 {
   font-size: 125%;
 }
 h2 {
   font-size: 100%;
 }
 h3 {
   font-size: 90%;
   font-weight: normal;
 p {
   font-size: 90%;
   line-height: 150%;
   text-align: justify;
   text-indent: 24px;
   color: #444444;
 }
 p.noindent {
   text-indent: 0px;
 }
 em {
     color: teal;
 }
</style>
```

Die Validierung erfolgt mit den Entwickler-Tools von Firefox. Die Fehlerkonsole weist keine Fehler und Warnungen aus. Die Stilvorgaben werden ohne den Style-Tag unter *css/textformat.css* gespeichert und im Head-Bereich des HTML-Codes mit

```
<link rel="stylesheet" type="text/css" href="css/textStandard.css" />
```

eingefügt. Mit der zuvor erläuterten Vorgehensweise werden die weiteren Formatvorlagen entwickelt und können dann individuell als Bausteine Verwendung finden. Wir stellen fest, dass die Stilvorlagen in Gruppen auftreten, die auch vom Oiko-Editor in der Form in Themen angeboten werden. Wenn sie dann noch die Karteireiter Selektor, Eigenschaft, Wert (mit Sets, Colors, Fonts) beachten, verschaffen Sie sich schnell einen Überblick über die Möglichkeiten von CSS.

2.3.2 Gestaltung von Bildern

Im Abschnitt 2.1.4 wurde der Figure-Tag bereits vorgestellt. Dieser Tag beinhaltet die Grafikeinfügung mit dem Image-Tag und die Bildunterschrift durch den Figure-Caption-Tag. Für die Angabe der Bildabmessungen werden die Einheiten in Pixel als nicht skalierbare Größe angegeben. Zunächst ein Blick in den HTML-Code:

```
<figure>
  <img src="img/burgKarlstein.jpg" alt"=Bauwerk"
       width ="180px" height="135px" />
  <figcaption>Burg Karlstein</figcaption>
</figure>
<p> ...der Textabschnitt ... </p>
```

Der Figure-Tag wird vor dem Textabschnitt eingefügt. In der Stilvorlage wird mit der Eigenschaft `margin` ein äußerer Abstand vom Rand um das Element festgelegt. Mit der Eigenschaft `padding` wird innerhalb des Elements ein Abstand zum Rand eingefügt. Das Figure-Element soll nur auf der rechten Seite einen Abstand zum umfließenden Text erhalten. Man kann die Eigenschaften durch ein Suffix top, right, bottom oder left individuell vereinbaren oder in abgekürzter Form durch einen Wert für alle vier Seiten gleich mite einem Wert oder durch vier Werte angeben. Die Zuordnung beginnt oben in Uhrzeigerrichtung, vgl. im Attribut `padding` 9px für den rechten Rand.

Das Bild selbst soll einen dünnen Rand erhalten. Mit der Eigenschaft `padding` erzielen wir den inneren Abstand und mit der Eigenschaft `border` wird ein Rechteck um den Bereich gezeichnet. Darunter soll die Bildunterschrift mittig erscheinen. Die Bildunterschrift gestalten wir ohne Ränder.

Damit das Bild vom Text umflossen wird erhält es die Eigenschaft `float`. Blockelemente nehmen normalerweise die gesamte Breite innerhalb eines Elternelements ein. Mit der Eigenschaft `float` wird das Blockelement `<figure>` zum schwebenden Element erklärt und von weiteren Blockelementen ignoriert. Texte und darin eingebettete Bilder umfließen jedoch das schwebende Element. Das gerenderte Ergebnis ist in der Abbildung 2.15 zu sehen. Die Aufhebung eines Textumflusses erzielt man mit dem Attribut `clear` für das Folgeelement.

```
<style type="text/css">
figure {
  margin: 0px;        /* gilt für alle Ränder außen */
  padding: 0px 9px 0px 0px; /* Reihenfolge top, right, bottom, left */
  float:left         /* Element links vom umgebenden Element */
}
img {
  margin:0px; padding:2px;
  border: 1px solid #444444;
}
figcaption {
  padding:0px; margin: 0px;
  font-family: serif;
  font-weight: bold;
  font-size: 85%;
  text-align: center;
}
</style>
```

Abbildung 2.16: Textumfluss für das Element Figure

Beim Abstand der Bildränder zum umlaufenden Text kann bei der linken Ausrichtung `margin-right` und `margin-bottom` gesetzt werden. Bei der rechten Ausrichtung wird man sich für `margin-left` entscheiden.

2.3.3 Formulargestaltung

Ein Formular soll zwei zusammen gehörende Gruppen von Textfeldern zeigen, z.B. die Rechnungsanschrift und Lieferanschrift einer Bestellung. Wir wollen im Gegensatz zur Positionierung in einer Tabelle, vgl. Abschnitt 2.1.5, die Gestaltung ohne Tabelle durchführen. Es kommen die Elemente `<fieldset>`, `<label>` `<input>` und `<div>` zur Anwendung. Im ersten Abschnitt wird die Rechnungsanschrift eingegeben, im zweiten Abschnitt die Lieferanschrift. Standardmäßig wird `<fieldset>` umrahmt und oben links erscheint der Text aus dem Element <legend>. In einem Div-Container befinden sich jeweils die Elemente `<label>`und `<input>`. Der Submit-Button ist mittig in einem eigenen `<fieldset>` angeordnet.

```
<form>
  <fieldset>
    <legend>Rechnungsanschrift</legend>
    <div>
      <label for="nameRng">Name: </label>
      <input type="text" name="nameRng" /></div>
    <div>
      <label for="eMail">eMail: </label>
      <input type="text" name="eMail"></div>
  </fieldset>
  <fieldset>
    <legend>Lieferanschrift</legend>
    <div>
      <label for="nameLie">Name: </label>
      <input type="text" name="nameLie" /></div>
    <div>
      <label for="strasse">Strasse: </label>
      <input type="text" name="strasse"></div>
    <div>
      <label for="ort">Ort: </label>
      <input type="text" name="ort"></div>
  </fieldset>
  <fieldset class="noBorder">
    <input id="btn" type="submit" value="Bestätigen"/>
  </fieldset>
</form>
```

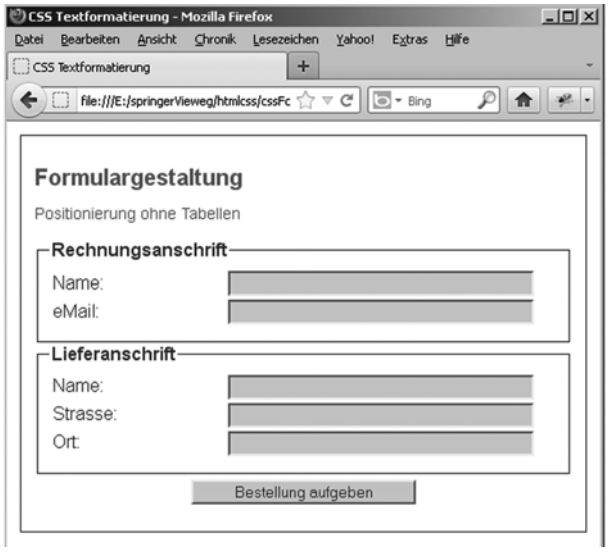

Abbildung 2.17: Formulargestaltung mit CSS ohne Tabellen

Für die Input-Tags wird in der Stilvorlage ein Bereich von 60% vorgesehen, die Hintergrundfarbe ist ein helles Grau. Die Label werden in der Form des Nachfahren-Selektors `form div label` mit 35 % angegeben mit dem Attribut `float` nach links angeordnet. Der Pseudo-Selektor `:focus` setzt beim Fokus auf ein Element die Hintergrundfarbe auf Gelb. Der Submit-Button wird durch `margin-left` und `margin-right` mit den Werten *auto* zentriert. Dieser Abschnitt ist mit der Klasse *noborder* ausgezeichnet.

```
<style type="text/css">
  form {
    background-color:white;
  }
  input{
    color:#444444;
    background-color:lightgray;
    width: 60%;
  }
  fieldset{
    font-family: Geneva,Arial,Helvetica, sans-serif;
    font-size: 100%;
    font-weight:bold;
    text-align: left;
    color: #444444;
    border: 1px solid #444444;
  }
  #btn{
    margin-left:auto;
    margin-right:auto;
    display:block;
     width:200px
  }
  input:focus{
    background-color:yellow;
  }
  form div {
    margin:3px
  }
  form div label {
    width: 35%; font-weight:normal; float: left
  }
  .noBorder{
    border:none;
  }
</style>
```

2.3.4 Seitennavigation

Mit CSS kann man text-basierte und ansprechende Navigationselemente entwickeln, die grafischen Lösungen in vielerlei Hinsicht überlegen sind. In diesem Abschnitt werden horizontale und vertikale Linklisten betrachtet, die nicht geschachtelt sind. Ausgangssituation ist eine Auswahlliste, deren Listenelemente mit dem Anchor-Tag als sensitive Flächen ausgezeichnet sind. Hier mal ein einfaches Beispiel als HTML-Code und die Bildschirmdarstellung ohne CSS. Für dieses Beispiel werden wir die Stilvorlage für ein vertikales und horizontales Menü entwickeln.

```
<nav>
  <ul>
    <li><a href="#">Einführung</a></li>
    <li><a href="#">Bericht</a></li>
    <li><a href="#">Kontakt</a></li>
    <li><a href="#">Impressum</a></li>
    <li><a href="#"> </a></li>
    <li><a href="#">Home</a></li>
 </ul>
</nav>
```

2.18: Navigationsmenü ohne Stilvorlage

Sieht doch sehr spartanisch aus, unser Navigationsmenü mit den Standardeinstellungen des Browsers. Zunächst begrenzen wir die Breite des Menüs auf 192px. Danach folgt eine Gestaltung der unsortierten Auswahlliste durch Ausschalten des Liststyles und die Randabstände werden auf null gesetzt.

```
nav{
  width:192px;
}
nav ul {
  list-style:none;
  margin:0;
  padding:0;
}
```

Danach werden die Links formatiert, `link` und `visited` werden gleich behandelt. Die Inline-Eigenschaft des Elements `<a>` wird zum Blockelement umgebaut. Damit erzielen wir eine gleiche Längenausdehnung für alle Elemente. Der Text wird entsprechend ausgezeichnet. Der untere Rand der Listenelemente wird noch durch eine dünne weiße Linie markiert.

```
nav li a:link, nav li a:visited {
  display:block;
  background-color:teal;
  color:white;
  font-size:100%;
  font-weight:bold;
  text-decoration:none;
  padding: 3px;
}
nav li{
  border-bottom: 1px solid white;
}
```

Jetzt muss noch die Eigenschaft bei Mausberührung für `:hover` definiert werden, der sog. Rollover–Effekt. Es wird einfach Hintergrund- gegen die Vordergrundfarbe getauscht und der rechten Rand verstärkt dargestellt. Abbildung 2.18 zeigt die Veränderung - erzielt durch unsere CSS-Formatierung gegenüber den Standardvorgaben.

```
nav li a:hover {
  background-color:white;
  color:teal;
```

```
  border-right: 12px solid teal;
}
```

Im nächsten Schritt soll das Ausgangsmenü der Abbildung 2.17 als ein horizontales Menü zum Einbau in eine Kopfzeile umgearbeitet werden. Dazu muss die Display-Eigenschaft der Listenelemente auf *inline* gesetzt werden. Der Zeilenumbruch wird so vermieden. Schriftgröße und Farbgestaltung wird weniger auffällig ausfallen.

2.19: Navigationsmenü mit CSS Vorgaben

Bei gleichem HTML-Code wie oben wird die Stilvorlage wie folgt modifiziert:

```
<style type = "text/css">
  nav{
    width:800px;
  }
  nav ul {
    list-style:none;
    margin:0px;
    padding:0px;
  }
  nav li a:link, nav li a:visited {
    background-color:white;
    color:teal;
    font-size:75%;
    font-weight:normal;
    text-decoration:none;
    padding: 0px 3px 0px 3px;
    margin-left: 0px;
    margin-right:5px;
    border: 1px solid teal;
  }
  nav li{
    border: 1px solid white;
    display:inline;
  }
  nav li a:hover {
    background-color:teal;
    color:white;
    border: 1px solid teal;
  }
</style>
```

Der Navigationsbereich erstreckt sich hier über 800 Pixel, Farbwerte und Schriftgrößen werden entsprechend angepasst. Die Listenelemente erhalten einen Rand. Der äußere Abstand der Listenelemente wird durch `margin-left` und `margin-right` verändert. Dadurch wird ein größerer Zwischenraum erzielt.

Abbildung 2.20: Menü in horizontaler Anordnung

2.3.5 Tabellengestaltung

CSS bietet vielfältige Möglichkeiten umfangreiche Tabellen lesbar und anspruchsvoll zu gestalten. Wir kommen zurück auf die Bundesliga-Tabelle aus dem Abschnitt 2.1.3. Der HTML-Code bleibt völlig unverändert, es wird lediglich eine Stilvorlage eingebunden. Mit den Pseudoselektoren von CSS3 können wir auf die Angabe verschiedener Class-Attribute verzichten. Zunächst werden alle Ränder entfernt, und eine serifenfreie Schrift wird vorgeschrieben. Für die Bereiche `thead` und `tfoot` wird der Hintergrund grau eingefärbt, die Schriftfarbe wird weiß. Der Text wird zentriert. Das Problem liegt in der zweiten Zeile von `thead`. Hier ist die Textausrichtung nicht zufriedenstellend, vgl. Abbildung 2.20.

```
table {
  width: 512px;
  border-collapse: collapse;
  border-style:none;
  font-family: sans-serif;
  font-size: 100%;
}
td,th{
  border-style:none;
}
thead, tfoot {
  background-color: #888888;
  color: white;
  text-align: center;
```

Abbildung 2.21: Textausrichtung Tabellenkopf und Zeile 2

Die zweite Zeile des Tabellenkopfes soll weiterhin linksbündig ausgerichtet werden. Hinzu kommt eine verkleinerte Schriftgröße. Auch die Abstände nach unten und oben sollen unterschiedlich zur ersten Zeile sein. Wir nutzen den Nth-of-type-Selektor für die gerade Zeile im Tabellenkopf und deren Elemente `th`. Nth-of- type

sucht jedes Element eines Typs nach einer Vorgabe bzw. eines Schlüsselwortes. Wir benutzen hier *even* und gelangen dadurch in die zweite Zeile.

```
  thead tr:nth-of-type(even) th {
  font-size: 75%;
  font-weight: normal;
  text-align: left;
  padding-top: 0px;
}
```

Nun sollen die ersten drei Zeilen grün eingefärbt werden und die vierte Zeile soll einen gelben Hintergrund bekommen. Die Zeilen fünf und sechs werden grau hinterlegt. Die letzten beiden Zeilen werden rot eingefärbt die drittletzte Zeile wieder gelb. Die Erklärung zur Farbgebung ist in der Fußzeile der Tabelle nachzulesen.

```
tbody tr:nth-child(-n+3) {
  background-color: lime;
}
tbody tr:nth-child(+4) {
  background-color: yellow;
}
tbody tr:nth-child(+5){
  background-color: #dddddd;
}
tbody tr:nth-child(+6){
  background-color: #dddddd;
}
```

Der Selektor `nth-child` sucht vorwärts nach dem ersten Kindelement `tr` im Tabellenkörper. Wir erhöhen das Element um 3 und zählen durch das negative Vorzeichen vor dem n rückwärts, so können wir die ersten drei Zeilen grün einfärben. Wir suchen das vierte Element, färben die Zeile gelb ein und verfahren auf gleiche Weise mit der Zeile 5 und 6.

Die Elemente der drei letzten Zeilen suchen wir mit `nth-last-child`. Wir finden die Zeile 18 und zählen um 2 rückwärts (negatives Vorzeichen) für die Zuordnung orange. Verbleit noch die drittletzte Zeile. Wir finden das letzte Kindelement und erhöhen in der Suchrichtung um 3. Die weiteren Stilvorgaben legen Randabstand, Buchstabengröße und Textausrichtung fest.

```
tbody tr:nth-last-child(-n+2){
  background-color: orange;
}
tbody tr:nth-last-child(+3){
  background-color: yellow;
}
td {
  border-collapse: inherit;
  font-size: 90%;
  text-align: left;
  padding: 3px;
}
tfoot tr td{
  padding-top: 12px;
  text-align: center;
  font-size: 75%;
}
```

Die benutzten Pseudoselektoren `nth-of-type`, `nth-last-child` und `nth-first-child` suchen in einer bestimmten Richtung nach einem Muster in den HTML-Elementen. In Abhängigkeit von der Suchrichtung werden Bereiche oder einzelne Zeilen durch eine Formel angegeben, auf die die Stilvereinbarung angewandt wird.

1. Fussball Bundesliga Saison 2011/2012 21. Spieltag

	Verein	Sp	G	U	N	Tore	+/-	P
1	Borussia Dortmund	21	14	4	3	46:14	32	46
2	Bayern München	21	14	2	5	49:14	35	44
3	Bor. Mönchengladbach	21	13	4	4	34:12	22	43
4	FC Schalke 04	21	13	2	6	46:28	18	41
5	Werder Bremen	21	9	6	6	34:35	-1	33
6	Bayer Leverkusen	21	8	7	6	28:28	0	31
7	Hannover 96	21	7	10	4	23:25	-2	31
8	VfL Wolfsburg	21	8	3	10	27:38	-11	27
9	VfB Stuttgart	21	7	5	9	31:28	3	26
10	Hamburger SV	21	6	8	7	26:34	-8	26
11	1899 Hoffenheim	21	6	7	8	23:25	-2	25
12	1. FC Köln	21	7	3	11	29:41	-12	24
13	1. FSV Mainz 05	21	5	8	8	29:35	-6	23
14	1. FC Nürnberg	21	6	4	11	19:31	-12	22
15	Hertha BSC	21	4	8	9	25:36	-11	20
16	1. FC Kaiserslautern	21	3	9	9	15:26	-11	18
17	FC Augsburg	21	3	9	9	19:33	-14	18
18	SC Freiburg	21	4	5	12	27:47	-20	17

1-3 Champions League, 4 Champions League Quali.,5-6 Europa League,16 Relegation,17-18 Abstieg

Abbildung 2.22: Mit Pseudoselektoren gestaltete Tabelle

2.4 Positionierung mit CSS

HTML-Elemente können mit der Eigenschaft `position` auf einer Webseite positioniert werden. Das Verhalten der Positionierung richtet sich nach den Positionsarten *static, fixed, absolute, relative* und den Positionseigenschaften des Elternelements.

Mit *static* wird der normale Elementfluss beschrieben, *fixed* ist die absolute Positionierung am Browserfenster. Zu *absolute* und *relative* betrachten wir ein Beispiel. Es liegen zwei Div-Container t1 und t2 vor. In denen sich jeweils ein Textabschnitt befindet.

```
<div class="t1">
  <p> ... </p>
</div>
<div class="t2">
  <p></p>
</div>
```

Beide Container werden absolut positioniert. Der erste Container 32px vom oberen Rand des Browserfensters und 64px vom linken Rand. Der zweite Container befindet sich mit der oberen linken Ecke in einem Abstand von 128px zum Browserfenster.

```
*{margin:0px;padding:0px;}
t1 {
  position:absolute;
  top: 32px;
  left:64px;
}
t2{
  position:absolute;
  top:128px;
  left:128px;
}
```

Attribute für die Positionsangaben sind *top, bottom, left, right*. Die Einheiten können in festen Pixelwerten oder relativ in Prozent oder Em angegeben werden. Elemente, die *absolute* oder *fixed* positioniert werden, sind vom normalen Formatierungsmodus des Dokuments ausgenommen. Im obigen Fall wird das Ergebnis bei absoluter und relativer Positionierung gleich sein. Die absolute Positionierung kann sich aber auch relativ auf die Ränder von Elternelementen beziehen. Wir fügen die Container t1 und t2 mit der absoluten Positionierung in einen leeren Container ein, der selbst mit einem Randabstand von 64px oben und links positioniert ist.

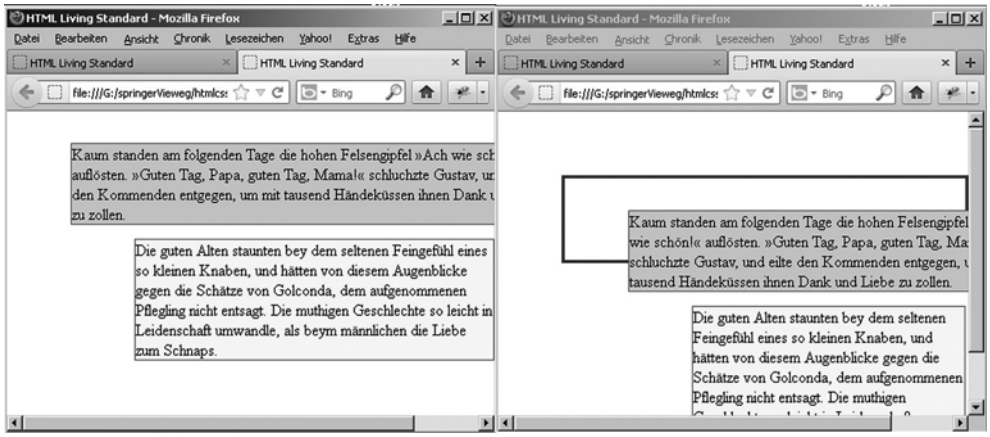

Abbildung 2.23: Positionierung von HTML-Elementen

Ohne umgebenden Container ist das Ergebnis mit absoluter und relativer Positionierung gleich, vgl. Abb. 2.22 links. Mit umgebendem Container positionieren wir t1 relativ und t2 absolut. Wir stellen fest, dass `absolute` sich ebenfalls wie `relative` zu dem Wrapper-Container verhält, vgl. Abb. 2.22 rechts.

Bei den folgenden Stilvorlagen zur Seitengestaltung werden wir zur Positionierung weniger auf die Eigenschaft `position` zurückgreifen, dafür mehr die Möglichkeiten von `margin` nutzen.

2.4.1 Einspaltiges Seitenlayout

Zu Beginn der Entwicklung von Standardseitenlayouts betrachten wir zunächst ein einspaltiges Dokument fester Größe mit einem Seitenheader, einem horizontalen Navigationsmenü, einem Inhaltsbereich, der die Artikel aufnimmt, und einer Fußzeile. Im ersten Schritt positionieren wir die Abschnitte um dann später die Details der Formatierung zu diskutieren. Der Body-Bereich im HTML-Dokument ist wie folgt codiert:

```
<body>
  <header class="seitenkopf">
    <h1>Header mit Background-Image</h1>
  </header>
  <nav>.. hier wird die horizontale Seiten-Navigation eingefügt</nav>
  <section class="inhalt">
    <!-- Der Inhaltsbereich -->
    <article>
      <header>Artikel-Überschrift</header>
      <p>Textabsatz</p>
      <img ...Abbildungen im Text />
      <footer>Fußzeile im Artikel</footer>
    </article>
    <article> ...weiter Artikel </article>
  </section>
  <footer>Seiten-Fußzeile</footer>
 </body>
```

Das gerenderte Ergebnis einer exemplarischen Seite mit allen CSS-Anweisungen finden Sie in der Abbildung 2.23. Betrachten wir nun die Anordnung der Seitenbereiche. Die Seite hat eine feste Breite von 640px, der Seitenbereich wird im Browserfenster zentriert (Wert *auto* für `margin-left` und `margin-right`), Farben und Schriftart werden für den Body festgelegt. Der Seitenheader bekommt eine feste Höhe von 96px, die Navigationsleiste 32px.

```
/* Zurücksetzen aller Randabstände */
*{
  margin:0px;
  padding:0px;}
body {
  margin-left:auto;
  margin-right:auto;
  width:640px;
  background-color:white;
  font-family: sans-serif;
  color:#4e3f31;}
```

Der äußere Abstand (`margin`) für den Seitenheader beträgt 3px. Der gleiche Wert wird für den inneren Abstand (`padding`) vereinbart. Hinzu kommt 1px für den Rand. Somit verringert sich die Größe des Header-Bereichs auf 626px. Die Gesamthöhe des Headers ist 96px. Das Hintergrundbild füllt den gesamten inneren Bereich bis zum Rand aus, hat also eine Größe von 632x88px. Den Header versehen wir mit dem Class-Attribut und vergeben den Bezeichner *seitenkopf*. Innerhalb des Headers ist noch eine Titelzeile vorgesehen.

```
header.seitenkopf {
  top:0px;
  height:82px;
```

```
  width:626px;
  background-color:white;
  margin:3px;
  padding:3px;
  border:1px solid #4e3f31;
  background-image: url(img/kurparkOeynhausen.jpg);
  background-repeat:no-repeat;}
```

Unmittelbar an den Header wird die randlose Navigationsleiste angefügt. Die Angaben zu `margin-top` und `margin–left` beziehen sich auf die Nachbarelemente Header und Body.

```
nav {
  margin-top:0px;
  margin-left:3px;
  height:32px;
  background-color:#4e3f31;
  width:634px;
}
```

Der folgende Inhaltsbereich, ebenfalls randlos, nimmt später die zu umrandenden Artikel auf. Das Class-Attribut wird mit dem Wert *inhalt* belegt. Daher notieren wir wie oben beim Nav-Element ebenfalls mit dem Class-Selektor.

```
section.inhalt{
  margin:0px;
  background-color:white;
  width:634px;}
```

Bevor wir zum inneren Bereich des Seiteninhalts kommen, zunächst der Blick auf den Seitenfooter: Dieser schließt an den Seiteninhalt an, zeigt aber eine verkleinerte Schriftgröße. Der Text wird zentriert und der Rand wird nur an der oberen Seite ausgeführt, vgl. Abbildung 2.23.

```
footer{
  margin-bottom:24px;
  margin-top:3px;
  margin-left:3px;
  padding:3px;
  font-size:75%;
  background-color:white;
  text-align:center;
  width:628px;
  border-top: 1px solid #4e3f31;}
```

Somit können wir den Blick auf das Innere des Seiteninhalts, die Artikel richten. Die Artikel haben einen eigenen Header- und Footer-Bereich. Innerhalb des Artikels sind Textabschnitte und Abbildungen vorgesehen. Der Class-Selektor wird gefolgt von einer Liste mit Nachfahren, getrennt durch ein Leerzeichen. Randabstände und Rand reduzieren die Größe des Bereichs auf 626px. Den Header versehen wir links mit einen kräftigen Rand von 24px Breite. Der Textabschnitt wird als Blocksatz (`text-align`) mit Einzug der ersten Zeile (`text-indent`) formatiert.

```
inhalt article{
  margin:3px;
  padding:3px;
  width:626px;
  background-color:white;
  border:1px solid #4e3f31;}
```

```
. inhalt article header {
  margin: 6px 0px 0px 6px;
  padding:12px;
  border: 0px 0px 0px 12px;
  border-left: 24px solid #4e3f31;
  font-weight:bold;}
inhalt article p {
  margin:6px;
  text-indent:12px;
  text-align:justify;}
```

Es folgt noch die Stilvorgabe für den Image-Tag. Mit dem Attribut `float:left` steht das Bild links und wird vom umgebenden Text umflutet. Auf der rechten Seite des Bildes benötigen wir etwas mehr äußeren Randabstand. Mit `padding` und `border` wird die Umrandung des Bildes erzielt. Der Artikel-Footer wird ohne Rand links mit 75% Schriftgröße positioniert.

```
inhalt article img {
  float:left;
  margin: 0px 6px 0px 6px;
  padding:2px;
  border:1px solid #4e3f31;}
inhalt article footer {
  margin:6px;
  border:none;
  text-align:left;font-size:75%;
  width:75%;margin-top:12px;}
```

Somit können wir uns der horizontalen Seitennavigation zuwenden, die als unsortierte Liste mit Auszeichnung der Listenelemente durch den Anchor-Tag realisiert ist.

```
<nav>
  <ul>
    <li><a href="#">Seite 1</a></li>
    <li><a href="#">Seite 2</a></li>
    <li><a href="#">Seite 3</a></li>
    <li><a href="#">Seite 4</a></li>
    <li><a href="#">Seite 5</a></li>
    <li><a href="#">Seite 6</a></li>
  </ul>
</nav>
```

Zunächst schalten wir den List-Style der unsortierten Liste aus. Den Anchor-Tag in den List-Elementen definieren wir als Block-Element mit einer festen Größe von 640/6px. Die Höhen des Navigationsbereichs, der Blockelemente und Textzeilen sind gleich. Durch die Display-Eigenschaft *block* wird der gesamte Hintergrund für den Link verfügbar. Der Text wird ausgezeichnet und durch die Eigenschaft `float` wird der Zeilenumbruch vermieden. Für das Pseudoelement `hover` werden die Hintergrundfarbe und die Textfarbe geändert. Zuletzt wird noch eine `id="home"` eingeführt, die eine Farbmarkierung für die aktuelle Seite vorsieht.

Abbildung 2.24: Einspaltiges Seitenlayout mit CSS

```
nav ul li{
  list-style:none;}
nav ul li a{
  display:block;
  width:105px;
  color:white;
  height32px;
  line-height:32px;
  text-decoration:none;
  text-align:center;
  text-transform:uppercase;
  font-size:75%;
  font-weight:bold;
  float:left;}
nav ul li a:hover {
  background-color:orange;
  color:#4e3f31;
}
a#home{
  color:orange;
}
```

Mit diesem einspaltigen Layout, haben wir ein erstes brauchbares Template erstellt. Modifikation von Farben und Schriftgestaltung sind in der Stilvorlage mit wenigen Mausklicks vorzunehmen. Sofern der Wunsch besteht, die Fußzeile im-

mer am unteren Bildschirmrand zu halten, wird auf eine geringfügige Modifikation verwiesen, die auf der Website zum Buch dokumentiert ist.

2.4.2 Zweispaltiges Seitenlayout

Die im vorhergehenden Abschnitt benutzte Website soll jetzt als zweispaltiges Layout ausgewiesen werden. Der Navigationsbereich `nav` wird danach mit vertikalem Menü im linken Bereich erscheinen, vgl. Abbildung 2.18. Der Inhaltsbereich rechts neben dem Menü soll sich dynamisch verhalten, d.h. dem Browserfenster anpassen. Lediglich das Menü auf der linken Seite soll in fester Größe bestehen bleiben. Hier noch einmal die Strukturelemente in HTML:

```
<body>
  <header class="seitenkopf></header>
  <nav> ... </nav>
  <section class="inhalt">
     <article>
       <header></header>
       <!-- Der Inhaltsbereich -->
       <footer id=”footerArticle”> </footer>
     </article>
   </section>
  <footer></footer>
 </body>
```

Im Body und den weiteren Elementen der rechten Seite wird die Eigenschaft `width` entfernt. Der Navigationsbereich erhält die Eigenschaft `float:left` mit einer festen Breite von 128px. Für den Seiteninhalt werden keine Breiten festgelegt. Der Abstand zum Rand wird mit `margin-left` und `margin-right` bestimmt. Im Navigationsbereich wird die Eigenschaft `float` für die Anchor-Tags entfernt. Das Hintergrundbild im Seitenkopf wird bei Bedarf wiederholt.

```
header.seitenkopf {
  top:0px;
  height:82px;
  margin:3px;
  padding:3px;
  border:1px solid #4e3f31;
  background-image: url(img/kurparkOeynhausen.jpg);
  background-repeat:repeat-x;}
nav {
  float:left;
  margin:3px; padding:0px;
  background-color:#4e3f31;
  width:128px;}
section.inhalt{
  margin-left:138px;
  margin-right:0px;
  margin-top:7px;
}
```

Mit wenigen Modifikationen der Stilvorlage kommen wir so zum zweispaltigen Standardlayout der Abbildung 2.25.

Abbildung 2.25: Zweispaltiges Layout mit CSS

2.4.3 Dreispaltiges dynamisches Seitenlayout

Ein weiteres Standard-Seitenlayout wird im Folgenden in den Grundzügen erläutert. Ein vollständig ausgearbeitetes Template mit der Stilvorlage finden Sie auf der Website *www.seiten-programmierung.de*. Das Layout ist dreispaltig und erstreckt sich über das gesamte Browserfenster. Der rechte und linke Bereich ist mit fester Größe angegeben. Der mittlere Bereich passt sich dem verbleibenden Platz an. Eingeleitet wird die Seite mit einem Header und mit einem Footer abgeschlossen. Der Footer erstreckt sich über die gesamte Fensterbreite unter der längsten Spalte. Aus diesem Grund sind die Inhaltsbereiche in einem Container abgelegt, der dem Header folgt. Darauf folgt wiederum ein Container mit dem ausschließlichen Zweck zur Aufhebung der Float-Eigenschaften. Der HTML-Code wird wie folgt notiert:

```
<div id="wrapper">
  <nav></nav>
  <aside></aside>
  <section></section>
</div>
<div id="clear"></div>
  <footer></footer>
</body>
```

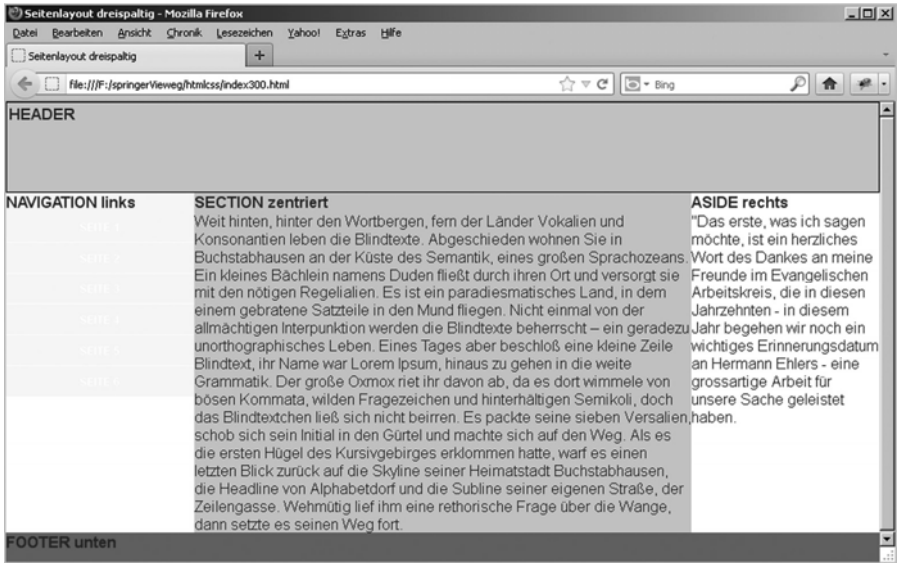

Abbildung 2.26: Dreispaltiges dynamisches Layout

Die Stilvorlage beginnt wie üblich mit dem Zurücksetzen der Randbereiche, der globalen Textdarstellung im Body und einem absolut positioniertem Header. Der Navigationsbereich links wird gefloatet, der Seitenbereich rechts ebenfalls. Die Breite beider Bereiche wird auf 196px gesetzt.

```
*{
  margin:0px;
  padding:0px;}
body {
  background-color:white;
  font-family: sans-serif;
  color:#404040;}
header{
  top:0px; left:0px;
  height:90px;
  background-color: lightgray;
  padding:2px;
  border: 1px solid black;}
nav{
  float:left;
  background-color:yellow;
  width:196px;}
aside{
  float:right;
  width:196px;}
```

Es folgt die Vereinbarung für den Inhaltsbereich mit festen Seitenabständen rechts und links entsprechend der Weite von `nav` und `aside`. Der Footer wird gestaltet und für die Identität *clear* das Floating zurückgesetzt.

```
section{
  margin-left:196px;
  margin-right:196px;
  background-color:lightgray;
}
footer {
```

```
  background-color:red;
  height:32px;}
#clear{clear:both;}
```

Soweit sollte der Exkurs über CSS für den Einsteiger hinreichend sein. Das Thema CSS wurde in diesem Kapitel auf ca. 30 Seiten abgehandelt. Die ausführliche Literatur wie etwa *Designing Without Tables* von Dan Shafer kommt mit 500 Seiten daher. Aufgrund unserer Komprimierung bleiben natürlich etliche Details auf der Strecke. Sie haben aber die Grundlage für eine korrekte Gestaltung von Webseiten geschaffen und können die vorgestellten Templates für anstehende Projekte nutzen und modifizieren. Der eigene individuelle Webauftritt ist vorbereitet: Fügen Sie Ihre Inhalte in die Bereiche ein und verändern Sie Farb- und Textgestaltung nach eigenen Vorstellungen.

2.5 Vor dem Upload

Selbstverständlich haben Sie Ihre Webseiten mit den gängigen Browsern und deren Webentwickler-Tools oder Firebug getestet. Wir programmierten für moderne Browser den kommenden Web-Standard. Auf die Kompatibilität zu veralteten Systemen haben wir verzichtet. Ein Qualitätssiegel auf der Webpräsenz mit der Konformität zu HTML und CSS würde in jedem Fall vorteilhaft sein. Vor dem Upload kommt das Testen der Seiten, die Validierung, das Einfügen von Meta-Tags und die Erstellung eisnes Impressums.

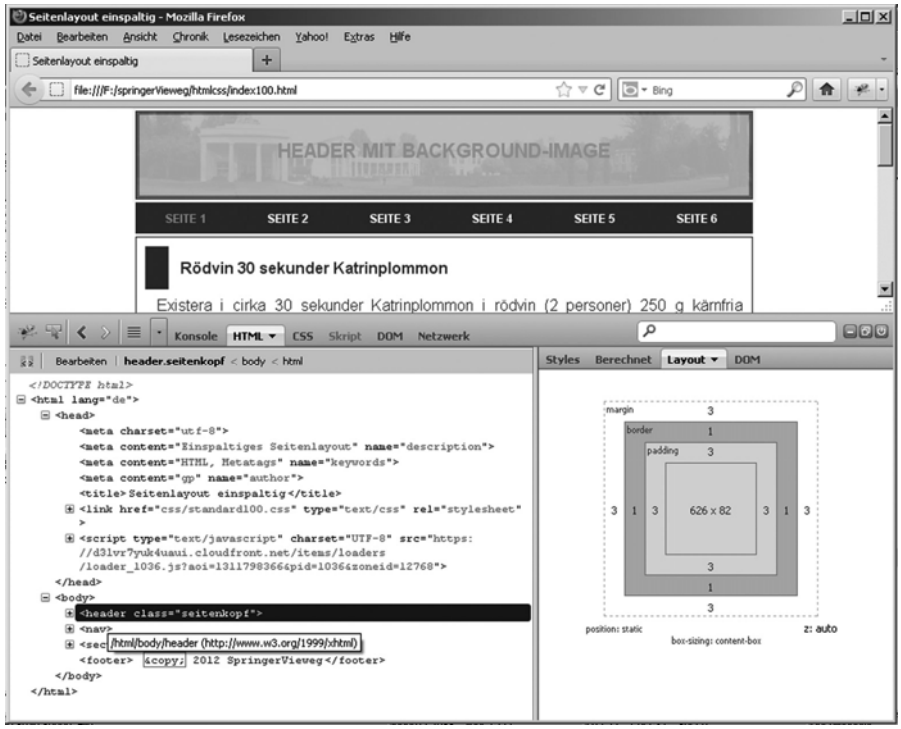

Abbildung 2.27: Firebug zeigt den HTML-Elementenbaum und die Eigenschaften

Firebug wurde bereits 2006 eingeführt. Später kamen Web-Inspector für Safari und Google Chrome oder Dragonfly für Opera hinzu. Auch der Internet Explorer besitzt mittlerweile Entwicklertools. Ein kurzer Blick auf Firebug zeigt im oberen Fenster die Webseite, im unteren Fenster der Abbildung 2.27 sehen wir links den HTML-Code. Bei Doppelklick auf ein Element werden im rechten Fenster entsprechend der Einstellung dessen Eigenschaften angezeigt. In der Abbildung ist es die Box mit den Randeinstellungen für den Header, der im oberen Fenster markiert ist.

Das W3C bietet unter *validator.w3c.org* einen Markup Validation Service an. Sie können für ein Dokument die URI angeben, das Dokument hochladen oder auch direkt eingeben. Das Validierungsergebnis erhalten Sie umgehend angezeigt. Unser HTML-Code wird als HTML5 validiert. Der CSS-Validierungsservice ist unter *http://jigsaw.w3.org/css-validator/* aufzurufen. Die Webadresse ist auf der Protokollseite der Markup-Validation auch verlinkt. Nach erfolgreicher Validierung können Sie auch das Gütesiegel mit Ihrer Seite verlinken. Die Codezeilen werden auf der Ergebnisseite zum Copy-Paste angeboten, vgl. Abb. 2.28.

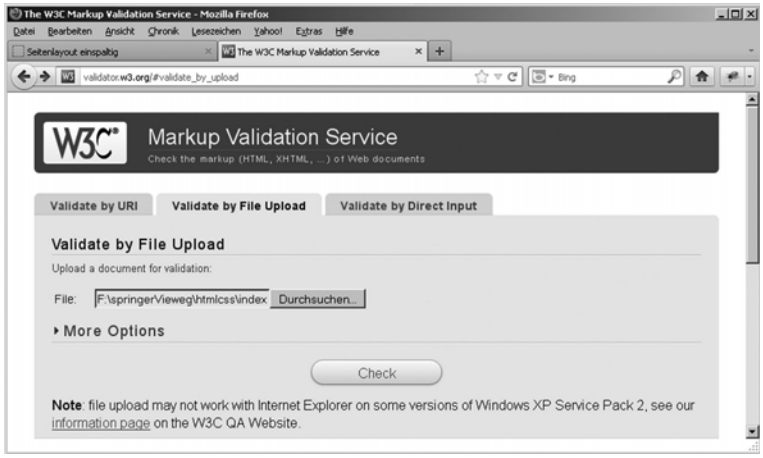

Abbildung 2.28: Markup Validation Service

Metadaten sind im Allgemeinen Daten über Daten. In den Meta-Elementen oder auch Meta-Tags des Headers einer HTML-Seite werden Daten aufgenommen, die nicht vom Browser gerendert werden. Die Daten füttern Suchmaschinen und Robots und dienen auch der SEO (Search Engine Optimization). Von den etwa 30 gängigen Meta-Tags spielen Description und Keywords eine Rolle. Auf diese Daten sollten Sie nicht verzichten. Auch die Angaben über Autor und Erstellung sind nützlich. Einen kompletten Satz Meta-Tags können Sie mit Online-Diensten generieren. Bei diesen Anbietern können Sie sich auch bei Suchmaschinen anmelden. Wenn Sie von einem derartigen Service Gebrauch machen wollen, googeln Sie Metatag-Generator und klicken Sie bei den Ergebnissen auf den Anbieter Ihrer Wahl.

Zu jeder Website gehören ein Impressum und Angaben über den Haftungsausschluss. Auch hier können Sie sich eines kostenpflichtigen oder -freien Dienstes im Internet bedienen. Das Impressum der Webseite zum Buch wurde mit dem Generator von *www.e-recht24.de* erstellt.

Abbildung 2.29: CSS-Validation mit Code für das Gütesiegel

Soweit sind die Vorbereitungen zum Upload der Website abgeschlossen. Ihre Site soll auf einem Server gehostet werden, dessen Betriebssystem Sie möglicherweise noch nicht kennen. Eine klare Verzeichnisstruktur, die auch für eine mehrsprachige Präsenz vorzusehen ist, kann wie folgt projektorientiert gestaltet sein:

```
Auflistung der Ordnerpfade
C:.
└───projekt
    ├───css
    ├───de
    │   ├───seite1
    │   ├───seite2
    │   └───seite3
    ├───gb
    ├───img
    └───javascript
```

Für CSS, HTML , Javascript und Grafiken existieren eigene Verzeichnisse. Im Root-Verzeichnis ist *index.html* oder *index.php* die Homepage. Die Dokumente sind in den landessprachlichen Verzeichnissen *de* oder *gb* für die englische Sprachversion abgelegt. Jede Seite hat ein eigenes Unterverzeichnis, dort auch wieder mit *index.html* oder *index.php* die Seite, die der Browser als Voreinstellung übernimmt. Wählen Sie für Dateinamen immer Kleinschreibung, benutzen Sie keine Sonderzeichen in den Dateinamen und verwenden Sie auch nicht zu lange Bezeichner. Die alte *acht-Punkt-drei* Konvention (acht Zeichen für den Namen, drei für die Extension, getrennt durch Punkt) hat sich bei einer übersichtlichen Verzeichnisstruktur immer noch bewährt. Geben Sie alle Referenzen in Ihren Dokumenten als relative

Pfadangaben an. Sie gehen immer von dem Ort der Ursprungsdatei aus und verfolgen den Pfad bis zur Referenzdatei mit dem Schrägstrich als Trennung. Mit `../` gelangen Sie in der Verzeichnishierarchie eine Ebene höher. Wenn Sie von einem Dokument innerhalb *de/seite3* auf eine Grafik im Verzeichnis *img* verweisen wollen, dann benötigen Sie *href="../../img/meineGrafik.jpg"*.

Zum Hochladen der Seiten wollen wir Filezilla benutzen. Filezilla ist ein Open-Source FTP-Client, die deutsche Referenz ist *www.filezilla.de*. Das Programm hat eine recht übersichtliche Fensterstruktur. Adresse, Benutzer, Passwort und Port sind Ihre persönlichen Daten zum Einloggen in die Web-Präsenz bei Ihrem Provider. Das obere Fenster ist ein Statusfenster mit Nachrichten für Übertragungsbefehle. Links haben Sie den lokalen Bereich und rechts den serverseitigen Bereich. Dort können Sie auch die Benutzerberechtigungen für die Dateien eintragen.

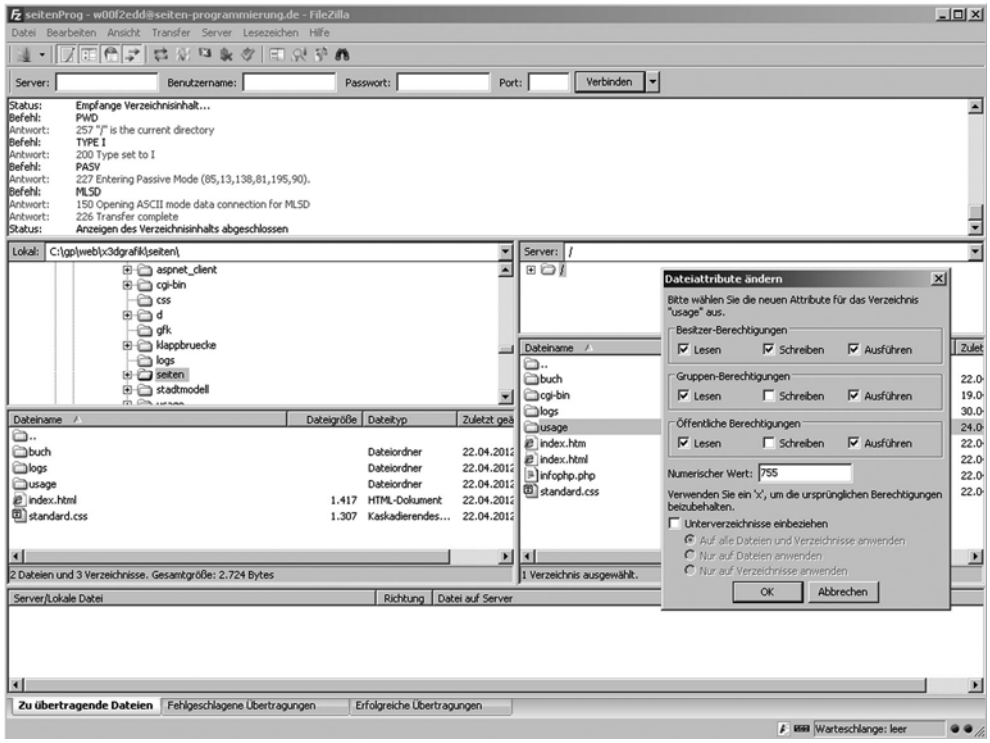

Abbildung 2.30: FTP-Client Filezilla mit Kontextmenü Dateiberechtigungen

Sie markieren entweder serverseitig oder clientseitig die Dateien, bei Klick auf die rechte Maustaste erhalten Sie im Kontextmenü `Download` oder `Upload` angeboten. Mehrere ausgewählte Dateien erscheinen im unteren Fensterbereich als Warteschlange. Dort wird Ihnen auch der Status des Transfervorgangs angezeigt. Nach Abschluss eines Transfers sind Entwicklungsrechner und Server synchronisiert.

Jetzt ist es geschafft, Sie sind mit einer eigenen Internetpräsenz im Web. Rufen Sie Ihre eigene Seite im Browser zum abschließenden Test auf. Die Anforderung wird

vom Web-Server bedient, Ihr Browser rendert hoffentlich in der von Ihnen gewünschten Weise. Aber Vorsicht, löschen Sie Ihren Browser-Cache, sonst bekommen Sie ggf. Daten angezeigt, die sich noch im Zwischenspeicher befinden und evtl. nicht mit der Präsenz auf dem Web-Server übereinstimmen. Bereits geladene Objekte speichert der Browser und holt diese bei erneuter Anzeige nicht unbedingt vom Web-Server. Es gibt verschiedene Methoden der Benutzerkontrolle über den Cache. Sie können die Größe auf null setzen oder das Alter der Ressourcen begrenzen. Beim Firefox-Browser wählen Sie Extras > Einstellungen > Erweitert > Netwerk und können dort die Cache-Behandlung manipulieren.

2.6 Zusammenfassung HTML und CSS

Nach dem Durcharbeiten dieses Kapitels haben Sie die Grundlagen zur Publikation statischer Webseiten gelegt. Sie können validierte HTML-Seiten mit CSS-Stilvorlagen gestalten.

Eine HTML-Seite ist in die Bereiche `<head>` und `<body>` eingeteilt. Es gibt für jedes Element einen Start-Tag und ein Ende-Tag. Die Elemente werden mit Attributen versehen, denen Werte zuzuweisen sind. Groß- und Kleinschreibung werden nicht unterschieden. Spitze Klammern und doppelte Hochkommata sind reservierten Zeichen. Wollen Sie diese in Ihrem Text verwenden, benötigen Sie eine Codierung.

CSS-Selektoren sind Muster, die nach HTML-Elementen suchen und die Gestaltungsregeln auf die gefundenen Elemente anwenden. Die Regeln eines Selektors enthalten in geschweiften Klammern die Attribute mit Doppelpunkt getrennt vom Wert, der mit einem Semikolon abgeschlossen wird. Gleiche Elemente werden unterschiedlich durch Einsatz von Klassen dargestellt. Die kaskadierende Stilvorlage kann sich irgendwo im Internet befinden. Individuelle Eintragungen überschreiben die globalen.

Zum Einstieg in die Syntax der Sprachen wird ein wenig Übung mit einem Texteditor empfohlen. Danach können Sie gängige Werkzeuge wie HTML-Editor, Stylesheet-Generator, FTP-Client und Meta-Tag-Generator einsetzen. Ein Bildbearbeitungsprogramm zur Aufbereitung von Grafiken vervollständigt Ihren Werkzeugkasten. Gestalten Sie einfache Seiten, vermeiden Sie den Multimedia-Overkill.

Etliche nützliche Webreferenzen wurden vorgestellt, einige Tipps und Tricks wurden angewandt. Lassen Sie Ihre Seiten auf Konformität zu den W3C-Standards überprüfen. Mit der gelungenen Validierung können die Seiten den Web-Crawlern bedenkenlos zur Verfügung gestellt werden.

2.7 Weiterführende Literatur zu Kapitel HTML und CSS

[2.1] Peter Kröner: HTML5, Open Source Press, München, 2010

[2.2] Brian P.Hogan: HTML5 UND CSS3; O'Reilly, 2011

[2.3] Rachel Andrew: DER CSS PROBLEMLÖSER, dPunktVerlag, 2008

[2.4] Dan Shafer: DESIGNING WITHOUT TABLES. Using CSS, 2003 VIC Australia

Benutzte Internetquellen sind innerhalb des Textes angegeben.

http://www.springer.com/978-3-8348-2484-4

Webseiten-Programmierung
Sprachen, Werkzeuge, Entwicklung
Pomaska, G.
2012, XII, 255 S. 100 Abb., Softcover
ISBN: 978-3-8348-2484-4