

# Homepages mit HTML und CSS

**Lizensierte  
PDF-Ausgabe**

**Autor: Johann-Christian Hanke**

**[www.KnowWare.de](http://www.KnowWare.de)**



## **In eigener Sache: Copyright**

Wir freuen uns, dass Sie sich für eine PDF-Ausgabe unserer KnowWare-Computerbücher entschieden haben!

Damit wir Ihnen auch zukünftig immer wieder interessante und aktuelle Titel anbieten können, müssen wir natürlich dafür Sorge tragen, dass unsere Autoren entsprechend vergütet werden. Denn bis eine KnowWare-Ausgabe fertig ist, vergehen einige Wochen Arbeitszeit.

Aus diesem Grunde haben wir uns entschlossen, unsere PDF-Ausgaben nach dem Kauf über unseren Bestellshop bzw. per Direktbestellung bei unserer Bestellannahme (und nur hier sind sie erhältlich) mit einer auf den Käufer personalisierten, aber versteckten Codierung zu versehen.

So haben wir die Möglichkeit, bei einem Verdacht der Verbreitung von Raubkopien, den ursprünglichen Bezieher des kopierten Textes zu ermitteln.

Natürlich dürfen Sie die PDF-Ausgabe für Ihren eigenen Bedarf und Ihre eigenen Geräte kopieren, damit Sie die Ausgaben sowohl auf Ihrem Tablet, als auch auf Ihrem PC und Notebook nutzen können.

Wir hoffen, dass Sie viel Freude und Erfolg mit der vorliegenden Ausgabe haben!

Ihr KnowWare-Team

---

# Homepages mit HTML und CSS

## 6. komplett neubearbeitete Auflage

Autor: Johann-Christian Hanke

ISBN: 978-87-90785-79-6, 6. Neub. Auflage: 2007-07, korr. Nachdruck 2008-09

© Copyright 2006-2007 by KnowWare Vertrieb GmbH, [verlag@knowwware.de](mailto:verlag@knowwware.de)

Cover: Sylvio Droigk, Lektorat: Karl Antz, Schlussredaktion: Manfred Buchholz

Published by KnowWare Vertrieb GmbH

---

### Bestellung für Endverbraucher und für den Buchhandel

KnowWare-Vertrieb GmbH

Postfach 3920

D-49029 Osnabrück

Tel.: +49 (0)541 33145-20

Fax: +49 (0)541 33145-33

E-Mail: [bestellung@knowwware.de](mailto:bestellung@knowwware.de)

Web: [www.knowware.de](http://www.knowware.de)

### Worum es geht

Hinter KnowWare steht der Gedanke,  
Wissen leicht verständlich und preisgünstig  
zu vermitteln.

### Wo sind die Hefte erhältlich

In Buchhandlungen und im  
Bahnhofsbuchhandel sowie über  
verschiedene Internetshops.

Alle beim Verlag vorrätigen  
Titel kannst du immer bestellen.

### Bestellungen:

- am einfachsten über unsere Webseite  
[www.knowware.de](http://www.knowware.de)
- oder mit dem Bestellformular am Ende  
eines KnowWare Heftes
- oder per Fax, Telefon sowie Email  
Details siehe oben

### unter [www.knowware.de](http://www.knowware.de) findest du:

- Beschreibungen und Bilder aller  
Hefte.
- Bei unseren Heften stehen dir die  
ersten 15-20 Seiten pro Titel als  
kostenlose PDF-Datei zur  
Verfügung. So lässt sich jedes  
Heft online testen.
- Online-Bestellungen über den Web-  
Shop
- Kostenloser Newsletter mit vielen  
Vorteilen und Informationen
- Interne Suchfunktion nach  
Schlagworten. Du findest schnell,  
was du suchst.
- Online-Titel eine neue Form der  
Knowware Computerhefte. Diese  
Ausgaben sind nach dem Erwerb  
über unsere Webseite sofort  
verfügbar. Kein Papierdruck, kein  
Herunterladen – einfach und  
umweltfreundlich im Internet  
arbeiten.

## Inhaltsverzeichnis:

Homepage-Kurs für alle: Ein paar Hinweise zu Beginn .....	4	Zusatzwissen für Profis: Mehr zu Hintergrundeffekten.....	44
<b>Kapitel 1: HTML ist einfach! Dein Eigenheim im Web .....</b>	<b>5</b>	✓ Übungsteil D: Rahmen, Ränder, Effekte .....	46
Projektordner einrichten.....	5	<b>Kapitel 5: CSS professionell! DIV, SPAN, hover und Co. ...</b>	<b>47</b>
HTML, HEAD, TITLE und BODY: Grundlagen .....	6	Mehr Freiheit: <div> und <span> .....	47
Gar nicht so schwer: Grafik im GIF-Format! .....	8	Block- und Zeichenebene .....	47
Erlaubte Dateiformate .....	8	<b>Volle CSS-Power: Style Sheet-Bereich extern auslagern.....</b>	<b>48</b>
Dein erstes GIF-Bild mit Paint.....	8	Externe CSS-Datei erstellen .....	48
HTML pur: Das erste Projekt im Quelltext.....	9	<b>Quelltext der externen CSS-Datei im Überblick.....</b>	<b>49</b>
So sieht's im Browser aus.....	10	Link zur Style Sheet-Datei.....	49
Blick hinter die Kulissen: HTML-Tags im Detail .....	11	<b>So erzeugst du dynamische Querverweise mit hover .....</b>	<b>50</b>
Der Kopfbereich im Detail .....	11	Fünf Pseudo-Klassen für Links .....	50
Absätze und Zeilenumbruch .....	11	<b>Aufzählungspunkte mit Bildchen und Liste als Menü.....</b>	<b>51</b>
So fügst du eine Grafik ein: <img> und seine Attribute.....	12	Individuelles Aufzählungszeichen .....	51
Verknüpfen statt Speichern! .....	12	Auch das geht: Menüs als Liste .....	51
Grafik ausrichten mit align.....	12	<b>Profi-Wissen: Hinweise zu externen Style Sheets.....</b>	<b>52</b>
<b>Zusatzwissen für Profis: Grundbegriffe kurz erklärt .....</b>	<b>13</b>	✓ Übungsteil E: Übungen zu externen Style Sheets.....	53
✓ Übungsteil A: Erste Schritte mit HTML.....	14	<b>Kapitel 6: Das Projekt – Spaltensatz mit Kopf- und Fußzeile</b>	<b>54</b>
<b>Kapitel 2: Hyperlinks, die „Kreuz- und Querverweise“ .....</b>	<b>16</b>	Das Projekt „Ferienhaus“.....	54
Seiten verbinden: Interne Links .....	16	Das Layout im Überblick.....	55
Grundsyntax der Hyperlinks .....	16	Schritt für Schritt zum Erfolg .....	55
<b>Mehr Feedback durch den E-Mail-Link .....</b>	<b>18</b>	DIV-Container mit Eigenschaft id.....	56
Das Schlüsselwort mailto: .....	18	Die CSS-Datei „struktur.css“ .....	56
<b>Links nach draußen: Die externen Hyperlinks .....</b>	<b>19</b>	Der Trick mit der Hauptspalte .....	59
<b>Hyperlinks für Profis: Von Downloadlink bis Zielangabe ..</b>	<b>20</b>	<b>Kapitel 7: Exakt positioniert: Dreispalter mit fixierten DIVs</b>	<b>60</b>
Zu Download anbieten: Musik & PDF .....	21	CSS-P für exaktes Positionieren .....	60
✓ Übungsteil B: Interne und externe Hyperlinks .....	22	Eigenschaften und Werte von CSS-P .....	60
<b>Kapitel 3: Gestaltung mit Cascading Style Sheets .....</b>	<b>24</b>	Dreispalter mit Header .....	61
Kopie des Ordners eigenheim .....	24	CSS-P: Übersicht der Eigenschaften .....	61
Die Gesamt-Schriftart ändern.....	24	✓ Übungsteil F: Übungen zum Spaltensatz mit CSS.....	62
<b>Die wichtigsten Stil-Eigenschaften zur Schriftgestaltung.....</b>	<b>26</b>	<b>Kapitel 8: Die Feinheiten – Menü und Inhalt chic gestalten</b>	<b>63</b>
Schrifteigenschaften .....	26	CSS-Datei „layout.css“ .....	63
Ausrichtung und Zeilenabstand .....	26	Die Inhalte formatieren.....	64
Die Maßeinheiten von CSS .....	27	Gestaltung des main-Containers .....	66
<b>Vordergrund- und Hintergrundfarbe einstellen.....</b>	<b>28</b>	<b>Kapitel 9: Tabellen de luxe mit HTML und CSS .....</b>	<b>67</b>
Farben in HTML und CSS .....	28	Tabelle für Belegungsübersicht.....	67
Eigenschaft color: Farbe zuweisen .....	28	Tabelle mit CSS formatieren .....	69
<b>Wichtige Gesetzmäßigkeiten von Style Sheets .....</b>	<b>29</b>	Zusatzwissen zu Tabellen .....	69
<b>Style Sheets an zentraler Stelle im HEAD notieren.....</b>	<b>30</b>	✓ Übungsteil G: Übungen zu Tabellen und zum CSS-Layout.....	70
Style Sheets im HEAD notieren .....	30	<b>Kapitel 10: Bitte bestellen! Attraktive Formulare dank CSS</b>	<b>71</b>
<b>Zusatzwissen für Profis: Schriftgestaltung und Listen .....</b>	<b>33</b>	TEXTAREA: Mehr Feedback! .....	73
✓ Übungsteil C: Erste Schritte mit CSS.....	35	FIELDSET und LEGEND .....	74
<b>Kapitel 4: Rahmen, Ränder und Hintergrundeffekte.....</b>	<b>37</b>	Submit- und Reset-Button .....	74
Rahmen und Ränder .....	37	Mann oder Frau? Pull-down-Menü! .....	74
Umrandungsstile von CSS.....	37	Nur einer gewinnt: Radioknöpfe .....	74
<b>Flexible Gestaltung durch separate Stilklassen .....</b>	<b>38</b>	Bitte wählen Sie per Checkbox .....	74
Separate Stilklassen notieren .....	38	<b>Die optische Gestaltung des Formulars mit CSS .....</b>	<b>75</b>
Gebundene und freie Klassen .....	38	Der CSS-Code für das Formular .....	75
<b>Exaktes Layout: Ränder, Kasten und Innenrand.....</b>	<b>39</b>	✓ Übungsteil H: Übungen zu Formularen und zu CSS... 76	
Randeinstellungen mit margin .....	39	<b>Nützliche Tools, Links und Sites für HTML und CSS .....</b>	<b>77</b>
Mehr Luft bitte: Innenrand .....	40	HTML und CSS prüfen.....	77
<b>Absolut oder relativ? Breitenangaben justieren .....</b>	<b>41</b>	HTML- und CSS-Referenzen.....	77
Breite des Elements einstellen .....	41	Editoren.....	77
Individuelle Breiten mit width.....	41	Browser-Tools .....	77
<b>Alles im Rahmen: Hintergrundinfos zum Boxmodell .....</b>	<b>42</b>	<b>Stichwortverzeichnis.....</b>	<b>78</b>
Boxmodell-Fehler vom Explorer.....	42		

## Homepage-Kurs für alle: Ein paar Hinweise zu Beginn

### Schwerpunkt liegt auf CSS

Herzlich willkommen, liebe Leserin, lieber Leser! Vielen Dank für das Vertrauen in KnowWare und danke auch, dass du dich (wieder) für ein Homepage-Heft von mir entschieden hast.

In diesem Titel legen wir den Schwerpunkt vor allem auf CSS, die beliebte und flexible Gestaltungs- und Formatiersprache für Seiten im Web!

Ich zeige dir an Beispielen, wie du in Handarbeit schnell und einfach attraktive Webseiten erstellst und vor allem gestaltest. Und das nur mit CSS (Stilvorlagen), HTML und etwas Phantasie!

Falls du kein HTML kannst, ist das nicht schlimm. Du lernst es hier ganz nebenbei!

Ich will, dass du sofort Erfolgserlebnisse hast. Das bedeutet: So viel Praxis wie möglich, Theorie nur dann, wenn unbedingt nötig.

### Sehr wichtig: Windows-Kenntnisse

Für dieses Heft benötigst du gute Kenntnisse deines Betriebssystems – genauso wie für das Autofahren Kenntnisse in der StVO. Du arbeitest mit einer Version von Windows (95/98/Me/2000/XP/Vista)? Damit du nicht „gegen den Baum fährst“ setze ich voraus, dass du weißt, wie man

- den Windows-Explorer aufruft und
- mit dem Windows-Explorer Ordner, Unterordner und Dateien einrichtet und kopiert.

Außerdem erwarte ich, dass bei dir die in Windows von Hause aus leider abgeschalteten Dateiendungen eingeblendet sind!

Bei Wissenslücken bringt dich eins unserer preiswerten Grundlagenhefte zu Windows schnell auf Trab!

### Welche Programme brauchst du?

Windows genügt. Wir arbeiten anfangs mit der hier mitgelieferten Minimal-Textverarbeitung Editor (Notepad), später dann mit einem Editor wie Weaverslave ([www.weaverslave.ws](http://www.weaverslave.ws)). Als Browser verwende ich vorerst den Internet Explorer, weil dieser unter Windows automatisch als Standard-Browser gewählt wird. Dein Internet Explorer sollte am besten die Versionsnummer 6 oder 7 tragen, der „5er-Browser“ älterer Windows-Versionen dagegen interpretiert viele Maße (Breite, Höhe) nicht korrekt.

Spätestens wenn du fortgeschrittener bist, solltest du auf Firefox umsteigen ([www.firefox-browser.de](http://www.firefox-browser.de)). Ich persönlich ziehe den Firefox selbst dem Internet Explorer 7 vor. Nicht zuletzt wegen der tollen Webdeveloper Toolbar, siehe Seite 77. Google Chrome habe ich dagegen nicht in mein Herz geschlossen.

### Wie publiziere ich die Seiten?

Zum Publizieren deiner Seiten – was nicht Thema dieses Hefts ist – empfehle ich das Freeware-Programm FileZilla ([www.filezilla.de](http://www.filezilla.de)). Du kennst dich mit FileZilla noch nicht aus? Schau zur Rubrik „FTP-Schnelleinstieg“ auf [www.filezilla.de](http://www.filezilla.de). Dort findest du eine ausführliche Anleitung!

### Beispieldateien und Feedback

Alle Beispieldateien zu diesem Heft kannst du von [www.knowware.de/?book=css](http://www.knowware.de/?book=css) herunterladen! Sie liegen als Zip-Archiv vor. Achte darauf, das richtige Archiv für diese Auflage zu erwischen, da es im Vergleich zu den Voraufgaben viele Änderungen gibt!

### ■ Änderungen in der sechsten Auflage

Die Urausgabe dieses Heftes erschien Anfang 2002, also vor fünfeinhalb Jahren! Seitdem hat sich viel geändert – neue Browserversionen sind erschienen und es gibt neue Ansätze beim Webdesign. Das wirkt sich vor allem auf das „Ferienhaus-Beispiel“ aus. Die vorherige Version (casa-Projekt) arbeitet mit absoluter Positionierung. Damals war das ein machbarer und zuverlässiger Weg, um tabellenfreies Layout mit CSS zu realisieren. Heute ist dieser Ansatz überholt. Deshalb habe ich die entsprechenden Kapitel komplett neu geschrieben!

Aber auch in den anderen Anleitungen findest du neue Erkenntnisse – beispielsweise zum fehlerhaften Boxmodell des alten Internet Explorers (siehe Seite 42). Entfallen ist dafür das Frame-Kapitel, da Frames immer mehr an Bedeutung verlieren.

Kursleiter müssen nicht völlig umlernen. Die ersten zwei Drittel des Heftes (der Einstieg in CSS) haben sich bewährt und wurden nur leicht modernisiert.

Wenn du Interesse am „alten“ Framekapitel hast – du findest es als PDF bei den Downloaddateien!

Und nun viel Spaß und Erfolg mit HTML und CSS!

Johann-Christian Hanke

Berlin 2002, 2003, 2004, 2006, 2007 und 2008

## Kapitel 1: HTML ist einfach! Dein Eigenheim im Web

Auf los geht's los! Präsentieren wir uns im Web. Im ersten Projekt stellst du dich, deinen Beruf und deine Interessen auf einer Homepage vor. Füge außerdem gleich eine Grafik ein.



### Mit Foto, Farben und Rahmen: Selbstdarstellung.

Egal ob als Neueinstieg oder Wiederholung: Bei dieser Gelegenheit zeige ich dir die Grundbegriffe von HTML. Danach steigen wir sofort ein in die attraktive Gestaltung mit CSS! Nur Mut, du schaffst das auch!

### Projektordner einrichten

Zuerst richtest du direkt auf deiner Festplatte C: einen eigenen Projektordner ein namens `homepage`. Dort erstellst du für unser erstes Projekt einen weiteren Ordner namens `eigenheim`.

Der Ordner `eigenheim` ist dein Start-Verzeichnis für dieses Projekt, quasi der Stammordner deiner Webpräsenz. Man redet auch vom Root-Verzeichnis des Projekts (`root` = Wurzel).

Ich empfehle die generelle Kleinschreibung von Ordner- und Dateinamen. Auf gar keinen Fall solltest du Groß- und Kleinschreibung durcheinanderbringen, da Webserver im Gegensatz zu Windows zwischen Groß- und Kleinschreibung unterscheiden!

Und noch etwas: HTML-Dateien sind simple Textdateien mit der Endung `htm` bzw. `html`. Die Startdatei wird meist `index.html` genannt.

### Startdatei `index.html` erstellen

Nun folgt sinngemäß die gleiche Vorgehensweise wie in „Homepages für Einsteiger“. Auch hier fangen wir bescheiden mit einem Ordner an – steigern uns aber später, versprochen!

1. Erstelle im Ordner `eigenheim` eine Datei namens `index.html`. Achte darauf, dass es eine vorerst leere Textdatei ist, die die Endung `html` trägt.
2. Doppelklicke auf diese Datei, um sie im Internet Explorer zu öffnen. (Das klappt, wenn der Internet Explorer dein Standard-Browser ist.)
3. Wähle im Internet Explorer **ANSICHT | QUELTEXT**, um den Editor zu öffnen. Beginne mit dem Schreiben.

Auch hier empfehle ich nun, zwischen dem Internet Explorer und dem Editor hin- und herzuwechseln. Im Editor nimmst du deine Änderungen vor und speicherst. Im Internet Explorer aktualisierst du dann die Ansicht mit `F5`.

Du kannst also sowohl Editor als auch Internet Explorer offen lassen – wechsele einfach per Taskleiste zwischen beiden Anwendungen hin und her.

### ■ Tipps zur Wiederholung

Ordner, Dateien? Vergessen wie es geht?

**Ordner erstellen:** Starte den Windows-Explorer; am besten bei gedrückt gehaltener „Windows-Logo-Taste“ (zweite Taste von links in der unteren Reihe) durch zusätzliches Tippen von „e“. Markiere den übergeordneten Ordner, beispielsweise das Festplattensymbol C:.

Wähle **DATEI | NEU | ORDNER**. Ein neuer Ordner mit dem Platzhalternamen *Neuer Ordner* erscheint. Überschreibe den Platzhalternamen mit deinem eigenen Ordnernamen und drücke `Enter`.

**Leere Textdatei erstellen:** Klicke mit der rechten Maustaste in den noch leeren Ordner. Wähle im Kontextmenü **NEU | TEXTDATEI**. Eine leere Textdatei mit einem Platzhalternamen erscheint. Tippe den neuen Namen, z.B. `index.html` darüber, bestätige die Warnung mit **JA**. Diese Datei kannst du nun in einem Texteditor bearbeiten.

## HTML, HEAD, TITLE und BODY: Grundlagen

### Kopf und Rumpf

Zuerst zeige ich dir das Gerüst einer HTML-Datei. Schreibe ruhig alles so ab, auch mit den hier kursiv gestellten Platzhalter-Texten. Später ersetzen wir die Platzhalter und kümmern uns um den Rest:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Seitentitel</title>
<meta http-equiv="content-type"
content="text/html; charset=iso-8859-1">
</head>
<body>
Inhalt der Seite
</body>
</html>
```

### Dokumenttyp-Deklaration (DTD)

In der ersten Zeile steht die Dokumenttyp-Deklaration (DTD) – ich habe sie aus Platzgründen auf drei Zeilen umbrochen. Das ist die Angabe, die dem Browser verrät, um welche HTML-Version es sich handelt. Im Beispiel „deklarieren“ wir damit, dass wir es mit HTML 4.01 in der „Übergangsfassung“ (Transitional) zu tun haben wollen. Das ist die aktuelle Sprachfassung von HTML. Schreibe diese lange Zeile so ab und gewöhne dich daran! (Den Zeilenumbruchpfeil schreibst du natürlich nicht mit ab!) Die Zeichenfolge EN steht für die Sprache, aus der die Tags, also die Steuerbefehle in HTML, stammen – du wirst sie unschwer erraten können. Und W3C (World Wide Consortium) bezieht sich auf den Verbund, der sich um diese Sprachstandards kümmert.

Die DTD ist eine Formalie, die aber in den letzten Jahren verdammt wichtig geworden ist. Sie dient vor allem zur Unterscheidung zu anderen Sprachen bzw. Sprachstandards, z.B. XML, XSL oder XHTML. (XHTML ist übrigens die Neufassung von HTML nach strengeren Regeln.)

Über XML und damit auch über XHTML kläre ich dich ganz ausführlich auf im KnowWare-Heft „XML leicht & verständlich“ (E13).

Wenn du diese DTD-Zeile weglässt, passiert im Beispiel erst einmal überhaupt nichts – die Anzeige klappt weiterhin. Der Browser kennt ja die HTML-Tags und weiß, wie diese dargestellt werden sollen.

Trotzdem solltest du diese DTD stets mit-schreiben. Du weißt schließlich nicht, wie zukünftige Browser die Sache „sehen“ werden. Ein weiterer Effekt: Bei Weglassen dieser DTD fallen die aktuellen Browser in einen so genannten Kompatibilitätsmodus (Quirks-Modus bzw. „Pfusch-Modus“) zurück und zeigen deine Seite ggf. nicht korrekt an. Mehr dazu verrate ich dir auf Seite 42 und Folgeseite.

Es gibt auch andere Varianten von HTML, u.a. eine strengere Fassung (*Strict*), die uns weit weniger Freiheiten erlaubt. Auch eine Fassung für Framesets (Aufteilung in Rahmen) ist üblich.

### Die große Klammer: Tagpaar

**<html></html>**

Und nun auf zur zweiten Zeile, dem „eigentlichen“ Beginn der HTML-Seite! Denn das gesamte Dokument wird von <html></html> umschlossen. Dieses Tagpaar bildet also die große Klammer, die eine HTML-Seite zusammenhält.

Innerhalb dieser großen Klammer gibt es zwei „Unterklammern“, und zwar <head></head> für den Kopfbereich und <body></body> für den „Körper“ der HTML-Datei.

### Kopfbereich und Seitentitel

Schauen wir uns zuerst diesen Kopfbereich an. Das interessanteste Tag im Kopfbereich ist der Seitentitel. Dieser wird von den Tags <title></title> umfasst.

Was du zwischen den <title>-Tags notierst, erscheint später in der Titelzeile des Browsers! Außerdem dient diese Passage als Überschrift in den Suchmaschinen. Du solltest höchsten Wert darauf legen und nicht bloß „Willkommen online“ dort eintragen.

Aber es gibt im Beispiel noch ein zweites Tag im Kopfbereich, und zwar das ...

### Meta-Tag für den Zeichensatz

Schau in die fünfte Zeile. Was habe ich dort notiert? Ein Meta-Tag! Und zwar das Meta-Tag für unseren westeuropäischen Zeichensatz. Die Zeile

```
<meta http-equiv="content-type"
content="text/html; charset=iso-8859-1">
```



sorgt dafür, dass du die Umlaute oder das „ß“ nicht (mehr) umschreiben musst. Das finde ich ungeheuer praktisch und darum empfehle ich dir diese Zeile dringend!

### Der eigentliche Inhalt des Dokuments

Innerhalb von `<body></body>` (*Körper*) steht dann der eigentliche Inhalt der HTML-Seite. Mit diesem Bereich werden wir uns noch ganz ausführlich beschäftigen.

Doch nun speichere deine hoffentlich schon abge-schriebene `index.html` und schaue sie dir an!

### Tags als Steuerbefehle

Zum „Gestalten“ brauchst du die Tags, die Steuerbefehle. Diese stehen in spitzen Klammern. Fast jedes Tag besitzt ein zweites Tag zum Ausschalten. Beachte den vorangestellten Slash (Schrägstrich) im Ausschalt-Tag.

Überschriften (*headings*) stehen je nach Ebene innerhalb von `<h1></h1>`, `<h2></h2>` usw. bis `<h6></h6>`. Absätze (*paragraphs*) werden in `<p></p>` gekleidet und `<br>` signalisiert einen *break*, den einfachen Zeilenumbruch.

Manche Tags wie ein Zeilenumbruch `<br>` oder der Befehl zum Einfügen einer Grafik `<img>` brauchen kein Ausschalt-Tag.

Nur die Tags zählen. Deshalb kannst du deinen Quelltext im Editor mit Leerzeichen, Absatzumbrüchen, Tabulatorsprüngen oder Leerzeilen sauber gliedern. Alle diese Zeichen gelten für den Browser als Leerzeichen, als *white spaces*. Und wenn mehrere Leerzeichen aufeinander folgen, verbindet der Browser diese zu einem Leerzeichen.

### Keine End-Tags vergessen!

Mein wichtigster Tipp: Mache es dir zur Angewohnheit, nach dem Schreiben eines Start-Tags gleich das korrespondierende End-Tag zu setzen. Tippe es sofort in dein Dokument!

So tippe ich `<html>`, umbreche einige Leerzeilen und ergänze `</html>`. Dann schreibe ich `<head>`, tippe eine Leerzeile und füge das ausschaltende `</head>` ein, usw. usw.

Schnell vergisst du sonst das Setzen der End-Tags. Unter bestimmten Umständen (Frames, Tabellen) kann es dann Anzeige-probleme geben!

### Entitäten: Umlaute/Sonderzeichen

Umlaute und Sonderzeichen wurden früher grundsätzlich maskiert ins HTML-Dokument eingetragen, als so genannte Entitäten. Dabei handelt es sich um vordefinierte, in HTML gültige Kurzzeichen.

Heute jedoch ist das „Maskieren“ von Umlauten oder dem scharfen „s“ (ß) nicht mehr nötig – dafür haben wir schließlich das Meta-Tag für den Zeichensatz! Allerdings müssen Zeichen wie `<` `>` oder `&` unbedingt weiterhin durch ihr Kurzzeichen maskiert werden. Grund: Sie haben in HTML ihre eigene Bedeutung! Jede Entität beginnt mit `&` und endet mit `;`! Hier folgt eine kleine Auswahl, ich habe die Umlaute und das ß der Vollständigkeit halber mit angegeben.

Zeichen	Kodierung	Zeichen	Kodierung
ä	<code>&amp;auml;</code>	»	<code>&amp;raquo;</code>
Ä	<code>&amp;Auml;</code>	«	<code>&amp;laquo;</code>
ö	<code>&amp;ouml;</code>	„	<code>&amp;ldquo;</code>
Ö	<code>&amp;Ouml;</code>	“	<code>&amp;rdquo;</code>
ü	<code>&amp;uuml;</code>	– (Ged.strich)	<code>&amp;ndash;</code>
Ü	<code>&amp;Uuml;</code>	€ (Euro)	<code>&amp;euro;</code>
ß	<code>&amp;szlig;</code>	©	<code>&amp;copy;</code>
<	<code>&amp;lt;</code>	®	<code>&amp;reg;</code>
>	<code>&amp;gt;</code>	™	<code>&amp;trade;</code>
&	<code>&amp;amp;</code>	· (Mittelpunkt)	<code>&amp;middot;</code>
"	<code>&amp;quot;</code>	¶	<code>&amp;para;</code>
' (Apostroph)	<code>&amp;apos;</code>	• (Bullet)	<code>&amp;bull;</code>
erzw. Leerz.	<code>&amp;nbsp;</code>	§	<code>&amp;sect;</code>

Hast du mein Fortgeschrittenen-Heft PLUS 12? Der dort empfohlene Super-Editor *HTML-Kit* kodiert Umlaute automatisch in die entsprechenden Entitäten und prüft auch die Syntax! Auch der kostenlose *Weaverslave* ([www.weaverslave.ws](http://www.weaverslave.ws)) verfügt über diese Funktion. Schaue ins Menü **EXTRAS** und wähle **ÄNDERUNGEN | UMLAUTE ZU HTML**.

Ich persönlich bevorzuge jedoch „demaskierte“ Umlaute und ärgere mich deshalb über ältere Versionen von Dreamweaver und Co, die jeden Umlaut und das ß automatisch konvertieren, ob ich das nun will oder nicht. Später – wenn du z.B. mit JavaScript oder PHP Suchmaschinen programmieren willst – wirst du dich darüber freuen, dass du dich *nicht mehr* mit diesen Maskierungen herumärgern musst. Das ist ein Grund mehr für Handarbeit!



## Gar nicht so schwer: Grafik im GIF-Format!

Reden wir kurz noch über das Thema Grafik! Kennst du dich schon mit Grafiken und Cliparts aus? Vielleicht im Zusammenhang mit deiner Textverarbeitung Microsoft Word?

In Word nutzt man Vektorgrafiken (die schicken ClipArts) oder bindet Fotos (die so genannten Bitmaps) direkt in das Dokument ein.

In HTML-Seiten kann man allerdings keine Grafiken direkt abspeichern. Auch die Arbeit mit ClipArts im Vektorgrafik-Format ist nicht ohne weiteres möglich.

Nur Bitmap-Formate („Pixelgrafiken“) sind erlaubt. Du musst dich an bestimmte Formate halten und die Bilder per Verknüpfung einbinden. (Dazu mehr ein paar Seiten später.)

### Erlaubte Dateiformate

Die erlaubten Dateiformate heißen GIF, JPEG und PNG. Ich empfehle GIF bzw. JPEG, da PNG noch zu neu ist und noch nicht von allen Browsern unterstützt wird.

#### ■ GIF ganz kurz gefasst

Das GIF-Format komprimiert durch eine Reduktion der Farben. Die Dateigröße bleibt gering. Das GIF-Format kann aber nur 256 Farben anzeigen, das ist okay für Texte als Grafik, Buttons und einfache Logos, aber nicht immer für hochqualitative Fotos.

Nur das GIF-Format erlaubt das Definieren von transparenten Hintergrundfarben (Freistellen) oder das Erstellen von Animationen. Die Endung lautet `.gif`, jede bessere Bildbearbeitung kennt dieses Format.

#### ■ JPEG ganz kurz gefasst

Das JPEG-Format speichert bis zu 16,7 Mio. Farben, komprimiert aber mit Verlust. Je nach Kompressionsstufe entstehen hässliche Bildartefakte, erkennbar durch ausgefranste Ränder und Farbsäume.

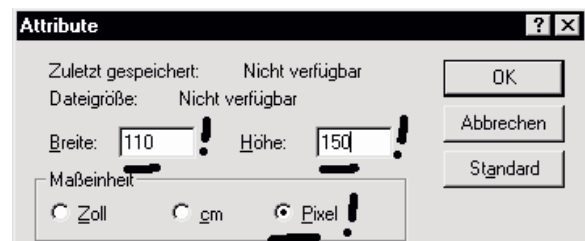
JPEG eignet sich daher nicht gut für Texte, jedoch für hochqualitative Fotos. JPEG kennt keine transparenten Farben. Man kann auch keine Animationen erstellen. Die Endung lautet `.jpg`. Auch einfache Bildbearbeitungsprogramme kennen JPEG.

### Dein erstes GIF-Bild mit Paint

Du hast keine Bildbearbeitung? Du weißt noch nicht, wie man Bilder einscannst oder von der Digi-cam übernimmt? Macht nichts, wir starten trotzdem sofort durch und malen das Bild einfach mit Paint aus Windows. Als Notbehelf.

Das Folgende funktioniert oft nur, wenn du Microsoft Word oder eine neuere Windows-Version besitzt. Denn nur dann kennt Paint die Formate GIF und JPEG, sonst nicht!

1. Starte Paint, wähle **START | AUSFÜHREN**, tippe `mspaint` und drücke `[Enter]`.

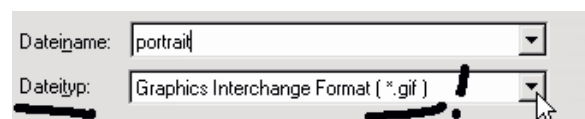


**Justiere Breite und Höhe, merke dir diese Werte!**

2. Unser Bild soll 110 Pixel breit und 150 Pixel hoch werden. Wähle also **BILD | ATTRIBUTE** und trage die gewünschten Werte in die Felder ein. Wähle **OK**.

Pixel sind Bildpunkte, das ist die Maßeinheit für die Bildschirmauflösung.

3. Zeichne dein Bild mit Paint. Male dein Porträt! (Tipp: Einen Super-Paint-Kurs gibt es auf [www.jchanke.de/windows/](http://www.jchanke.de/windows/))



**Kleinschreibung, keine Umlaute und Leerzeichen.**

4. Wähle **DATEI | SPEICHERN UNTER**, vergib einen Namen (`portrait.gif`). Wähle bei **DATEITYP** das richtige Format und speichere in deinem Projektordner ab.

Die Beispiel-Grafik heißt `portrait.gif`. Sie sollte im Beispiel exakt im gleichen Ordner liegen wie die HTML-Datei, also in `eigenheim`. Prüfe das!

## HTML pur: Das erste Projekt im Quelltext

Ran an die Praxis und damit an unser Projekt: Präsentiere dich im Web! Schreibe die folgenden Zeilen einfach ab. Die Leerzeilen und Einrückungen (white spaces) helfen lediglich, den Quelltext übersichtlicher zu gestalten, beeinflussen aber nicht das Endergebnis. Wandle den Text sinngemäß um, damit er auf deine Bedürfnisse passt. Bist du unsicher, schreibst du erst einmal alles so ab. (Halt! Die Nummerierung der Zeilen dient nur zur Veranschaulichung, die also bitte nicht mit abschreiben!)

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" ⚡
   "http://www.w3.org/TR/html4/loose.dtd">
2  <html>
3    <head>
4      <title>Homepage von Johann-Christian Hanke</title>
5      <meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
6    </head>
7    <body>
8
9    <p>Startseite - Mein Beruf - Meine Interessen - Feedback</p>
10
11   <h1>Willkommen bei Johann-Christian Hanke!</h1>
12
13   <p>
14     
15     Sie sind hier: --&gt; Startseite<br>
16     zuletzt aktualisiert: 10. Januar 2007</p>
17
18   <h2>Schön, dass Sie da sind!</h2>
19
20   <p>Guten Tag. Schön, dass Sie sich für meine Seite
21     interessieren. Ich heiße <b>Johann-Christian Hanke</b>, bin
22     38 Jahre alt und wohne in Berlin. Auf den folgenden Seiten
23     möchte ich mich Ihnen vorstellen.</p>
24
25   <h2>Mein Wahlspruch</h2>
26
27   <p>Mein Wahlspruch lautet:</p>
28
29   <p>Es gibt wichtigere Dinge im Leben als Computer!</p>
30
31   <h3>Bitte beachten:</h3>
32
33   <p>Alle Inhalte sind &copy; Johann-Christian Hanke, das Kopieren
34     der Inhalte in irgendeiner Form ist nicht gestattet.</p>
35
36   </body>
37 </html>
```

Keine Lust zum Abschreiben? Den jeweiligen „Stand der Dinge“ findest du stets im Ordner mit dem Kapitelnamen, hier also kapitell. Gehe dann einfach in den Unterordner mit dem aktuellen Stand, im Beispiel stand1. Du findest dieses Beispiel also unter kapitell/stand1/index.html. (Das fertige Endergebnis unseres Beispiels liegt dagegen im Ordner eigenheim. Download via [www.knowware.de/?book=css!](http://www.knowware.de/?book=css!))

## So sieht's im Browser aus

Schauen wir uns nun den Quelltext an, wie er im Browser dargestellt wird. Ob nun Internet Explorer oder Firefox, ist ziemlich egal. Ausgewählte Elemente habe ich beschriftet, damit du sie im Quelltext sofort wiederfindest. Auf der nächsten Seite klären wir die Einzelheiten im Detail!

Seitentitel <title></title> vgl. Zeile 4

Absatz <p></p> mit vorbereiteten Links vgl. Zeile 9

Willkommen bei Johann-Christian Hanke! <h1></h1> vgl. Zeile 11

Sie sind hier: --> Startseite  
zuletzt aktualisiert: 10. Januar 2007

Absatz <p></p> und „Break“ <br> vgl. Zeile 13-16

Schön, dass Sie da sind! Zeichenformat fett <b></b> vgl. Zeile 21

Guten Tag. Schön, dass Sie sich für meine Seite interessieren. Ich heiße **Johann-Christian Hanke**,  
<h2></h2> vgl. Zeile 18 und 25 den folgenden Seiten möchte ich mich Ihnen vorstellen.

In Absatz eingebundene Grafik <img> vgl. Zeile 14

Mein Wahlspruch

Me <h3></h3> vgl. Zeile 31

Es gibt wichtigere Dinge im Leben als Computer!

Bitte beachten: Entität &copy; vgl. Zeile 33

Alle Inhalte sind © Johann-Christian Hanke, das Kopieren der Inhalte in irgendeiner Form ist nicht gestattet.

Fertig

Der Browser stellt alle Tags auf eine fest vorgegebene Art und Weise dar. Als Anzeigeschrift wählt der Browser eine Times bzw. Times New Roman.

Was stellst du fest? Richtig, es handelt sich um die unerträgliche Selbstdarstellung eines EDV-Autors. (Obwohl eine „Web-Visitenkarte“ für potenzielle Brötchengeber durchaus nützlich sein kann!)

Abgesehen davon habe ich in dieses Dokument ein paar didaktisch-nützliche Details eingebaut: Gib auf Wunsch das *Änderungsdatum* an und verweise ggf. auch auf dein *Copyright* – auch wenn's oft nicht viel hilft. Zumindest übst du in diesem Fall gleich mal das ©-Zeichen. Denke vor allem an eine *Kontaktmöglichkeit* (Feedback, Impressum o. ä.), damit dich die Besucher erreichen können.

Wiederholen wir kurz die „Grobstruktur“! **HTML**: In Zeile 2 wird das Dokument mit <html> begonnen, der Abschluss erfolgt in Zeile 37 mit </html>. **HEAD**: Der Kopfbereich <head></head> beginnt in Zeile 3 und endet in Zeile 6. In unserem Beispiel besitzt der HEAD lediglich einen Seitentitel. **BODY**: Die „eigentliche Musik“ spielt innerhalb des „Containers“ <body></body>. Der BODY wird in Zeile 7 eingeleitet und in der vorletzten Zeile (Zeile 36) wieder geschlossen.

Schauen wir uns jetzt die Einzelheiten an, blättere dafür später immer wieder zurück!

## Blick hinter die Kulissen: HTML-Tags im Detail

Werfen wir nun einen kritisch-genauen Blick auf den HTML-Code. Was haben wir da gemacht?

### Der Kopfbereich im Detail

Jedes HTML-Dokument besitzt bekanntlich einen Kopfbereich, den HEAD. Bei uns befinden sich hier ein lumpiger Seitentitel und das schon erwähnte Meta-Tag für unseren westeuropäischen Zeichensatz, mehr nicht. Der HEAD dient jedoch oft als „Behälter“ für mehr oder weniger sinnvolle Anweisungen und Befehle.

**Stichwort Style Sheets:** Um unser Dokument zu gestalten, brauchen wir Style Sheets. Später werden wir unsere Stilanweisungen u. a. im Kopfbereich ablegen bzw. einen Verweis auf eine externe Stildatei einbinden.

**Stichwort JavaScript:** Im Fortgeschrittenen-Heft lernst du JavaScript kennen. Damit verleihst du deinen Seiten mehr Pepp. In der Regel legt man den JavaScript-Code ebenfalls im Kopfbereich ab.

**Stichwort Meta-Tags:** Im Einsteiger-Heft habe ich dich kurz mit den Meta-Tags vertraut gemacht. Einige Meta-Tags sorgen z. B. dafür, dass deine Seite besser gefunden wird: Gib auf Wunsch Autor (author), Stichworte (keywords) oder Beschreibung (description) der Seite an. Das sind zumindest die drei wichtigsten zusätzlichen Meta-Tags.

### Absätze und Zeilenumbruch

In Zeile 9 habe ich die Hyperlinks vorbereitet. Schließlich besteht ein Web-Projekt aus mehreren Seiten, die (im Beispiel später) miteinander verbunden werden. Die Links stehen innerhalb eines Absatzes `<p></p>`.

Text sollte innerhalb eines Absatzes notiert werden (oder innerhalb eines anderen so genannten Block-Elements). Absätze werden vom Browser stets mit einem kleinen vertikalen Abstand angezeigt.

Interessant ist der zweite Absatz, der von Zeile 13 bis 16 reicht. Er beherbergt nicht nur unsere Grafik in Zeile 14, sondern einen Zeilenumbruch (*break*). Du willst, dass Text auf einer neuen Zeile beginnt? Füge das Tag `<br>` ein. Die Passage *zuletzt aktualisiert* beginnt auf einer neuen Zeile; der sonst für Absätze typische Abstand fehlt. Einen längeren Absatz findest du in Zeile 20 bis 23, einen kurzen in Zeile 33/34.

Beachte, dass der Umbruch im Browserfenster nichts mit dem Umbruch im Quelltext zu tun hat! Die Breite des Textes passt sich normalerweise der Breite des Browserfensters an!

### Die Überschriftsebenen

Die erste Überschrift findest du in Zeile 11. Es ist eine `<h1></h1>`, eine Hauptüberschrift. Zeile 18 und 25 beherbergen je eine Unterüberschrift zweiter Ordnung (`<h2></h2>`) und in Zeile 31 gibt es eine Überschrift 3 (`<h3></h3>`).

Die Überschriften sind von der Größe her abgestuft. Sie „schaffen“ sich ebenfalls einen Abstand nach oben und unten – zumindest im Browser. Auch Überschriften zählen übrigens als „Blockelemente“.

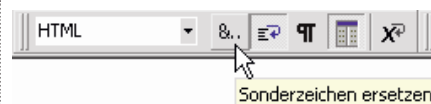
Der Browser zeigt Textpassagen, Absätze und Überschriften normalerweise in der Standardschrift Times (New Roman) an. Es gibt dafür jedoch keine zwingende Regel.

### Formate für fett und kursiv

Für meinen Namen habe ich in Zeile 19 das Format fett (*bold*) gewählt. Dazu dient das Tag-Paar `<b></b>`. Kursiv (*italic*) wäre `<i></i>`. Du kannst statt `<b>` aber auch `<strong>` (*stark*) verwenden und statt `<i>` einfach `<em>` (*deutlich*). Diese alternativen Tags werden von den Machern von HTML bevorzugt, da es logische Tags sind, die keine direkten Formateigenschaften transportieren.

### Welchen Editor bevorzugst du?

Die ganz Hartgesottenen arbeiten weiter mit dem Windows-Editor, dem „Notepad“. Ich empfehle dir jedoch den besagten Weaverslave, den du dir unter [www.weaverslave.ws](http://www.weaverslave.ws) herunterladen kannst. Wähle **DATEI|NEU** und bleibe im ersten Register. Die Auswahl des Symbols **HTML 4 TRANS.** erzeugt das Grundgerüst einer HTML-Datei mit DTD! Das „Farbhervorhebungsschema“ wird damit auch gleich auf HTML eingestellt:



Deaktiviere außerdem die Schaltfläche **SONDERZEICHEN ERSETZEN**, damit deine Umlaute **nicht** automatisch in die Entitäten konvertiert werden!

## So fügst du eine Grafik ein: <img> und seine Attribute

Richte deinen Adlerblick doch einmal zielgenau auf Zeile 14. Hier haben wir die Grafik eingebunden. Doch wie? Und warum ist die Zeile so lang? Ran an das Objekt: Auf zum Sturzflug!

### Verknüpfen statt Speichern!

Da liegt dein Konterfei nun unter dem Namen `portrait.gif` im Ordner `eigenheim`. Prima. Doch wie kommt so eine Grafik in das HTML-Dokument?

In HTML werden Grafiken durch Verknüpfen eingebunden!

Zum Verknüpfen bedienst du dich des `<img>`-Tags. Das Kürzel `img` steht für *image*, Bild.

Die Grundsyntax lautet:

```

```

Den Pfad sparen wir uns, denn die Grafik liegt im gleichen Ordner. Im Beispiel notieren wir:

```

```

Und siehe da, deine Abbildung wird folgerichtig in das Dokument eingebunden. Vergleiche mit meinem Musterdokument „Grafik als Zeichen“ aus dem Ordner `kapitel1/stand2`.



Die Grafik benimmt sich wie ein (riesiges) Zeichen.

Allerdings geht dadurch der schöne Textfluss flöten. Die Grafik fläzt sich ganz am Anfang der Zeile herum. Und tatsächlich benimmt sich eine Grafik eigentlich wie ein riesiges Zeichen!

Sollte eine Grafik alleine stehen, müsstest du ihr einen eigenen Absatz spendieren!

### Grafik ausrichten mit align

Unsere Grafik soll aber rechtsbündig ausgerichtet werden. Dafür nutzen wir das HTML-Attribut `align`, denn dieses sorgt zusätzlich für den Textumfluss. Mit `align` legst du die horizontale Ausrichtung eines

Elements fest. Ich empfehle bei Grafiken die Werte `left` bzw. `right` (links oder rechts). Ergänze die Zeile also wie folgt:

```

```

Und schon marschiert die Abbildung nach rechts. Der Text fließt links daran vorbei. Du siehst, dass Attribute viel bewirken können.

Nach dem Attribut folgt ohne Leerzeichen das Gleichheitszeichen, danach der entsprechende Attribut-Wert in Gänsefüßchen. Attribute werden durch Leerzeichen voneinander getrennt.

Vorgriff auf CSS: Du kannst statt `align="right"` auch die Zeichenfolge `style="float: right;"` einsetzen. Das ist moderner! Dahinter verbirgt sich eine Inline-CSS-Anweisung, die exakt das Gleiche bewirkt wie `align="right"`.

Die Reihenfolge der Attribute innerhalb eines Tags spielt keine Rolle.

### Breite und Höhe angeben

Füge noch die Attribute für Breite (`width`) und Höhe (`height`) in das `<img>`-Tag ein. Nimm dafür die Pixel-Werte, die du aus deiner Bildbearbeitung ausgelesen hast (Seite 8). Im Beispiel ergänze ich die Attribut-Werte-Paare: `width="110" height="150"`.

Damit sicherst du, dass die Breite und Höhe des Bildes von Anfang an feststehen. Das hässliche und unprofessionelle „Springen“ der Bilder beim Seitenaufbau entfällt.

### Wichtig: Alternativtext

Als letztes Attribut haben wir einen Alternativtext in das `<img>`-Tag eingefügt. Auch das ist Pflicht! Dafür dient das Attribut `alt`. Als Wert nimmst du einen beschreibenden Text. Das kann ein Wort oder eine Wortgruppe sein. Bei mir heißt es vorerst: `alt="Willkommen"`.

Benutzer mit abgeschalteter Bildanzeige sehen wenigstens den Alternativtext. Mehr Vorteile der `alt`-Angabe zeige ich dir im Übungsteil.

Wenn du den Alternativtext vergisst, handelt es sich nicht um eine korrekt eingebundene Abbildung! Die HTML-Seite wird nicht als gültig anerkannt. Mehr zum Prüfen von HTML-Dokumenten erfährst du auf Seite 77.

## Zusatzwissen für Profis: Grundbegriffe kurz erklärt

### Was ist eine Homepage?

Eine Homepage ist die Startseite deiner Publikation im World Wide Web. Der Dateiname lautet in der Regel *index.html* (oder *index.htm*). Die Publikation, auch als Website bezeichnet, besteht in der Regel aus mehreren Seiten.

Landläufig wird allerdings das gesamte Projekt als Homepage bezeichnet. Fachleute sagen dazu aber Website, das Wort *site* steht für Platz oder Stelle.

### Was bedeutet HTML?

HTML ist die Abkürzung für Hypertext Markup Language, Sprache zur Auszeichnung von Hypertext. Hypertext ist Text mit Hypersprüngen, den Querverweisen. „Ausgezeichnet“ wird der Text durch Tags. HTML besteht aus normalem Text.

### Was ist ein Tag?

Das Wort Tag heißt so viel wie Etikett. Es handelt sich um einen Steuerbefehl zum Einschalten einer Eigenschaft. Eine Überschrift erster Ordnung wird durch `<h1>` eingeschaltet.

Fast jedes Start-Tag besitzt in der Regel ein End-Tag, eine Überschrift 1 wird durch `</h1>` wieder abgeschaltet. Tags müssen korrekt verschachtelt werden: `<h1><b></b></h1>`, falsch ist dagegen `<h1><b></h1></b>`.

### Welches sind die wichtigsten Tags?

Tag(-Paar)	Erläuterung
<code>&lt;html&gt;&lt;/html&gt;</code>	umschließt HTML-Dokument
<code>&lt;head&gt;&lt;/head&gt;</code>	umschließt Kopfbereich
<code>&lt;title&gt;&lt;/title&gt;</code>	definiert Seitentitel
<code>&lt;body&gt;&lt;/body&gt;</code>	umschließt Seitenkörper
<code>&lt;h1&gt;&lt;/h1&gt;</code>	erzeugt Überschrift erster Ordnung
<code>&lt;h2&gt;&lt;/h2&gt;</code>	erzeugt Überschrift zweiter Ordnung
...	...
<code>&lt;h6&gt;&lt;/h6&gt;</code>	erzeugt Überschrift sechster Ordnung
<code>&lt;p&gt;&lt;/p&gt;</code>	umschließt Absatz
<code>&lt;br&gt;</code>	erzeugt Zeilenumbruch (kein End-Tag!)
<code>&lt;hr&gt;</code>	erzeugt horizontale Linie (kein End-Tag!)
<code>&lt;b&gt;&lt;/b&gt;</code>	formatiert fett, empfohlene Alternative: <code>&lt;strong&gt;&lt;/strong&gt;</code>
<code>&lt;i&gt;&lt;/i&gt;</code>	formatiert kursiv, Alternative: <code>&lt;em&gt;&lt;/em&gt;</code>
<code>&lt;u&gt;&lt;/u&gt;</code>	Unterstreichungen – nur in Ausnahmen verwenden – Verwechslungsgefahr mit Links!
<code>&lt;img&gt;</code>	bindet eine Grafik ein, Attribut <code>src</code> nötig, <code>&lt;img src="portrait.gif"&gt;</code> , verwende auch die Zusatz-Attribute <i>width</i> , <i>height</i> und <i>alt</i> !

### Was sind die wichtigsten Attribute?

Einige wichtige Attribute zeige ich dir hier:

Attribut	Wert	Erläuterung
<code>align</code>	<code>left</code>	Ausrichtung <b>links</b> (für <code>&lt;p&gt;</code> , <code>&lt;img&gt;</code> , <code>&lt;h1&gt;</code> bis <code>&lt;h6&gt;</code> )
	<code>right</code>	Ausrichtung <b>rechts</b> (für <code>&lt;p&gt;</code> , <code>&lt;img&gt;</code> , <code>&lt;h1&gt;</code> bis <code>&lt;h6&gt;</code> )
	<code>center</code>	Ausrichtung <b>mittig</b> (für <code>&lt;p&gt;</code> und <code>&lt;h1&gt;</code> bis <code>&lt;h6&gt;</code> )
	<code>justify</code>	Ausrichtung im <b>Block</b> ( <code>&lt;p&gt;</code> )
<code>width</code>	Pixelangabe	legt <b>Breite</b> fest ( <code>&lt;img&gt;</code> , <code>&lt;hr&gt;</code> )
<code>height</code>	Pixelangabe	legt <b>Höhe</b> fest ( <code>&lt;img&gt;</code> , <code>&lt;hr&gt;</code> )
<code>src</code>	Pfad zu Datei	Grafik-Quelle <code>&lt;img src=""&gt;</code>
<code>alt</code>	Wort(gruppe)	Alternativtext für Grafik
<code>hspace</code>	Pixelangabe	Horizontalabstand für Grafik
<code>vspace</code>	Pixelangabe	Vertikalabstand für Grafik
<code>border</code>	Pixelangabe	<b>Rahmen</b> um Grafik

### Wie setzt man Kommentare?

Du möchtest im HTML-Quelltext einen Kommentar einbinden, eine Textzeile, die nicht im Browser angezeigt wird? Setze den Kommentar innerhalb von `<!-- ... -->`:

```
<!-- Das ist ein Kommentar -->
```

### Wer entwickelt diese Standards?

Die Web-Standards werden von einer Organisation namens World Wide Web Consortium (W3C) erarbeitet. Das W3C wird von Tim Berners-Lee geleitet, dem Erfinder von HTML. Ins Deutsche übersetzte Artikel des W3C gibt's auf [www.w3c.de](http://www.w3c.de), ansonsten siehe: [www.w3.org](http://www.w3.org)!

### Müssen alle Dateien in einen Ordner?

Nein, natürlich nicht! Bei großen Projekten ist es sinnvoll, themenspezifische Unterordner einzurichten. Bei kleinen Projekten genügt aber ein Ordner, da man sich so die Pfade beim „Verlinken“ der Seiten sparen kann.

### Warum als Startseite *index.html*?

Per Voreinstellung wird eine *index.html* auf einem Server auch dann gefunden, wenn man nur den Ordernamen (also den jeweiligen „Root“) eintippt. Tippe [www.jchanke.de/homepage](http://www.jchanke.de/homepage), im Hintergrund wird [www.jchanke.de/homepage/index.html](http://www.jchanke.de/homepage/index.html) ins Browserfenster geladen.



## ✓ Übungsteil A: Erste Schritte mit HTML

Du weißt jetzt, wie man:

- ein HTML-Grundgerüst erstellt
- Überschriften einsetzt
- Absätze, Text und Zeilenumbrüche notiert
- eine Grafik einbindet und mit Attributen arbeitet



Meine Übungen sind Pflicht, denn du lernst auch Neues! Erstelle für die Übungen unter deinem Ordner homepage einen Ordner namens aufgaben. Richte unter dem Ordner aufgaben folgende Unterordner ein: a1, a2, a3. Kopiere die bisher erstellten Dateien index.html und portrait.gif als „Übungsmuster“ jeweils in diese drei Unterordner. (Profi-Übungen sind freiwillig, da schwerer! :-)

### ■ Übung A1a: Linie einsetzen, Block-Element „Absatz“ ausrichten

Gehe in den Ordner a1, bearbeite die dortige index.html. Setze unter die Hauptüberschrift `<h1></h1>` eine Linie `<hr>` ein. (Linien können alleine stehen, brauchen kein Block-Element drum herum.)

Richte außerdem den Absatz *Es gibt wichtigere Dinge im Leben als Computer!* zentriert aus.



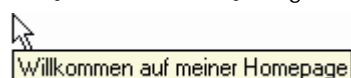
Nicht schön, aber selten: Linie in HTML.

### ■ Übung A1b (Profi): Die Attribute width und align einsetzen

Bleibe im Ordner a1. Das Attribut width kann auch auf eine Linie angewandt werden. Sorge dafür, dass die Linie 200 Pixel breit ist. Nanu? Welches ist offenbar die voreingestellte Standard-Ausrichtung für Linien? Sorge mit dem align-Attribut dafür, dass die Linie wieder am linken Rand beginnt!

### ■ Übung A2: Grafik in eigenen Absatz setzen, alt-Text nutzen

Gehe in den Ordner a2 und bearbeite die index.html. Stelle die Grafik in einen eigenen Absatz. Sie soll direkt unter der `<h1></h1>` erscheinen. Richte sie zentriert aus, ohne dass Text drum herum fließt. *Tipp:* Nimm align aus dem `<img>`-Tag heraus, richte die Grafik mit Hilfe des Absatzes aus!



Ändere den Alternativtext von *Willkommen* zu *Willkommen auf meiner Homepage*. Parke den Mauszeiger eine Weile über der Grafik. Was passiert? Es passiert nichts? Weil dein Browser nicht der Internet Explorer ist? Dann gehe zu Seite 20 und lies den Abschnitt über das Attribut title. Du kannst es im img-Tag zusätzlich zum Attribut alt verwenden. Nimm einfach für beide Attribute den gleichen Text – und dann erscheint das kleine Fensterchen auch in den meisten anderen Browsern.

### ■ Übung A3a (Profi): Grafik linksbündig ausrichten

Gehe in den Ordner a3 und bearbeite die index.html. Richte die Grafik linksbündig aus, der Text soll rechts daran vorbeifließen. Du merkst, dass der Text zu dicht neben der Grafik steht. Verwende das Attribut hspace, um 10 Pixel Raum zwischen der Grafik und dem umgebenden Text zu lassen.

### ■ Übung A3b (Profi): Grafik im JPEG-Format abspeichern

Lade die Grafik in deine Bildbearbeitung, notfalls in Paint. Speichere sie zusätzlich im JPEG-Format ab, lege sie in den Ordner a3. Vergleiche die Dateigrößen beider Grafiken. Welche ist größer, die GIF-Datei oder die JPEG-Fassung? Binde nun statt der GIF-Datei diese JPEG-Grafik in deine Datei index.html aus dem Ordner a3 ein. Ändere das `<img>`-Tag also dementsprechend ab.

### ■ Übung A3c (Profi): Kommentar einfügen

Füge vor der Grafik einen Kommentar ein, der besagt, dass du hier ein Porträt eingebunden hast.

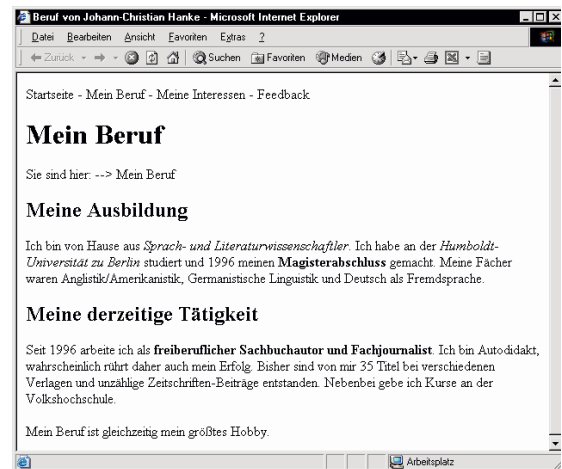


### ■ Übung A4: Ein neues HTML-Dokument erstellen: beruf.html

Arbeite ab jetzt wieder im Ordner `eigenheim`. Erstelle ein neues HTML-Dokument. Nenne das Dokument `beruf.html`. Berichte über deinen Beruf, arbeite mit den drei Überschriftsebenen und mit Textabsätzen. Nutze ruhig das Format `kursiv`.

Baue das Dokument ähnlich auf wie die `index.html`. Vergiss nicht die Vorbereitung für die Linkeiste ganz oben. Verzichte jedoch auf die Passage *zuletzt aktualisiert* zu Beginn und den Abschnitt *Bitte beachten* am Schluss. Auch eine Grafik ist nicht nötig. Achte auf den Seitentitel. Setze die Formate **fett** und *kursiv* ein.

Wenn du keine eigenen Ideen hast, orientiere dich an meinem Vorschlag. (Tipp: Mit *Speichern unter* lassen sich Teile des ersten Dokuments übernehmen.)



Ist es sinnvoll, die Absätze mit dem Format `Block (align="justify")` zu versehen? Probiere es aus. Um den Effekt von Blocksatz zu sehen, musst du das Browserfenster sicher verkleinern.

### ■ Übung A5: Nummerierung und Aufzählung: Die Seite hobby.html

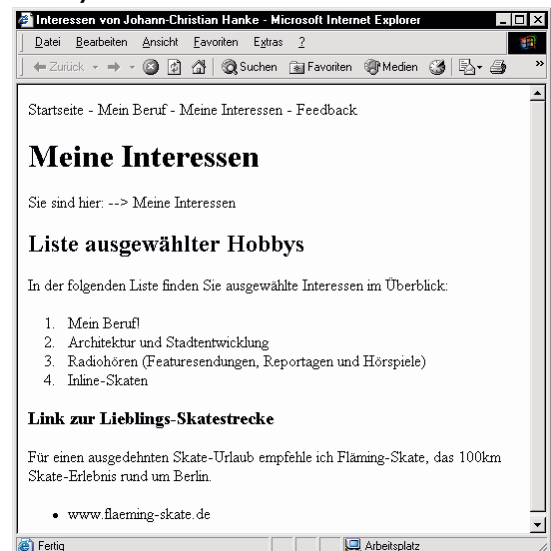
Zeit, dir etwas Neues beizubringen. Erstelle in HTML eine automatische Nummerierung!

Eine Nummerierung wird insgesamt von `<ol></ol>` eingerahmt, `ol` steht für *ordered list*. Jeder einzelne Eintrag wiederum steht innerhalb von `<li></li>`

(`li` für *list item*, Listeneintrag):

```
<ol>
  <li>Eintrag 1</li>
  <li>Eintrag 2</li>
</ol>
```

Mit diesem Wissen erstellst du die Seite `hobby.html`. Berichte über deine Interessen. Liste deine Hobbys in so einer „geordneten Liste“ auf. Lege das Dokument in den Ordner `eigenheim`. Bereite einen Link vor zu einer interessanten Seite, die mit deinem Hobby zu tun hat.



Setze diesen künftigen Link ebenfalls in eine „Liste“. Wähle diesmal jedoch statt einer *ordered list* eine *unordered list*. Dazu „umhüllst“ du die Liste statt mit `<ol></ol>` einfach mit `<ul></ul>`:

- Am Anfang eines Listeneintrags steht nun ein Bullet, ein kleiner Aufzählungspunkt.

Orientiere dich an meinem Gestaltungsvorschlag, wenn du möchtest. (Die Lösungen für die letzten beiden Aufgaben findest du übrigens unter `kapitel1/stand3`.)

### ■ Übung A6 (Profi): Was ist eine Dokumenttyp-Deklaration (DTD)?

Warum beginnen manche HTML-Seiten mit einer solchen oder ähnlichen Zeile? Muss das sein?

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

Informiere dich ausführlicher über dieses Thema! Eine gute Adresse dafür ist Stefan Münz' berühmte SELFHTML, hier speziell die Seite <http://de.selfhtml.org/html/allgemein/grundgeruest.htm#dokumenttyp>.

## Kapitel 2: Hyperlinks, die „Kreuz- und Querverweise“

### Seiten verbinden: Interne Links

Seien wir ganz offen: So richtig lebendig wird das Web erst durch die Hyperlinks, die Querverweise. Und genau an dieser Stelle machen wir jetzt weiter, nun wird das Projekt intern verlinkt!

Mit internen Links „verknüpfst“ du mehrere HTML-Seiten zu einer Web-Publikation.

Wir haben bisher folgende Seiten erstellt:

- index.html (Startseite)
- beruf.html (Mein Beruf)
- hobby.html (Meine Interessen)

Diese „verweben“ wir nun zu einer richtiggehenden Web-Publikation. Auf jeder Seite soll es Verweise zu den beiden anderen geben.

Werfen wir zuerst einen Blick auf die Startseite, auf die Datei index.html. Öffne die Datei, falls du sie momentan nicht vor dir hast. (Den bisherigen Stand findest du auch im Ordner kapitel2/stand1.)

Startseite - Mein Beruf - Meine Interessen - Feedback

#### Die künftigen Links warten auf ihren Auftritt.

Hier hast du die künftigen Links schon vorbereitet. Einmal werden wir auf beruf.html, zum anderen auf hobby.html verweisen.

Der vorbereitete Feedback-Link kommt auch noch zu seinem Recht, und zwar etwas später!

### Grundsyntax der Hyperlinks

So sieht die Grundsyntax für Hyperlinks aus:

```
<a href="pfad/dateiname.end"> ... </a>
```

Du arbeitest mit dem <a>-Tag und verwendest das Attribut href. a steht für *anchor*, Anker. Die Zeichenfolge href ist die Abkürzung für **hyper** **referen** **ce**, Querverweis.

Du setzt einen Querverweis auf einen Anker!

Statt der Pünktchen wird die „Linkbeschreibung“ eingesetzt, also das, was der Surfer später sieht. Das kann Text oder auch eine Grafik sein!

### Der erste Hyperlink

Die ersten Hyperlinks setzen wir gemeinsam! Du steckst im Quelltext der Startseite? Dann mal los! Der erste Querverweis soll auf die Seite beruf.html „zeigen“.

1. Suche zuerst die Zeichenfolge *Mein Beruf*. Das ist die Linkbeschreibung, sie steht schon da. Nun stricken wir den „Anker-Code“ drumherum. Bisher sieht es in deinem Quelltext so aus:  
  
Mein Beruf
2. Klicke vor die Zeichenfolge *Mein Beruf*. Schreibe <a href="">. Vergiss nicht: Innerhalb der Gänsefüßchen notierst du den Dateinamen der Seite, auf die du verweisen willst. Insgesamt lautet der Quelltext bisher:

```
<a href="beruf.html">Mein Beruf
```

Der Dateiname ist hierbei der Anker, auf den die Referenz gesetzt wird!

3. Denke auch an das End-Tag. Ergänze das fehlende </a>.

```
<a href="beruf.html">Mein Beruf</a>
```

Fertig ist der Link. Herzlichen Glückwunsch!

### Der zweite Hyperlink

Vergiss nicht, den zweiten Link zu setzen:

```
<a href="hobby.html">Meine Interessen</a>
```

Denke an die Leerzeichen und die kleinen Striche zwischen den Links (wegen der Optik).



Links werden blau und unterstrichen dargestellt.

(Unbesuchte) Hyperlinks erscheinen in den meisten Browsern per Voreinstellung blau und unterstrichen, besuchte färben sich dagegen lila. Beim Darüberfahren wird der Mauszeiger zu einer Hand mit ausgestrecktem Zeigefinger. Auch das ist eine Interpretation des Browsers.

## Hyperlinks ausprobieren

Klar, warum nicht. Probiere deine neuen Hyperlinks gleich einmal aus. Klicke auf [Mein Beruf](#). Schon landest du auf der Seite `beruf.html`. Von hier gelangst du mit dem **ZURÜCK**-Button des Browsers zurück zur Ausgangsseite.



### Auf der Berufs-Seite fehlen noch die Links!

Aber bleiben wir doch einfach auf der Seite *Mein Beruf*. Schließlich fehlen auch hier die Hyperlinks. Öffne den Quelltext, wähle im Internet Explorer beispielsweise **ANSICHT | QUELLTEXT** oder rufe die Seite im Weaverslave auf.

Warum bleibt auf der `index.html` der Text *Startseite* unbearbeitet? Warum werde ich auf der Seite `beruf.html` die Wortgruppe *Mein Beruf* und auf der `hobby.html` die Passage *Meine Interessen* in Ruhe lassen? Nun, ich will erstens keine Links setzen, die auf sich selber verweisen. Später hebe ich zweitens diese Textstellen noch so hervor, dass der Besucher sofort sieht, auf welcher Seite sie oder er gerade steht!

### Die Links auf der Seite `beruf.html`

Versuche einmal, die Links auf der Seite `beruf.html` ganz selbstständig einzusetzen.

Zur Erinnerung: Nur die Passage *Startseite* (an erster Stelle) und *Meine Interessen* (dritte Stelle) werden in einen Link eingebunden. Die Textstellen *Mein Beruf* (zweite Position) und *Feedback* (letzte Stelle) lässt du unbearbeitet.

Geschafft? Dann vergleiche und schreibe nach Möglichkeit alles auf eine Zeile.

Falls du doch Umbrüche im Quelltext setzen willst, dann bitte da, wo ein Leerzeichen steht. Nur innerhalb von Gänsefüßchen notierte Attributwerte sollten nicht „auseinandergerissen“ werden.

```
<p><a href="index.html">Startseite</a>
- Mein Beruf -
<a href="hobby.html">Meine Interessen</a>
- Feedback</p>
```

Hat alles funktioniert? Dann müsste deine Seite jetzt so aussehen:



### Besuchte Links (Startseite) werden lila angezeigt.

Du merkst: Besuchte Links werden vom Browser in der Regel lila dargestellt, unbesuchte dagegen blau. Das ist die Interpretation, die der Browser seinem so genannten eingebauten Style Sheet entnimmt. Dieses eingebaute Style Sheet ist praktisch ein vorgegebenes Standard-Gestaltungsschema.

Keine Sorge, mit individuellen Style Sheets werden wir später all diese Parameter beeinflussen und sogar die Unterstreichungen abschalten!

### Die Links auf der Seite `hobby.html`

Zum Schluss fehlen noch die Links auf der Seite `hobby.html`. Sie sehen folgendermaßen aus:

```
<a href="index.html">Startseite</a> -
<a href="beruf.html">Mein Beruf</a>
```

Das hättest du bestimmt auch alleine geschafft!



### Fertig! Alle Seiten sind miteinander verbunden.

Zur Feier dieses Ereignisses gönnst du dir ein Glas echten Champagner (oder auch zwei)!

Hoppla, da ist zwischendurch irgendetwas schief gelaufen? Vergleiche mit meinen Beispieldateien. Den aktuellen Stand des Projekts findest du unter `kapitel2/stand2`.

## Mehr Feedback durch den E-Mail-Link

Bestimmt möchtest du wissen, wie deine Seite bei den Besuchern ankommt. (Oder du willst, dass sich ein potenzieller Brötchengeber auch wirklich bei dir meldet.)

Dann kannst du auch eine Feedback-Möglichkeit einbauen. Schließlich steht bei uns auf allen drei Seiten schon das Wort *Feedback* und wartet darauf, dass du es mit Leben füllst. Rein zu Übungszwecken natürlich.

### Das Schlüsselwort **mailto:**

Es ist einfacher als du vielleicht denkst. Nutze die schon bekannte `<a>`-Syntax für einen simplen Hyperlink, in leicht abgewandelter Form.

Statt der entsprechenden HTML-Seiten (bzw. der Webadresse) bauen wir die E-Mail-Adresse in die Gänsefüßchen ein.

Zusätzlich muss das Schlüsselwort **mailto:** eingefügt werden, mit Doppelpunkt!

Und so sieht solch ein Muster-E-Mail-Link aus:

```
<a href="mailto:deine@adresse.de">
Linkbeschreibung (Feedback etc.)</a>
```

Setze kein Leerzeichen hinter dem Schlüsselwort `mailto:`!

### ■ Übung macht den Meister!

Deine Aufgabe: Ergänze auf allen drei Seiten den E-Mail-Link! Stricke ihn um das Wort *Feedback* drum herum. Setze deine E-Mail-Adresse ein.

Meine Variante sieht so aus:

```
<a href="mailto:css@lexi.de">Feedback</a>
```

Wenn du mir Feedback „geben“ möchtest, nutze bitte das über [www.jchanke.de](http://www.jchanke.de) erreichbare Feedback-Formular und keinesfalls diese Adresse!

Innerhalb der `<a>`/`</a>`-Tags kannst du im Prinzip schreiben was du willst. Egal ob *Feedback*, *E-Mail an mich*, *Schreib mir mal* oder *Klicken verboten*. Du kannst auch deine E-Mail-Adresse im Klartext hier eintragen. Es passiert auf jeden Fall das Gleiche, doch was?

### So funktioniert **mailto:**

Was passiert, wenn du auf einen E-Mail-Link klickst? Das Schlüsselwort `mailto:` signalisiert dem Browser, dass eine E-Mail-Adresse folgt.

Er versucht jetzt, dein Standard-E-Mail-Programm zu starten, ruft hier ein neues E-Mail-Formular auf und trägt die E-Mail-Adresse aus dem E-Mail-Link gleich als Empfänger ein.



Wenn es klappt, erscheint das E-Mail-Formular.

Schreibe deine Mail und ab geht das Feedback! Praktisch, wenn es funktioniert.

Das entsprechende E-Mail-Programm muss richtig eingerichtet sein!

### E-Mail-Link mit **Betreff**

Du möchtest diesem E-Mail-Link gleich einen Betreff mit auf den Weg geben? Dann schreibe nach dem Muster

```
<a href=
"mailto:deine@adresse.de?subject=Betrefftext">
Linkbeschreibung (Feedback etc.)</a>
```

Setze statt *Betrefftext* eine aussagekräftige Wortgruppe ein. So wird automatisch auch gleich die Betreffzeile im E-Mail-Formular ausgefüllt.

### Nachteile des E-Mail-Links

Wer seine E-Mails aber über eine Webseite managt (Google-Mail, GMX, Freemail usw.), kann häufig mit diesen E-Mail-Links nichts anfangen. Surfer im Internetcafé werden von den E-Mail-Links ebenfalls „überfordert“ sein, schließlich fehlt das E-Mail-Programm.

Außerdem sind derartige Feedback-Links ein gefundenes Fressen für Spamversender. Sie lesen solche Adressen automatisch von den Webseiten aus und nutzen sie für den Versand von Werbemails. Legal ist das nicht, aber das stört solche Leute kaum.

Die professionellere Feedback-Versendemethode erfolgt daher über Formulare und das Verwenden eines so genannten „Formmailers“. Alle Details dazu verrate ich dir in meinem Heft zu PHP 5.

## Links nach draußen: Die externen Hyperlinks

Du merkst es: Wir steigern uns. Zum Schluss bauen wir einen Hyperlink ein, der nach draußen führt, also ins World Wide Web.

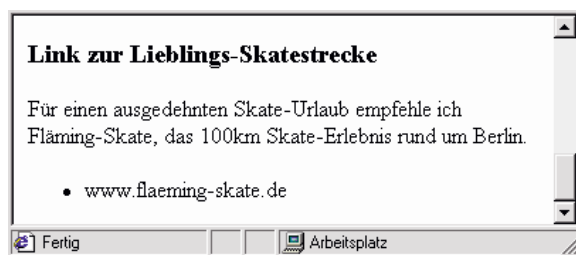
Auch hier bleiben wir bei der bekannten Grundsyntax. Auch hier hilft uns wieder das `<a>`-Tag.

Diesmal wird als „Anker“ jedoch eine externe Webseite eingebunden, und zwar in voller Pracht.

### Workshop: Der Hobby-Link

Ran an die Praxis, mache wieder mit! Rufe dafür deine Seite `hobby.html` auf. Schließlich hatte ich dich in Übung A5 darum gebeten, einen künftigen Link vorzubereiten.

Damit es noch schicker aussieht, solltest du diesen auch in eine *unordered list* einbauen. Hast du das? Wenn nicht, blättere flugs ein paar Seiten zurück!



Die Linkbeschreibung wurde schon vorbereitet.

In meinem Beispiel werde ich auf die Adresse `www.flaeming-skate.de` verweisen. Es handelt sich um eine attraktiv gestaltete Info-Site zu „Europas Skate-Region südlich von Berlin“, einem Verbund unzähliger Skate-Routen, die mitten durch die liebeliche Brandenburger Natur führen.

Im eigentlichen Link musst du stets die volle Webadresse angeben, auch das `http://`!

Hier noch einmal die Grundsyntax:

```
<a href="URL">Linkbeschreibung</a>
```

Mit URL habe ich hier lediglich die komplette Webadresse gemeint. Als Linkbeschreibung wähle ich diesmal die Webadresse in Kurzform, also nur mit `www`.

### Der fertige Hyperlink

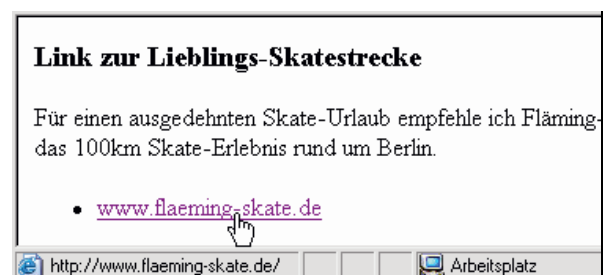
Hast du deinen Link schon gesetzt? Mein Querverweis sieht so aus, ich zeige ihn dir mitsamt der Liste. Die von mir fett hervorgehobene Passage ist die Linkbeschreibung, also das, was unterstrichen dargestellt wird (und bei dir schon da stand).

```
<ul>
```

```
<li><a href="http://www.flaeming-skate.de">www.flaeming-skate.de</a></li>
```

```
</ul>
```

Hast du deine Seite gespeichert und die Ansicht im Browser aktualisiert?



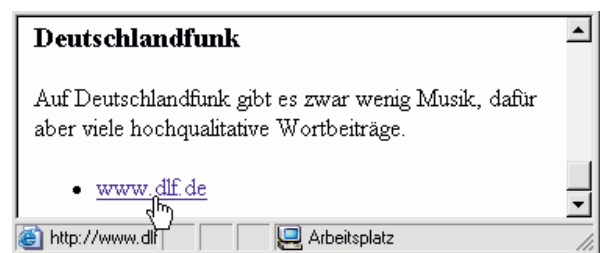
Die Link-Adresse erscheint auch in der Statuszeile.

Dann probiere den Hyperlink einfach aus. Dabei macht es gar nichts, dass du deine Seite noch nicht publiziert hast. Es funktioniert auch so.

### Übung macht den Meister!

Füge ruhig ein bis zwei weitere externe Links in deine Hobby-Seite ein. Vielleicht zu deinem Lieblings-Fußballclub?

Oder zu einer Suchmaschine? Oder zum Know-Ware-Verlag, der unter [www.knowware.de](http://www.knowware.de) zu finden ist? Oder zu [www.cmbasic.de](http://www.cmbasic.de)?



Radio ohne Hit-Mix und Millionen-Gewinne.

Das Ergebnis findest du unter `kapitel2/stand4`.

## Hyperlinks für Profis: Von Downloadlink bis Zielangabe

Gerade zum Thema Hyperlinks habe ich noch ein paar interessante Informationen für dich parat.

Wie wäre es mit einer praktischen Beschreibung? Oder mit Links, die sich automatisch in einem neuen Browserfenster öffnen? Machen wir ...

### ■ Hyperlinks mit Beschreibung: title

Du möchtest, dass dein Hyperlink beim Berühren mit einer kleinen Kurz-Info versehen wird? Dann verwende einfach das `title`-Attribut. Als Wert gibst du ein Wort oder eine Wortgruppe an, also das, was sich später in die angezeigte Information verwandeln soll.

Egal ob interner, externer oder E-Mail-Link, `title` zaubert in allen neueren Browsern einen informativen Text unter deinen Mauszeiger.

Hier als Beispiel der „aufgebesserte“ E-Mail-Link mit `title`-Attribut. Das `<a>`-Tag gehört mit allen Attributen auf eine Zeile oder es darf wie hier bei Leerzeichen umbrochen werden:

```
<a href="mailto:css@lexi.de"
title="Schreiben Sie mir!">
Feedback</a>
```

Das sieht dann so aus:



Wer kann dieser Aufforderung schon widerstehen? Allerdings ist der E-Mail-Link sehr „spamgefährdet“.

### ■ Übung macht den Meister!

Ergänze diese Aufforderung bei insgesamt allen drei Feedback-Links auf deiner Site. (Um die übrigen Links kümmern wir uns im Übungsteil.)

Vgl. mit den Dateien im Ordner `stand5`.

### ■ Links auf neuer Seite aufrufen

Egal ob interner oder externer Link: Nach Klick auf den unterstrichenen Text wird die alte Seite durch die neue Seite ersetzt.

Für unsere internen Links ist das auch gut so. Doch wie hältst du das mit den externen Verweisen? Da klickt der Besucher auf meine Empfehlung zu [www.flaeming-skate.de](http://www.flaeming-skate.de) und zack, schon ist er weg.

In diesem Fall wäre es gut, wenn meine Seite im Hintergrund geöffnet bliebe.

„Aufgeklärte Surfer“ kennen den Trick mit der rechten Maustaste. Wenn ich bei Google etwas gefunden habe, klicke ich mit der rechten Maustaste auf den entsprechenden Link. Dann wähle ich den Befehl *In neuem Fenster öffnen*.

### ■ Das target-Attribut im Überblick

Du kannst aber auch erzwingen, dass von vornherein ein neues Fenster geöffnet wird! Setze das Attribut `target="_blank"` zusätzlich in das `<a>`-Tag hinein.

Vergiss nicht den Unterstrich vor `blank`!

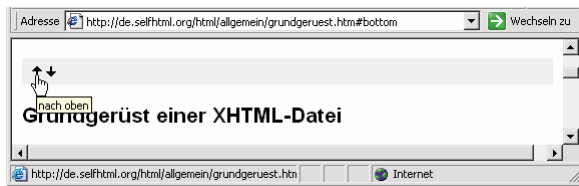
Neugierig, was es mit `target` auf sich hat? Insgesamt besitzt das Attribut folgende Werte:

Wert	Erläuterung	Bemerkung
<code>_self</code>	Seite im gleichen Fenster öffnen	Voreinstellung, daher normalerweise unnötig
<code>_blank</code>	Seite in neuem Fenster öffnen	erzeugt neues Browserfenster
<code>_top</code>	Seite wird auf jeden Fall im gesamten Browserfenster angezeigt	wichtig nur bei Frames (Rahmen)
<code>_parent</code>	Seite wird im gesamten Browserfenster bzw. im übergeordneten Frameset angezeigt	nur bei Frames wichtig
<code>Name</code>	Seite wird in den Rahmen mit diesem Namen geladen	nur bei Frames wichtig

### ■ Interne Anker und Verweise

Auch über interne Verweise sollten wir noch ein paar Worte verlieren. Denn manche Seiten sind so lang, dass man ziemlich weit rollen muss – nach unten und nach oben. Gerade auf solchen Seiten findest du dann oft Querverweise zum Seitenanfang, Seitenende oder direkt zu bestimmten Stellen. Häufig gibt es auch eine Art Inhaltsverzeichnis mit Sprüngen zu tiefer gelegenen Stellen auf der Seite.





Derartige Verweise auf interne Anker findest du beispielsweise auf <http://de.selfhtml.org> – hier in Form einer kleinen Pfeil-Grafik.

### ■ Internen Anker setzen

Wie fügst du nun solche anspringbaren Zielmarken ein? Nutze das `<a>`-Tag, aber diesmal nicht mit dem `href`-Attribut, sondern mit `name`. Vergib einen eindeutigen, nur einmal vorkommenden Namen (der keine Leerzeichen, Umlaute oder Sonderzeichen enthält). Schreibe beispielsweise am Anfang des Dokuments – gerne unterhalb des `BODY`-Tags Folgendes: `<a name="oben"></a>`, wobei oben der von mir frei gewählte Name ist.

Den Raum zwischen den `a`-Tags lässt du leer.

### ■ Verweis auf internen Anker setzen

Ein Verweis auf diesen internen Anker notierst du so, wichtig ist die Raute vor dem Namen:

```
<a href="#oben">nach oben</a>
```

Natürlich kannst du auch von einem anderen Dokument aus auf diesen internen Anker verweisen! Hänge die Raute und den Ankernamen an die Webadresse (URL) an. Beispielsweise so:

```
<a href="URL#Ankername"> ... </a>
```

Ganz am Schluss des Heftes folgt zu diesem Thema noch eine kurze Übung.

## Zu Download anbieten: Musik & PDF

### ■ Musik, Dateien, PDF: Mit Links!

Viele glauben gar nicht, wie einfach man Musik, Videos, Word-Dokumente, Excel-Tabellen, Access-Datenbanken, gepackte ZIP-Archive usw. auf seiner Homepage zum Download bereitstellen kann.

Egal um welchen Dateityp es sich handelt: Binde lediglich die entsprechende Datei wie gewohnt per Hyperlink ein. Es ist so einfach!

### ■ Beispiel MP3:

Angenommen deine MP3-Datei heißt `song.mp3` und liegt im gleichen Ordner wie die HTML-Datei. Dann bindest du den Song folgendermaßen ein:

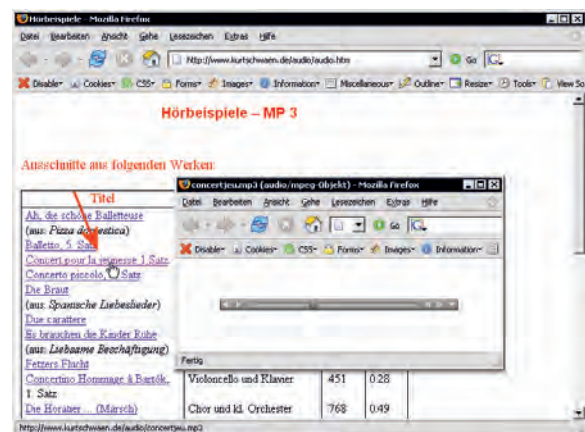
```
<a href="song.mp3">Lied 1</a>
```

Der Surfer klickt auf den unterstrichenen Text (hier [Lied 1](#)) und schon ... macht sein Rechner, was er kann. Folgende Grundregeln gelten:

- Bei Vorhandensein eines passenden „Plug-ins“ wird die Datei gleich angezeigt (bzw. abgespielt).
- Ansonsten wird die Datei per Dialogfenster zum Download angeboten.

Als „Plug-in“ für MP3 wird z. B. WinAmp oder beim Internet Explorer auch der Media Player betrachtet. Als „Plug-in“ für Word- bzw. Excel-Dokumente gelten Word bzw. Excel.

Leider kannst du als Homepage-Ersteller nicht wissen, welche Programme der Betrachter benutzt. Das ist der Haken bei der Geschichte!



Auf [www.kurtschwaen.de](http://www.kurtschwaen.de) bietet der Komponist Hörbeispiele zum Download bzw. zum Abhören an. Der HTML-Code ist ganz einfach!

### ■ Beispiel PDF:

Hast du schon die Möglichkeit entdeckt, auf [www.knowware.de](http://www.knowware.de) Vorschauseiten von allen anderen Heften abzurufen? Diese Vorschauseiten liegen im PDF-Format vor. PDF von Adobe ist ein Universalformat für die Anzeige gedruckter Medien am Bildschirm. Hier wird als „Anzeige-Plug-in“ übrigens der kostenlose Adobe Reader benötigt.

### ■ Trick für aufgeklärte Surfer

Als aufgeklärter Surfer speichere ich eine Datei in der Regel erst auf der Festplatte ab (und schaue dann, was ich damit machen kann).

Ich klicke dazu mit der rechten Maustaste auf den entsprechenden Link. Dann wähle ich den Befehl [Ziel speichern unter](#) (o. ä.). Und genau das empfiehlt dir KnowWare für den Download seiner PDF-Dateien.



## ✓ Übungsteil B: Interne und externe Hyperlinks

Du weißt jetzt, wie man:

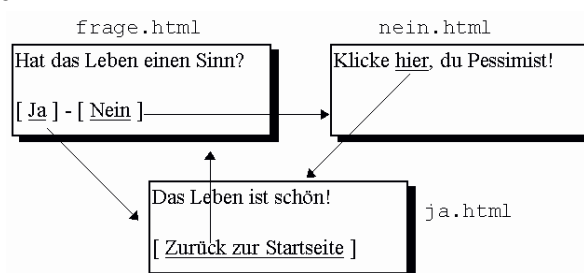
- interne Hyperlinks erstellt
- mit E-Mail-Links Feedback erhält
- externe Hyperlinks erstellt
- Links durch verschiedene Attribute interessant gestaltet



Für den Übungsteil A hast du dir ja schon unter deinem Ordner homepage einen Ordner namens aufgaben erstellt. Richte unter aufgaben nun folgende Unterordner ein: b1, b2, b3, b4 und b5. Lasse diese Ordner vorerst leer. Die Übungen setzen voraus, dass du das Heft bis hierhin durchgearbeitet hast. Die Profi-Übungen sind besonders schwer. Die Super-Profi-Übung ist nur etwas für wahre Spezis.

### ■ Übung B1: Sinn des Lebens: Seiten intern verlinken

Gehe in den Ordner b1. Erstelle hier drei einfache HTML-Dokumente. Nenne sie `frage.html`, `ja.html` und `nein.html`. Schreibe in der Datei `frage.html` lediglich den Absatz: *Hat das Leben einen Sinn?* Biete in einem neuen Absatz zwei Hyperlinks an. Die erste Linkbeschreibung soll *Ja* heißen, der Link führt auf die Seite `ja.html`. Als zweiten Link tippst du *Nein* und verweist auf `nein.html`.



In der `nein.html` schreibst du: *Klicke hier, du Pessimist!* Sorge dafür, dass das Wort *hier* zum Link wird, der zur Seite `ja.html` führt. Auf der Seite `ja.html` reicht der Satz: *Das Leben ist schön!* Füge außerdem einen Link zurück zur Startseite ein. Orientiere dich an der Skizze.

### ■ Übung B2: Wer sucht, der findet ... externe Links zu Suchmaschinen

Gehe nun in den Ordner b2. Erstelle ein HTML-Dokument namens `suchen.html`. Schreibe als Überschrift: *Wichtige Suchmaschinen* und füge in einer „unordered list“ (Aufzählung mit Bullets) Hyperlinks zu folgenden Suchmaschinen ein:

- [www.google.de](http://www.google.de)
- [www.yahoo.de](http://www.yahoo.de)
- [www.live.de](http://www.live.de)



Sorge dafür, dass alle drei Links in einem neuen Browserfenster geöffnet werden.

### ■ Übung B3: Linkbeschreibung als Grafik

Wie erwähnt, kannst du als Linkbeschreibung auch Grafiken nutzen. Dafür dient folgende Syntax:

```
<a href="URL"></a>
```

Erstelle als Übung das Dokument `bildlink.html`, und zwar im Ordner b3. Setze hier anhand einer selbst zu erstellenden Grafik einen Link auf eine beliebige Webseite. Im Beispiel nutze ich die Grafik `office2007.gif` und verweise auf <http://www.knowware.de>. Wie schaffst du es, den Rahmen um die Grafik zu unterdrücken? (Denke an das Attribut `border`, gib als Wert eine 0 an!)



### ■ Übung B4 (Profi): Musik liegt in der Luft: Links auf Media-Datei setzen

Suche auf deiner Festplatte nach je einer Datei im Wave-Format, im Midi-Format und wenn du hast, dann auch im MP3-Format. (Bei Wave und Midi handelt es sich um die zwei „klassischen“ Dateiformate für Klangdateien.)

Tipp: Spüre die Dateien mit der Suchfunktion von Windows auf, suche nach den Zeichenfolgen \*.wav, \*.mid bzw. \*.mp3!

#### Links auf Media-Dateien

[ [Wave-Datei](#) ] - [ [Midi-Song](#) ] - [ [MP3-Song](#) ]



Kopiere diese Dateien in den Ordner b4. Erstelle in diesem Ordner b4 eine neue HTML-Datei namens media.html. Notiere die Überschrift *Links auf Media-Dateien*. Biete alle diese Dateien als Download-Link an. Probiere die Links aus. Was passiert auf deinem Rechner?

### ■ Übung B5 (Super-Profi): Fremd-Datei im PDF-Format einbinden

KnowWare bietet auf [www.knowware.de](http://www.knowware.de) von jedem Heft 10-15 Seiten als Vorschau an. Dafür greift der Verlag, wie schon erwähnt, auf das universale PDF-Format zurück. Gehe zu [www.knowware.de](http://www.knowware.de) und suche nach diesem Download-Bereich. Informiere dich hier gegebenenfalls über das PDF-Format. Suche dir ein Heft heraus und merke dir die genaue Download-Adresse. (Dazu klickst du mit der rechten Maustaste auf diesen PDF-Link. Dann wählst du den Befehl **VERKNÜPFUNG KOPIEREN**.)

So, und nun erstellst du einen Querverweis auf genau diese eine PDF-Datei! Erzeuge dafür im Ordner b5 ein HTML-Dokument namens pdf.html. Wähle als Überschrift den Titel des von dir ausgesuchten Heftes. Füge einen Link ein, der heißt: *Vorschauseiten jetzt aufrufen*.

#### Bildbearbeitung für Einsteiger

[ [Vorschauseiten jetzt aufrufen](#) ] !

[http://www.knowware.dk/zip\\_pdf/bildbearbeitung\\_01-13.pdf](http://www.knowware.dk/zip_pdf/bildbearbeitung_01-13.pdf)

Wie du siehst, kannst du also auch auf Dateien verlinken, die auf anderen Servern liegen!

### ■ Übung B6: Attribut title als Linkbeschreibung einsetzen



Zurück zu unserem Hauptprojekt im Ordner eigenheim! Wir wollen unsere Hyperlinks noch etwas aufbessern!

Bis jetzt besitzt nur der Feedback-Link einen praktischen „Titel“.

Verpasse auch allen anderen Hyperlinks auf den Seiten index.html, beruf.html und hobby.html sinnvolle Kurzbeschreibungen mit Hilfe des title-Attributs.

### ■ Übung B7 (Profi): E-Mail-Link mit Betreff

Du möchtest gezielteres Feedback erhalten? Dich interessiert, von welcher Seite aus sich der Surfer bei dir meldet? Kein Problem! (Abgesehen davon, dass diese „Geschichte“ nicht ganz dem HTML-Standard entspricht. Dieser gestattet so etwas nicht, aber es funktioniert ja trotzdem.)

Gib allen drei E-Mail-Links (Feedback) im Ordner eigenheim einen selbsterklärenden Betreff mit auf den Weg. Für die index.html soll dieser *Startseite* lauten. Auf der beruf.html setzt du *Beruf* und auf der hobby.html *Interessen* ein.

Den letzten Stand des „eigenheim-Projekts“ findest du unter kapitel2/stand6.

## Kapitel 3: Gestaltung mit Cascading Style Sheets

Jetzt legen wir so richtig los! In dieser Lektion zeige ich dir, wie du dein Dokument gestaltest.

Zum Formatieren von Webseiten dient eine Sprache namens CSS. CSS ist die Abkürzung von Cascading Style Sheets, zu Deutsch: Kaskadierende Formatvorlagen.

Schriftart, -größe, Rahmen, Schattierung: Du wirst staunen, was man mit CSS alles machen kann. Zuerst lernst du übrigens Inline-CSS kennen. In späteren Lektionen lagern wir unsere Stile dann extern in separaten CSS-Dateien aus.

### Kopie des Ordners eigenheim

Üben ist wichtig! Zuerst experimentieren wir daher ausgiebig mit der `index.html`, unserer Startseite.

Damit kein „Schaden“ entsteht, richten wir uns dafür eine Kopie des gesamten Projekts von `eigenheim` ein!

Marschiere in den Windows-Explorer. Gehe hier folgendermaßen vor:

1. Klicke mit der rechten Maustaste auf den Projektordner `eigenheim`. Wähle den Befehl Kopieren. (Klar, du kannst auch **BEARBEITEN | KOPIEREN** oder `[Strg] + [C]` wählen.)
2. Markiere jetzt den übergeordneten Ordner `homepage` (durch kurzes Anklicken).
3. Wähle den **EINFÜGEN**-Befehl über das Kontext-Menü (oder **BEARBEITEN | EINFÜGEN** oder `[Strg] + [V]`).
4. Ein Ordner namens *Kopie von eigenheim* ist entstanden. Diesen benennst du um in `stiluebung`. (Zum Umbenennen drückst du die Funktionstaste `[F2]`, schreibst den neuen Namen über den alten und drückst `[Enter]`.)

Wir arbeiten vorerst im Ordner `stiluebung` weiter. Später werden wir den `eigenheim`-Ordner mit unseren „Stilkentnissen“ bearbeiten.

**Hinweis:** Meinen aktuellen Projektstand findest du weiterhin im `kapitel-` bzw. `stand-` Ordner, also unter `kapitelNr/standNr`.

### Die Gesamt-Schriftart ändern

Wir reden über CSS, die „Gestaltungs-Sprache“ für Webseiten. Wir reden über guten Stil, über „Style“. Und den sollten wir unseren Tags nun verpassen.

Zum Zuweisen von Stil-Eigenschaften benutzt man das `style`-Attribut!

Doch wo fügen wir dieses Attribut ein? Und welche Werte sind gültig? Fragen über Fragen!

### Das Kaskadenprinzip

Unterhalten wir uns zuerst über das Kaskadenprinzip, schließlich heißt es Cascading Style Sheets, Kaskadierende Formatvorlagen. Und tatsächlich: Wer hier an Wasserspiele denkt, liegt erst einmal richtig.

Fast wie bei einer Kaskade werden die Format-Eigenschaften bei CSS von oben nach unten „gespült“. Will heißen: Die festgelegten Formate vererben sich vom übergeordneten Element in der Regel auf untergeordnete Elemente.

Als HTML-kundiger Homepage-Freund weißt du, dass die eigentliche „Gestaltungs-Musik“ im `BODY` „spielt“. Und das in der Hierarchie des Dokuments am höchsten liegende Element ist tatsächlich der `<body>`-Container.

**Merke:** Die im `<body>`-Tag festgelegten Eigenschaften vererben sich (in der Regel) an alle untergeordneten Tags!

Machen wir ein erstes Experiment. Dazu verrate ich dir, wie du mit CSS die Schriftart festlegst.

### CSS-Eigenschaft font-family

Soviel schon vorweg: CSS besitzt eine zu HTML leicht abweichende Syntax.

Zum Ändern der Schriftart verwendest du die CSS-Eigenschaft `font-family`. Dann tippst du einen Doppelpunkt (kein `=`). Als nächstes empfehle ich der Übersichtlichkeit halber ein Leerzeichen. Als Wert gibst du den Namen der Schriftart an.

Gänsefüßchen sind bei „Einwort-Schriftartnamen“ wie *Arial* oder *Helvetica* nicht nötig!

### ■ Schriftart Arial

Du möchtest als Schriftart Arial festlegen? In CSS schreibst du folgende Zeile:

```
font-family: Arial
```

Allerdings kannst du nicht wissen, ob der Betrachter auf seinem Rechner wirklich Arial als Schriftart installiert hat. (Es gibt neben Windows noch viele andere Betriebssysteme.)

Gib stets eine Alternativschrift an!

Auf Macintosh-Systemen kennt man beispielsweise die Schriftart Helvetica. Sie ähnelt Arial.

### ■ Trenne Alternativschriften mit Komma

Du schreibst also:

```
font-family: Arial, Helvetica
```

Das Komma trennt verschiedene Werte voneinander. Wenn der Browser *Arial* findet, nutzt er diese Schriftart zur Anzeige. Wenn nicht, versucht er die nächste Schriftart, in diesem Fall *Helvetica*.

### ■ Die fünf Sammelbezeichnungen

Und wenn auf dem System keine Helvetica installiert ist? Weil es eine „Linux-Maschine“ ist?

Du solltest auf jeden Fall als kleinsten gemeinsamen Nenner eine „übergeordnete Alternativschrift“ angeben!

Es gibt fünf allgemeine Sammelbezeichnungen für solche übergeordneten Schriftfamilien.

Wert	Erläuterung
sans-serif	serifenlose, sachliche Schrift wie Arial, Helvetica usw. (Serifen sind kleine Füßchen an den Enden der Schriftzeichen.)
serif	Serifenschrift wie Times, Times New Roman usw.
cursive	allgemein für kursiv-geschwungene Schrift (Schriftart)
monospace	Schriftart mit fester Schrittweite (Courier)
fantasy	allgemeine „Fantasieschrift“

Schlussendlich sieht die Werte-Zeile für unsere font-family-Eigenschaft so aus:

```
Arial, Helvetica, sans-serif;
```

### ■ Am Ende steht ein Semikolon!

Wichtig: Nach Auflistung aller Eigenschafts-Werte setzt du zum Abschluss stets ein Semikolon!

### Ändern der Gesamtschriftart

Zu viel Gequatsche, Herr Hanke! Wie ändert man nun die Gesamt-Schriftart? Ganz einfach! Füge in das <body>-Tag die style-Eigenschaft ein. Notiere dahinter ein Gänsefüßchen-Paar:

```
<body style=" ">
```

Innerhalb des Gänsefüßchen-Paars kannst du so viele CSS-Eigenschaften angeben, wie du möchtest.



### Gesamt-Schriftart auf einen Schlag ändern.

Um die Schriftart zu ändern, schreibst du insgesamt also (möglichst auf eine Zeile):

```
<body style="font-family: Arial, Helvetica, sans-serif;">
```

Speichere dein Dokument und betrachte es in der Vorschau. Bisher klappt es perfekt: Das gesamte Dokument wird mit der Schriftart Arial (bzw. einer Alternativschrift) gestaltet. Vergleiche mit der Beispieldatei unter kapitel3/stand1.

## Die wichtigsten Stil-Eigenschaften zur Schriftgestaltung

Nach diesen ersten Hinweisen bist du sicher neugierig, wie man Text noch gestalten kann. Denn neben der Schriftart lassen sich natürlich auch Schriftgröße, Schriftgewicht (fett), Schriftstil (kursiv) usw. gestalten. Selbst um Farben und Hintergründe kümmern wir uns noch!

### Schrifteigenschaften

Wie du die Schriftart zuweist, weißt du schon. Eigentlich arbeitest du ohne Gänsefüßchen.

#### Schriftartnamen mit Leerzeichen

Sind die Namen der Schriftarten lang wie *Times New Roman* oder *Comic Sans MS*? Dann setze sie in ein Paar **einfache** Anführungszeichen!

```
font-family: 'Times New Roman', Times, serif;
```

Alles klar? Dann schauen wir uns nun die übrigen wichtigen Schrifteigenschaften an!

#### Schriftgröße in Punkt festlegen

Die Schriftgröße legst du über die Eigenschaft `font-size` fest. Als Maßeinheit empfehle ich dir vorerst die Maßeinheit Punkt (pt). So bestimmst du die Schriftgröße absolut wie in deiner Textverarbeitung. Für eine 12-Punkt-Schrift notierst du:

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

Du möchtest ein Element mit Arial in 12 Punkt gestalten? Dann schreibst du:

```
font-family: Arial, Helvetica, sans-serif; font-size: 12pt;
```

Bei diesem Beispiel siehst du übrigens, wie wichtig das Semikolon ist, da es zwei Eigenschafts-Wertebereiche voneinander trennt.

Du möchtest „halbe“ Punktwerte angeben? In der Textverarbeitung schreibst du 12,5. Da CSS aber der amerikanischen Notation folgt, musst du 12.5 tippen. (Punkt statt Komma!)

#### ■ Schriftgewicht fett

Du möchtest deine Zeichen fett formatieren? Dann nutze einfach die CSS-Eigenschaft `font-weight`. Als Wert gibst du `bold` an, *bold* steht für fett.

```
font-weight: bold;
```

#### ■ Schriftgewicht fett abschalten

Du kannst das Format fett aber auch abschalten. Wie? Na experimentiere doch einmal!

Gehe dafür an die Stelle deiner `index.html`, wo das Tag `<b>` (*bold* für fett) vorkommt. In meinem Beispiel habe ich den Namen so gestaltet. Schalte die Formatierung durch CSS ab:

```
<b style="font-weight: normal;">
Johann-Christian Hanke</b>
```

Mit den Eigenschaften des Style Sheets setzt du also die im `<b>`-Tag eingebaute Eigenschaft zurück. Nach diesem Experiment nimmst du den Stil aber wieder aus dem `<b>`-Tag heraus!

#### Schriftstil kursiv

Wie setzt du deine Schrift auf kursiv? Nutze die Eigenschaft `font-style`. Vergib als Wert *italic*. Die folgende Zeile stellt deinen Text *kursiv*:

```
font-style: italic;
```

Soll eine Passage fett und kursiv gleichzeitig dargestellt werden, schreibst du:

```
font-style: italic; font-weight: bold;
```

#### ■ Schriftstil kursiv abschalten

Wenn du den Schriftstil kursiv dagegen abschalten möchtest, schreibst du:

```
font-style: normal;
```

#### Format für Unterstreich

Zum Unterstreichen verwendest du die Eigenschaft: `text-decoration: underline;`



Weg mit der Unterstreich im Hyperlink.

Viel wichtiger als eine Unterstreich ist das Abschalten derselben. Nutze die Eigenschaft `text-decoration: none;`!

Setze es z.B. in das `<a>`-Tag für den Hyperlink: `style="text-decoration: none;"`

#### Fontvariante: KAPITÄLCHEN

Kennst du **KAPITÄLCHEN**, kleine Großbuchstaben? Zum Zuweisen dient die Anweisung:

```
font-variant: small-caps;
```

#### Ausrichtung und Zeilenabstand

Linksbündig? Rechtsbündig? Zentriert oder im Blocksatz? Auch diese Eigenschaften kannst du mit CSS wunderbar einrichten.

### Textausrichtung justieren

Zum Einrichten der vertikalen Textausrichtung nutzt du die CSS-Eigenschaft `text-align`. Die erlaubten Werte sind die gleichen wie beim `align`-Attribut von HTML. Es sind: `left` (linksbündig), `right` (rechtsbündig), `center` (zentriert) und `justify` (Blocksatz).

Diese Eigenschaft ist weniger wichtig. Wenn du `text-align` weglässt, wird weiterhin die voreingestellte Textausrichtung `left` verwendet.

### Zeilenabstand einstellen

Auch der vorgegebene Zeilenabstand (einzeilig) muss normalerweise nicht justiert werden. Du wünschst außer der Reihe einen eigenen Wert?



**Aufgelockert: Hier beträgt der Zeilenabstand 1.2.**

Zur Einstellung des Zeilenabstands nutzt du die CSS-Eigenschaft `line-height`. Als Wert kannst du eine Punkt-Angabe (14pt), eine Prozent-Angabe (150%) oder einen relativen Wert wie 1.5 einsetzen (entspricht 1,5-zeilig).

Mir persönlich gefällt die dritte Methode am besten, weil hier die Angabe ganz ohne Maßeinheit notiert wird. Für einen eineinhalbfachen Zeilenabstand schreibst du einfach:

```
line-height: 1.5;
```

Beachte unbedingt den Punkt statt des Kommas!

Soll der Zeilenabstand für das gesamte Dokument gelten? Notiere ihn direkt im `<body>`-Tag.

### Erstzeileneinzug einrichten

Vielleicht kennst du diesen Hervorhebungsstil aus Büchern oder Zeitschriften: Häufig wird die erste Zeile am Anfang eines Absatzes eingezogen. Das geht natürlich auch in CSS. Nutze dafür die CSS-Eigenschaft `text-indent`. Als Wert setzt du die gewünschte Angabe ein, entweder in Pixeln, Zentimetern oder Punkt.

### Schön, dass Sie da sind!

Guten Tag. Schön, dass Sie sich für meine Seite interessieren. Ich heiße **Johann-Christian Hanke**, bin 34 Jahre alt und wohne in Berlin. Auf den folgenden Seiten möchte ich mich Ihnen vorstellen.

**Die erste Zeile wird um 15 Pixel eingezogen.**

Um in einem Absatz einen Erstzeileneinzug von 15 Pixeln zu erzielen, notierst du:

```
<p style="text-indent: 15px;">
```

### Die Maßeinheiten von CSS

Hoppla, da erwähne ich als Maßeinheit Punkt. Dann wieder Pixel. Jetzt kommen plötzlich auch Zentimeter ins Spiel. Warum solch ein Chaos?

Nun, in CSS sind viele Maßeinheiten erlaubt:

Endung	Erläuterung
px	Pixel
mm	Millimeter
cm	Zentimeter
in	Zoll, Endung <i>in</i> von inch, 1 Zoll sind 2,54 cm
pt	typographischer Punkt, 1pt entspricht 1/72 inch
pc	typographischer Pica-Punkt, 1pc entspricht 12pt
em	relativ zur vom Nutzer eingestellten Schriftgröße, bezieht sich auf die Höhe des Buchstaben 'M' – z. B. font-size: 1.3em
%	Prozent: relativ in Bezug auf übergeordnetes Element – z. B. font-size: 130%

Ich bevorzuge die Verwendung von Pixeln (px) oder Punkt (pt). Zentimeter oder Millimeter sind für den Bildschirm wenig geeignet, da jeder Schirm andere Größen und Pixeldichten besitzt.

### Der Glaubenskrieg um die Maßeinheiten

Zur Verwendung der richtigen Maßeinheiten tobt ein richtiger Glaubenskrieg unter den Gestaltern. In vielen Anleitungen wirst du auf relative Einheiten wie % oder em eingeschworen. Als ein Argument gilt dabei, dass zumindest im Internet Explorer nur Texte mit relativen Einheiten in der Größe verändert, also skaliert werden könnten. (Tipp: Das „Skalieren“ gelingt durch Gedrückthalten der Taste **Strg** und Drehen des Mausekursors.)

Doch spätestens mit Erscheinen des Internet Explorers 7 zählt dieses Argument nicht mehr. Außerdem finde ich die Verwendung von % oder em viel schwieriger. Ich bleibe bei diesem Beispiel beim Punkt (der intern vom Browser in Pixel umgerechnet wird.) Später jedoch – bei unserem großen Hauptbeispiel – steigen wir auf em um.



## Vordergrund- und Hintergrundfarbe einstellen

### Farben in HTML und CSS

Farben machen das Leben erst so richtig bunt und schön. Schauen wir uns nun an, wie man Schrift sowie Vorder- und Hintergründe einfärbt.

#### Die Farbnamen

In HTML und CSS stehen dir bis zu 16,7 Mio. Farben zur Verfügung, wie auf deinem PC.

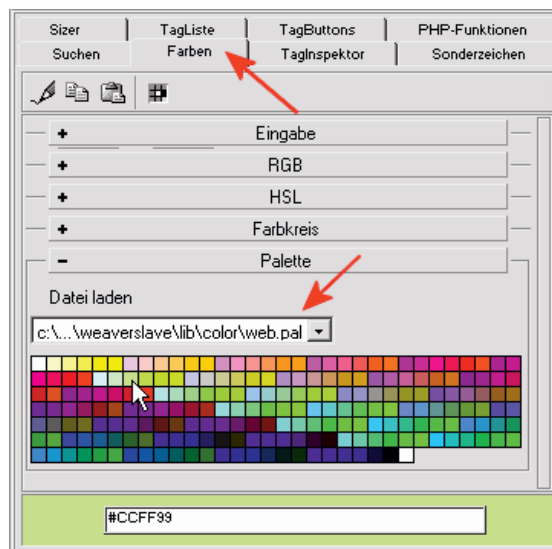
Jede Farbe wird durch einen hexadezimalen Farbcode bestimmt. Dieser beschreibt die Zusammensetzung aus den drei Grundfarben Rot, Grün und Blau. Die Syntax lautet #rrggbb. Ein Rot ist #ff0000, ein Grün #00ff00 und das reine Blau #0000ff. Weiß (alle Farbwerte maximal!) wird durch #ffffff und Schwarz (kein Farbwert!) durch #000000 dargestellt. Die einzelnen Farbwerte werden von 0 bis 9 und dann von a bis f abgestuft.

In der folgenden Tabelle liste ich dir die Farbnamen und Hex-Werte der 16 Grundfarben auf:

engl. Name	dt. Name	Hexadezimalwert
black	schwarz	#000000
silver	hellgrau	#c0c0c0
gray	grau	#808080
white	weiß	#ffffff
maroon	kastanienbraun	#800000
red	rot	#ff0000
purple	lila	#800080
fuchsia	helllila	#ff00ff
green	dunkelgrün	#008000
lime	hellgrün	#00ff00
olive	olivgrün	#808000
yellow	gelb	#ffff00
navy	dunkelblau	#000080
blue	blau	#0000ff
teal	blaugrün	#008080
aqua	himmelblau	#00ffff

Ich persönlich greife gerne auf die so genannte Browser-sichere Farbpalette zurück. Es handelt sich um die 216 Farben, die auf allen Browsern und unter allen Betriebssystemen ähnlich dargestellt werden. Wie stellst du die ein?

Im Weaverslave wählst du einfach **FENSTER | FARBEN** und gehst in den Bereich *Palette*. Achte darauf, dass im Feld *Datei laden* die Auswahl *web.pa1* eingestellt ist. Schon siehst du nur diese 216 Farben.



**Bereich „Farben“ im Weaverslave: Websichere Farben erkennst du daran, dass sie nur aus Kombinationen der Werte 00, 33, 66, 99, cc oder ff bestehen.**

Und wie weist du den jeweiligen Elementen nun die gewünschte Farbe zu? Ganz einfach! Auch das gelingt mit den entsprechenden CSS-Eigenschaften der Style Sheets.

#### Eigenschaft color: Farbe zuweisen

Zum „Einfärben“ von Text (oder anderen Elementen) benutzt du die Eigenschaft `color`.

Gib als Wert einfach einen der erlaubten Farbnamen oder den entsprechenden Hex-Wert an.

Wenn du dein gesamtes Dokument mit blauer Schrift versehen willst, notierst du einfach:

```
<body style="color: blue;">
```

#### Einfach: Hintergrundfarbe einstellen

Im Zusammenhang mit `color` muss ich natürlich auch `background-color` erwähnen. Denn mit dieser Eigenschaft kannst du die Hintergrundfarbe einstellen. Wie wäre es mit einem zartgelben „Farbteppich“ für das gesamte Dokument?

Notiere in der Stilanweisung im `<body>`-Tag:

```
background-color: #ffff99;
```

Wenn du eine Hintergrundfarbe wählst: Achte darauf, dass diese von der Vordergrundfarbe abweicht. Sonst kann der Betrachter deine Texte womöglich nicht mehr lesen!



## Wichtige Gesetzmäßigkeiten von Style Sheets

Zugegeben, in diesem Kapitel haben wir uns recht viel mit Theorie beschäftigt. Trotzdem noch ein paar wichtige Grundlagen und Gesetzmäßigkeiten von CSS. Ich fasse mich kurz – versprochen!

### ■ Vererbungs- oder Kaskadenprinzip

Wie schon erwähnt: CSS heißt Kaskadierende Formatvorlagen. Eigenschaften werden – von Ausnahmen abgesehen – vom übergeordneten Element an die jeweils tiefer liegenden Elemente vererbt.

Das in der Hierarchie am höchsten liegende Element ist `<body>`. Hier einmal festgelegte Eigenschaften vererben sich an untergeordnete Überschriften bzw. Absätze.

Du definierst im `<body>`-Tag beispielsweise eine Schriftfarbe?

```
<body style="color: blue;">
```

Dann wird diese Eigenschaft vorerst auch von allen Überschriften, Absätzen und selbst von `<b></b>` oder `<i></i>` innerhalb eines Absatzes übernommen werden. Probiere es aus! Es gilt das Vererbungsprinzip. Ist das nicht prima?

### ■ Elementeigenschaften haben Vorrang

Allerdings genießen die Eigenschaften eines entsprechenden Elements Vorrang vor den im übergeordneten Element festgelegten Eigenschaften.

Du definierst nun in einem Absatz eine andere Farbe? Bitte sehr, kein Problem! Schreibe beispielsweise vor *Guten Tag*... `<p style="color: red;">`

Auch das `<b>`-Tag innerhalb dieses Absatzes erbt nun diese „neue Farbe“ aus dem übergeordneten Absatz. Solange, bis du hier wieder eine individuelle Einstellung vornimmst.



Genießt Vorrang: 'Comic Sans MS' in `<h1>`.

Das gleiche gilt auch für andere Eigenschaften. Wenn du in einem Absatz eine Schriftgröße von 16pt festgelegt hast (`font-size: 16pt;`), ist diese Eigenschaft „stärker“ als beispielsweise eine Anweisung wie `font-size: 12pt;` im `<body>`-Tag!

Merke: Eigenschafts-Werte direkt im Element überschreiben die Werte der gleichen Eigenschaften aus übergeordneten CSS-Anweisungen.

Besonders wichtig: Auch die in den HTML-Tags „eingebauten“ Eigenschaften (größere Schriftgröße und Abstand bei Überschriften, Fettformatierung bei `<b>`, Kursivstellung bei `<i>`, usw. usw.) genießen Vorrang. Diese Eigenschaften werden aber nicht durch HTML festgelegt, sondern sie stammen aus dem eingebauten Standard-Style Sheet des Browsers. Denn wenn du keine eigenen Eigenschaften festlegst, greift der Browser selbsttätig auf voreingestellte Abstände und Größen zurück.

Was folgt aus diesen wichtigen Gesetzmäßigkeiten? Lege allgemeine Eigenschaften für alle Tags so weit oben in der Hierarchie fest wie möglich, praktisch als „Default-Wert“. Wenn du individuelle Werte zuweisen willst, begibst du dich dann direkt in das jeweilige Tag.

**Ausnahmen:** Einige Eigenschaften wie Hintergrundfarbe (`background-color`) oder die noch zu besprechenden Rahmen und Ränder werden nicht vererbt! Das ist auch gut so.

### ■ Wie „denkt“ der Browser?

Der Browser erstellt beim Interpretieren von CSS einen internen Stammbau, die so genannte Dokumentstruktur. Wie sieht die aus? Wähle im Firefox doch einmal **EXTRAS | DOM INSPECTOR** und schaue es dir an. Hier siehst du alle übergeordneten (Eltern-) und untergeordneten (Kindelemente).

Anhand dieses Baumes erkennt der Browser, ob eine Eigenschaft vom Elternelement vererbt wird oder ob du der Eigenschaft direkt einen Wert zugewiesen hast. Wenn beides nicht zutrifft, verwendet er die schon erwähnten fest eingestellten Standardwerte, im Zweifelsfall – beispielsweise wenn du das Angeben der `font`-Eigenschaft vergisst – eben die ungeliebte Schriftart *Times New Roman*.

## Style Sheets an zentraler Stelle im HEAD notieren

Die Sache mit den Style Sheets ist ja ganz schön und gut. Gewiss. Doch wenn du alle Überschriften in deinem Dokument besonders gestalten möchtest? Hast du Lust, in jeder Headline einzeln alle Stileigenschaften anzugeben? Nein?

Und tatsächlich scheint die bisher besprochene Methode mit den so genannten Inline-Stilen alles andere als elegant zu sein. Tatsache!

Höchste Zeit also, eine weitere Notationsform kennen zu lernen! Wechseln wir von der „Inline-Regional-Liga“ in die „HEAD-Oberliga“!

### Style Sheets im HEAD notieren

Mach einfach mit! Steckst du immer noch im gesonderten „Test-Ordner“ stiluebung? Wunderbar! Dann experimentieren wir diesmal mit der Datei `beruf.html`! Rufe sie auf.

Um optimal mit Style Sheets arbeiten zu können, gibt man sie an zentraler Stelle im Kopfbereich des Dokuments an.

Schau in den Kopfbereich des Dokuments, und zwar in den Bereich zwischen `<head></head>`.

Man verwendet den Platz zwischen dem ausschaltenden `</title>`- und dem `</head>`-Tag.

Nun denn, legen wir einfach los!

1. Gehe im Quelltext in den Bereich zwischen dem Meta-Tag für den Zeichensatz und vor `</head>`. Erzeuge ein bis zwei Leerzeilen.
2. Notiere zuerst das einleitende `<style>`-Tag. Gib als Eigenschaft `type="text/css"` mit an. Schreibe also:  

```
<style type="text/css">
```
3. Du hast das korrespondierende Ausschalt-Tag sofort gesetzt? Super! (Gut, dass du diese Regel beherzigst, denn vergessene Ausschalt-Tags sind einer der häufigsten Fehler!) Auch ich notiere nun zwei Zeilen tiefer:  

```
</style>
```
4. Insgesamt sieht der Style Sheet-Bereich bisher folgendermaßen aus:  

```
<style type="text/css">
  Platz für Stilanweisungen
</style>
```

Nun können wir fröhlich mit unseren Stilanweisungen loslegen! Dafür kannst du problemlos die eben besprochenen Eigenschaften und Werte der Inline-Stile benutzen. Hier ändert sich nichts. Doch wie bezieht man sich auf die einzelnen Elemente? Und wie sieht die Syntax aus?

### Das Tag wird zum Selektor!

Nehmen wir ein Beispiel: Ich zeige dir, wie du dich auf das Element `<body>` beziehst. Mach mit, ergänze den bisher noch leeren Style Sheet-Bereich im HEAD!

1. Du willst dich auf ein bestimmtes Tag beziehen? Dann notierst du dieses einfach ganz zu Beginn. Schreibe beispielsweise:

```
body
```

Lasse die spitzen Klammern auf jeden Fall weg! Man notiert an dieser Stelle das „nackige“ Tag und nennt es Selektor.

2. Dahinter setzt du der Übersichtlichkeit halber ein Leerzeichen und notierst eine öffnende geschweifte Klammer:  

```
body {
```
3. Drücke ein- bis zweimal auf `[Enter]`, um Platz für die eigentlichen Stilanweisungen zu schaffen. Setze die schließende geschweifte Klammer am besten in eine eigene Zeile.  

```
body {
  Eigenschaften und Werte von CSS
}
```
4. Innerhalb dieses Klammernpaares notierst du einfach die schon bekannten Eigenschaften und Werte von CSS. Soll die Schriftfarbe des gesamten Dokuments blau werden? Schreibe:  

```
body {
  color: blue;
}
```
5. Möchtest du zusätzlich die Schriftart verändern? Dann setze die entsprechende Eigenschaft in eine eigene Zeile!

Die Reihenfolge der Anweisungen ist in diesem Fall egal. Du kannst auch zuerst die Schriftart und dann die Farbe notieren!

```
body {
  color: blue;
  font-family: Arial, sans-serif;
}
```

### ■ Der Aufbau einer Stilregel

Die Gesamtheit der Anweisungen in geschweiften Klammern nennt man **rule**, Regel. Die einzelnen Zeilen innerhalb des Klammersblocks werden dagegen als Deklarationen bezeichnet. Eine Stilregel besteht also aus einer oder mehreren Deklarationen, die durch eine geschweifte Klammer zusammengefasst und mit einem Selektor versehen werden. Eine Deklaration wiederum besteht aus Eigenschaft (z.B. `color`), dem Zuweisungszeichen `:` und dem dazugehörigen Wert (z.B. `blue`). Wenn es mehrere Werte sind, werden diese durch Kommas getrennt. Jede Deklaration wird mit einem Semikolon abgeschlossen, dem „Zeilenendezeichen“.

Diese merkwürdige CSS-Schreibweise wurde tlw. von Programmiersprachen wie C oder C++ abgeschaut. Auch in JavaScript werden zusammenhängende Anweisungen mit Hilfe der geschweiften Klammern „als Block“ zusammengefasst. Am Ende jeder Zeile (hier am Ende eines Eigenschafts-Wertepaares) steht ein Semikolon.

### Mehrere Selektoren gruppieren

Du ahnst sicher schon, wie flexibel die Arbeit mit Selektoren und Stilanweisungen ist. Statt jede Überschrift einzeln zu bearbeiten, erstellst du eine eigene Regel und schreibst im Stilbereich:

```
h1 {
  font-family: Verdana, sans-serif;
}
```

Aber auch damit sind die Möglichkeiten der Stile noch nicht erschöpft.

Oft möchtest du, dass mehrere Tags die gleichen Eigenschaften übernehmen. Alle Überschriften der ersten und zweiten Ebene sollen beispielsweise die Schriftart *Verdana* zugewiesen bekommen?

Liste die jeweiligen Selektoren nur durch Komma getrennt auf. Gruppier sie!

Im Beispiel sieht die Regel folgendermaßen aus:

```
h1, h2 {
  font-family: Verdana, sans-serif;
}
```

Natürlich kannst du auch mehr als zwei Fliegen bzw. Elemente „mit einer Klappe“ schlagen. Grup-

piere drei, vier, fünf oder auch noch mehr Selektoren. Das Komma steht für „gleichberechtigte“ Nebenordnung.

### Selektoren ineinander verschachteln

Dass CSS eine zutiefst verschachtelte „Kiste“ ist, hast du sicher längst gemerkt. Apropos Kiste. Es ist durchaus hilfreich, sich die einzelnen Elemente als „Kisten“ vorzustellen. Das übergeordnete Element BODY ist die „Eltern-Kiste“, die einzelnen Überschriften, Absätze sind die „Kind-Kisten“. Diese stecken in der Eltern-Kiste drin. Falls du innerhalb eines Absatzes `<p>` das Tag `<b>` (bold, fett) verwendest, ist das die Kind-Kiste der Elternbox `<p>`.

Aber ich wollte dich nicht mit „Kistenschieberei“ langweilen. Ich verrate dir nun, wie du verschachtelte Tags individuell gestalten kannst!

### ■ Fett ist nicht gleich fett

Bleiben wir doch gleich beim Beispiel fett, also beim Tag `<b>`. Du möchtest, dass fett formatierte Passagen innerhalb von Überschriften schwarz und kursiv und in Absätzen dagegen nur rot eingefärbt werden?

Dann nutze folgende Syntax:

```
Oberselektor Unterselektor {
  Eigenschaften;
}
```

Du trennst Ober- und Unterselektor nur durch ein Leerzeichen.

Im Beispiel schreibst du also:

```
h1 b {
  color: black;
  font-style: italic;
}
p b {
  color: red;
}
```



Zweimal `<b>`, aber mit zwei verschiedenen Formaten.

Beachte, dass diese Technik nur mit Elementen funktioniert, die ineinander verschachtelt werden können. Setze ein Leerzeichen statt des Kommas, denn das Leerzeichen steht für Unterordnung.

### Kommentare in Style Sheets

Verliere nicht die Übersicht! Denn bald werden deine CSS-Blöcke lang und länger. Damit du später noch durch deine Stilanweisungen durchsteigst, empfehle ich dir auch in CSS die Arbeit mit Kommentaren.

Kommentare werden vom Browser nicht interpretiert, sie dienen einfach zu deiner Hilfe.

Wie du in HTML kommentierst, weißt du ja schon, du nutzt die Anweisung `<!-- -->`.

In CSS dagegen gilt eine ganz andere Schreibweise. Setze deine Kommentare innerhalb der Zeichen `/*` und `*/`. Hier ein komplettes Beispiel:

```
h1, h2 {  
    /* Schriftart definieren: */  
    font-family: Verdana, sans-serif;  
}
```

Diese Syntax für Kommentare wurde ebenfalls aus C++ oder Java abgeschaut und findet übrigens auch in JavaScript Verwendung.

Die aus Programmiersprachen bekannte Kommentarsyntax mit doppelten Slashes `//` oder mit einer Raute `#` ist in CSS leider nicht gestattet.

### Die Kompaktschreibweise

Bei der bisher gewählten Schreibweise bekommt jedes Eigenschafts-Werte-Paar eine eigene Zeile. Jede Zeile wird durch Semikolon abgeschlossen.

Diese Schreibweise ist sehr übersichtlich!

Aus Platzgründen wirst du in der Praxis oft die so genannte „Kompaktschreibweise“ vorfinden. Dabei lässt man die vielen Zeilenumbrüche einfach weg. Man notiert alle Anweisungen praktisch hintereinander auf einer langen Zeile. Praktisch eine Regel pro Zeile.

So sieht die generelle Syntax aus:

```
Selektor { Eigenschaften und Werte; }
```

Hier ein (kurzes) Beispiel:

```
body { font-size: 12pt; color: blue; }
```

Für den Browser ist es egal, welche Schreibweise du wählst. Schließlich dient das Semikolon als „Kennzeichen“ für das Zeilenende.

Ich empfehle eher die „Langfassung“. Sie ist sehr viel übersichtlicher. Später kannst du gerne auf die Kompaktschreibweise umsteigen. Das hat auch Vorteile, man spart Zeilen im Editor.

Da Profis gerne überall da kürzen, wo es geht, gibt es sogar eine Kurzschreibweise für Schrifteigenschaften. Wenn du dich damit belasten willst, lies das Kapitel „Schrift-Steno“ auf Seite 34. Bei unserem Hauptprojekt im zweiten Teil des Heftes werden wir auf die Kurzschreibweise zurückgreifen.

## Zusatzwissen für Profis: Schriftgestaltung und Listen

### Kurzübersicht der CSS-Eigenschaften

Du wünschst eine kurze und kompakte Übersicht aller bisher behandelten und auch neuer Schrifteigenschaften und -werte? Bitte:

Eigenschaft	Werte	Erläuterung
font-family	Schriftname	legt konkrete Schriftart fest
	serif	Serifenschrift allgemein
	sans-serif	serifenlose Schrift
	cursive	kursiv-geschwungene Schrift (Schreibschrift)
	monospace	Schrift mit fester Schrittweite
	fantasy	Phantasieschrift
font-size	pt, cm usw.	bestimmt Schriftgröße
font-style	normal	ohne Eigenschaft
	italic	kursiv
	oblique	Hohlschrift
font-weight	100 bis 900	„Fettfaktor“ (gemessen in Hunderterschritten)
	normal	entspricht Wert 400
	bold	entspricht Wert 700
font-variant	normal	ohne Eigenschaft
	small-caps	Kapitälchen (kleine Großbuchstaben)
word-spacing	pt, px, cm ...	Wortzwischenraum
letter-spacing	pt, px, cm ...	Buchstabenzwischenraum (für Sperrschrift)
text-transform	none	bleibt unverändert
	uppercase	Großbuchstaben
	lowercase	Kleinbuchstaben
	capitalize	erster Buchst. im Wort groß
text-decoration	none	schaltet Hervorhebung ab
	underline	unterstrichen
	overline	Strich darüber
	line-through	durchgestrichen
	blink	blinkender Text
text-align	left	Ausrichtung linksbündig
	right	Ausrichtung rechtsbündig
	center	zentriert
	justify	Blocksatz
line-height	1.2, 120%, 12pt ...	Zeilenhöhe (Zeilenabstand)
text-indent	px, pt usw.	Erstzeileneinzug
color	Farbname, Hex-Wert	Textfarbe
background-color	Farbname, Hex-Wert	Hintergrundfarbe

### Listen attraktiv gestalten

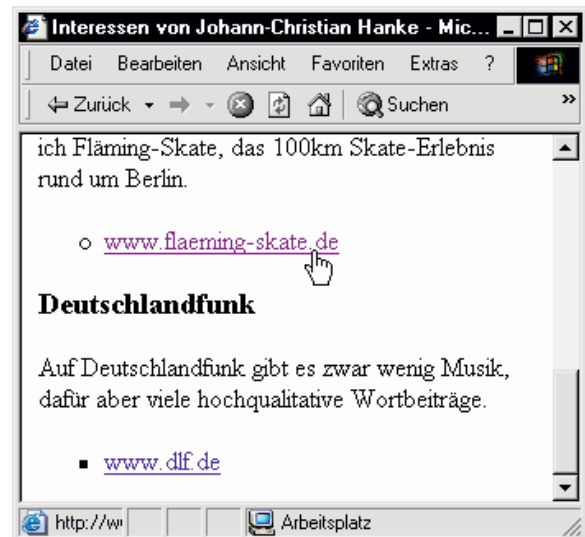
Auch Aufzählungen und Nummerierungen kannst du mit den entsprechenden Eigenschaften und Werten gestalten. Wir unterscheiden zum einen zwischen ungeordneten Listen (einfachen Aufzählungen) und den „ordered lists“ (Nummerierungen).

#### ■ Aufzählungen gestalten (UL)

Normalerweise besitzen einfache Aufzählungen per Voreinstellung Bullet-Zeichen, die kleinen gefüllten Kullern.

- Das ist eine Beispielaufzählung

Du kannst mit Style Sheets aber auch hohle Kreise oder gefüllte Rechtecke einstellen. Sogar das komplette Abschalten der Aufzählungszeichen ist möglich!



**Aufzählung: Es müssen nicht immer gefüllte Kreise sein.**

Dahinter verbergen sich die Werte der Eigenschaft `list-style-type`.

Die Eigenschaft kann folgende Werte annehmen:

Wert	Erläuterung
disc	Voreinstellung (gefüllter Kreis)
circle	hohler Kreis
square	gefülltes Quadrat
none	kein Aufzählungszeichen

Notiere die Angaben im übergeordneten Tag für die Aufzählung, wähle als Selektor also `ul`.

Möchtest du eine Liste mit gefüllten Quadraten versehen? Dann notiere folgende Regel:

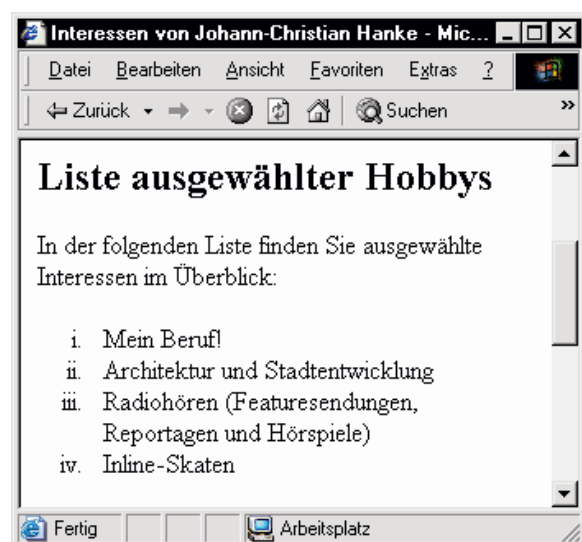
```
ul {
  list-style-type: square;
}
```

Auf Seite 51 zeige ich dir, wie du mit `list-style-image` eine kleine Grafik als Aufzählungszeichen einbindest!

### ■ Nummerierungen gestalten (OL)

Noch reichhaltiger ist das Angebot für Nummerierungen bei *ordered lists* (`<ol></ol>`).

Die einfache Durchnummerierung mit Dezimalzahlen genügt dir nicht? Bediene dich! Nimm Groß- oder Kleinbuchstaben oder große bzw. kleine römische Ziffern.



**Römische Zahlen:** `ol { list-style-type: lower-roman; }.`

Verwende auch hier die Eigenschaft `list-style-type`! Als Selektor wählst du `ol`. Folgende Werte sind erlaubt:

Werte	Erläuterung
decimal	Voreinstellung (1, 2, 3, ...)
lower-alpha	Kleinbuchstaben (a, b, c, ...)
upper-alpha	Großbuchstaben (A, B, C, ...)
lower-roman	kleine römische Zahlen (i, ii, iii, ...)
upper-roman	große römische Zahlen (I, II, III, ...)

### Schrift-Steno mit font

Kennst du Kurzschrift, die Stenografie? Nun, so ein „Steno“ gibt es auch für CSS. Nutze die Eigenschaft `font`. Hier zeige ich dir die Kurznotation für Schrift-eigenschaften.

Die Syntax lautet:

```
font: Pos1 Pos2 Pos3 Pos4 Pos5;
```

`Pos` steht als Platzhalter für ein Element einer bestimmten „font-Eigenschaften“. Mit der Zahl (z.B. `Pos3`) meine ich die entsprechende Position in der Aufreihung! Hier die Übersicht:

Position	Eigenschafts-Wert angeben von:	Pflicht?
Pos1	<code>font-style</code> (Schriftstil wie <i>italic</i> )	nein
Pos2	<code>font-variant</code> (Schriftvariante wie <code>SMALL-CAPS</code> )	nein
Pos3	<code>font-weight</code> (Schriftgewicht bzw. „Fettfaktor“ wie <b>bold</b> )	nein
Pos4	<code>font-size</code> (Schriftgröße wie 12pt), <b>kann</b> mit <code>line-height</code> kombiniert werden, und zwar mit Slash (/) in folgender Syntax: 12pt/1.2 oder 12pt/14pt oder 12pt/120%	ja
Pos5	<code>font-family</code> (Schriftart)	ja

Gib also zwei oder mehrere Eigenschaften an, mindestens `font-size` und `font-family`! Trenne die Werte durch Leerzeichen. Beachte die Reihenfolge: Notiere die ersten drei Attributwerte (falls gewünscht) stets zu Beginn. Wichtig: Setze die Angabe zur Schriftgröße und Schriftart an vorletzte bzw. letzte Position!

### ■ Beispiel Absatzformatierung

Grau ist alle Theorie, deshalb nehmen wir ein Beispiel. Du willst allen Absätzen `<p>` die Schriftart Arial in 12 Punkt zuweisen? Notiere:

```
p {
  font: 12pt Arial, sans-serif;
}
```

Das ist die vorgeschriebene „Mindest-Syntax“. Möchtest du zusätzlich die Eigenschaft *fett* und die Zeilenhöhe *1,2-zeilig* darstellen? Schreibe:

```
p {
  font: bold 12pt/1.2 Arial, sans-serif;
}
```

Den bisherigen Stand findest du unter `kapitel3/stand5`.

## ✓ Übungsteil C: Erste Schritte mit CSS

Du weißt jetzt, wie man:

- Schrifteigenschaften per `style`-Eigenschaft verändert
- Alternativschriften angibt und das Kaskadenprinzip (Vererbung) nutzt
- Weitere Eigenschaften für Vorder- und Hintergrundfarben einsetzt
- Style Sheets im Kopfbereich eines HTML-Dokuments notiert



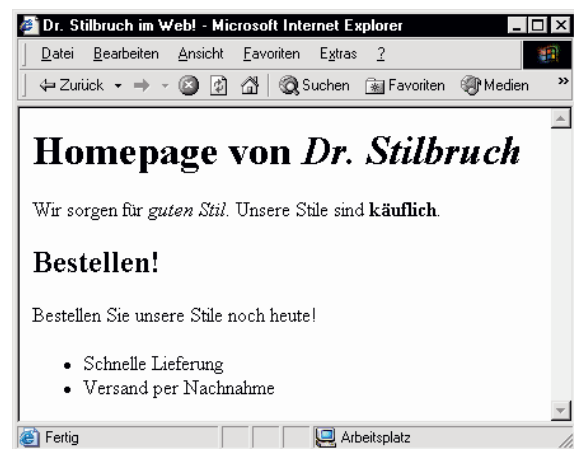
Für den Übungsteil A hattest du dir ja schon unter `homepage` einen Ordner namens `aufgaben` erstellt. Richte nun folgende Unterordner ein: `c1`, `c2`, `c3` und `c4`. Lasse diese Ordner vorerst leer.

### ■ Übung C1 (Wiederholung): HTML-Seite erstellen, Datei kopieren

Sind deine HTML-Kenntnisse noch frisch? Versuche einmal, das nebenstehende Projekt vollständig in HTML umzusetzen, und zwar noch ohne CSS. Speichere die Datei unter dem Namen `stil.html` im Ordner `c1`. Wir brauchen die Datei gleich zum Üben!

*Tipp:* Vergiss nicht den Seitentitel. Außerdem gibt es zwei Überschriftsebenen, eine `<h1></h1>` und eine `<h2></h2>`. In der `H1` steckt das Tag-Paar `<i></i>` (*Dr. Stilbruch*). Der erste der zwei Absätze enthält neben kursiv (*guten Stil*) auch eine Fett-Formatierung (**käuflich**). Die Aufzählung ist eine *unordered list*.

Fertig? Alles korrekt? (Vergleiche im Zweifelsfall mit dem Quellcode des downloadbaren Ordners `c1`.)



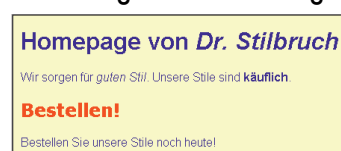
Dann kopiere dieses Muster-Dokument vom Ordner `c1` nun in die Ordner `c2`, `c3` und `c4`!

### ■ Übung C2a: Dokument mit Inline-CSS formatieren

Gehe nun in den Ordner `c2`. Dort liegt ja jetzt (siehe Übung C1) eine Kopie von `stil.html`. Gestalte das gesamte Dokument per Inline-CSS mit der Schriftart *Arial*. Gib als Alternativschrift *Helvetica* und eine allgemeine serifenlose Schrift an.

Wähle als Schriftfarbe für das Dokument insgesamt ein dunkles Blau. Die Hintergrundfarbe soll ein zartes Gelb sein. Nimm ruhig Browser-sichere Farben! Dazu wählst du im Weaverslave einfach **FENSTER | FARBEN**. Rechts erscheint das Register *Farben*. Rolle herunter bis zum Bereich *Palette*. Klicke auf die Schaltfläche **DATTE LADEN** und lade die Datei `web.pa1`. Nun werden dir nur die websicheren Farben angezeigt.

### ■ Übung C2b: Lokale Eigenschaften definieren



Die Überschrift 2 (`<h2>Bestellen!</h2>`) soll mit der Schriftart *Verdana* und einem kräftigen Rot gestaltet werden. Du brauchst keine weiteren Alternativschriften anzugeben.

Nutze wieder Inline-CSS und eine der 216 Browser-sicheren Farben.

### ■ Übung C3a: Style Sheets im HEAD auslagern

Gehe in den Übungs-Ordner `c3`, wo du eine weitere Kopie der `stil.html` abgelegt hast. Baue hier die gleichen Stilangaben wie in Übung C2a und C2b ein. Arbeite diesmal jedoch mit einem zentralen Stilbereich im `HEAD` des Dokuments! Erstelle also praktisch zwei Regeln, denen lediglich der entsprechende Selektor vorangestellt wird.



### ■ Übung C3b: Zeichenformat mit besonderer Schriftart einstellen

Bleibe noch im Übungs-Ordner c3. Hier hast du dir eben einen separaten Stilbereich im HEAD eingerichtet. Erstelle weiterhin eine Regel für die fett formatierte Passage (`<b>käuflich</b>`): Die Schriftgröße soll 14 Punkt, die Schriftfarbe #006600 (grün) sein.

### ■ Übung C3c: Mehrere lange Schriftartnamen angeben

Ergänze die eben erstellte Regel für `<b></b>` um eine Schriftenweisung. Gib als Hauptschriftart *Lucida Handwriting* an. Als Alternativschrift wähle *Tempus Sans ITC*. Falls auch diese Schrift nicht gefunden wird, soll eine allgemeine, kursiv-geschwungene Schrifttype ausgewählt werden.

Notationstipp: Denke an die langen Schriftartnamen mit Leerzeichen. Während in Inline-Stilen ausschließlich mit einfachen Anführungsstrichen gearbeitet wird, kannst du im separaten Stilbereich im HEAD auch doppelte Gänsefüßchen verwenden. Der Grund: Bei Inline-Stilen werden alle Stil-Eigenschaften und Werte schon von einer „Gänsefüßchen-Klammer“ zusammengehalten. Und Gänsefüßchen darf man nicht ineinander verschachteln. Diese Beschränkung fällt bei zentral im HEAD definierten Stilen weg.

### ■ Übung C3d: Unterschiedliche Formatierungen für `<i></i>`

Bleibe noch im Stilbereich deiner `stil.html` aus dem Ordner c3. Erstelle zwei weitere Regeln. Die erste Regel soll alle `<i></i>`-Elemente innerhalb einer H1 grau einfärben. Im Beispiel ist das die Passage *Dr. Stilbruch*. Befinden sich die `<i></i>`-Elemente allerdings innerhalb von Absätzen `<p></p>`, sollen sie lediglich zusätzlich fett formatiert werden.

### ■ Übung C3e (Profi): Erweiterte Formatierung: Listen interessant gestalten

Ändere die Aufzählungszeichen für die Bestell-Liste. Notiere eine Regel, mit der die Aufzählungszeichen wie Hohlkreise aussehen.

### ■ Übung C4a: Mehrere Selektoren gruppieren

Beenden wir unsere Experimente mit der „Dr. Stilbruch-Datei“ `stil.html`. Wage dich an das letzte vorbereitete Dokument im Ordner c4. Erstelle wieder einen separaten Stilbereich im HEAD. Lege als Gesamtschriftart *Verdana* in der Schriftgröße 11 Punkt fest, der Zeilenabstand soll 1,3 betragen. Gib weitere Alternativschriften an! Die Absätze **und die Aufzählung** sollen dunkelblau eingefärbt werden.

Verpasse **beiden** Überschriftsebenen die Farbe Dunkelgrau. Färbe die Elemente B (`<b></b>`) und I (`<i></i>`) dagegen rot. Nutze hierfür die Technik des Gruppierens von Selektoren.

**Profi-Tipp:** Wenn du eine Vordergrundfarbe mit `color` angibst, empfehlen die Macher von CSS, gleichzeitig eine Hintergrundfarbe anzugeben. Ergänze meinetwegen `background-color: white;`

### ■ Übung C4b (Super-Profi): font-Eigenschaften verwenden

Schaue zu den eben festgelegten Schriftarteneigenschaften im Selektor `body`. Notiere die `font`-Eigenschaften in der Kurzschreibweise. Speichere das Dokument unter dem Namen `profistil.html`.

## Kapitel 4: Rahmen, Ränder und Hintergrundeffekte

Style Sheets machen Spaß! Ich hoffe, dass du mir bis hierher unverdrossen und frohen Mutes gefolgt bist. War doch gar nicht so schwer, oder?

In diesem Übungsabschnitt lernst du Style Sheets nun so gut kennen, dass du mit dieser Technik richtig professionell arbeiten kannst.

Doch bevor wir uns wieder an unser Hauptprojekt wagen, müssen wir noch etliche neue CSS-Eigenschaften kennen lernen. Komm mit!

### Rahmen und Ränder

Zuerst zeige ich dir, wie man mit Rahmen und Rändern tolle Effekte erzeugt. Dann „triffst“ du die freien Absatz- und Zeichenformate und zum Schluss lagerst du dein Style Sheet extern aus.

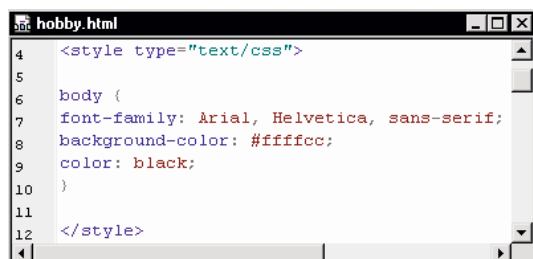
Doch der Reihe nach!

Gehe wieder in den Ordner `stiluebung`. Diesmal bearbeiten wir die Hobby-Seite!

#### ■ Übung macht den Meister

Rufe deine `hobby.html` auf. Ich gehe davon aus, dass hier noch keine Stilanweisungen notiert sind, abgesehen vielleicht von einem testweise notierten Aufzählungsformat.

Erstelle einen Style Sheet-Bereich im HEAD. Lege als Gesamtschriftart (Selektor `body`) *Arial* fest, gib als Alternativen *Helvetica* und eine allgemeine serifenlose Schrift an.



**Kein Problem mehr: Style Sheet-Bereich im HEAD.**

Stelle als Hintergrundfarbe `#ffffcc` ein. Da bei Angabe eines Hintergrunds auch ein „Vordergrund“ gern gesehen wird, nimmst du `black`.

Wir verzichten auf die Angabe einer Schriftgröße im `body`-Selektor. Eine einheitliche Schriftgröße für alle Elemente ist nicht sehr sinnvoll.

### Umrandungsstile von CSS

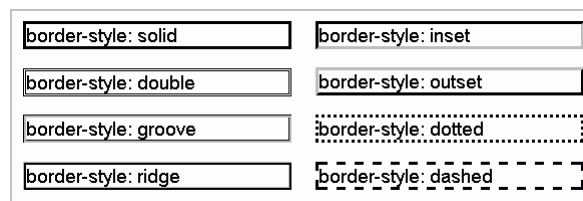
Wir experimentieren weiter. Wir versehen einen unserer Absätze mit einer Umrandung!

#### Umrandung mit border-style

Zum Zuweisen von Umrandungsstilen verwendest du die CSS-Eigenschaft `border-style`. Diese Eigenschaft kann folgende Werte annehmen:

Wert	Erläuterung
<code>solid</code>	durchgezogen
<code>double</code>	doppelte Rahmenlinie
<code>groove</code>	3-D-Effekt mit „Furche“ bzw. „Rille“
<code>ridge</code>	3-D-Effekt mit erhöhtem Rand, mit „Rücken“
<code>inset</code>	3-D-Effekt mit vertieftem Inhalt
<code>outset</code>	3-D-Effekt mit erhöhtem Inhalt („Schaltfläche“)
<code>dotted</code>	punktierter Rand
<code>dashed</code>	mit Strichen umzogen
<code>none</code>	kein Rahmenstil

So wird ein mehr oder minder schicker Rahmen um die Gesamtbreite des Elements gezogen:



**So sieht's aus: Werte von border-style im Überblick.**

**Beachte:** Nur neue Browser (Netscape 6, IE ab 5.5, Firefox) interpretieren *alle* Rahmenstile korrekt!

#### Umrandungsdicke und -farbe

Natürlich kannst du auch die Dicke der Rahmenlinie festlegen. Dafür nutzt du die Eigenschaft `border-width`. Gib den Wert in Pixeln an. Treibe es bunt! Auch die Rahmenfarbe lässt sich mit `border-color` ganz prima bestimmen.

Verpasse *einem* Absatz einen durchgezogenen roten Rahmen mit 1 Pixel Breite. Notiere also:

```

p {
  border-style: solid;
  border-width: 1px;
  border-color: red;
}

```

Das ist `stand1`. Und nun blättere bitte einmal um!

## Flexible Gestaltung durch separate Stilklassen

Was stellst du fest, nachdem du dir deine vom Bildschirmflimmern überanstrengten Äuglein gerieben hast?

Ja klar, verflixt und zugenäht. Jeder Absatz wurde mit solch einem Rahmen versehen! Logisch, schließlich bezieht sich der Selektor `p` auf alle Absätze gleichermaßen. Aber wir wollten eigentlich nur einen Absatz umranden.

Richtig, in diesem Fall könnten wir mit Inline-CSS arbeiten. Notiere die Stil-Eigenschaften also direkt im `<p>`-Tag unter Zuhilfenahme der `style`-Eigenschaft.

Doch wirklich flexibel ist die Inline-Lösung nicht.

### Separate Stilklassen notieren

Stell dir vor, du züchtest Hunde und vergisst, deinen vierbeinigen Freunden Namen zu verpassen. Wenn du „Komm“ rufst, kommen prompt alle angerannt. Nicht sehr praktisch! Doch exakt so verhält es sich bisher mit unseren Absätzen (und allen anderen Elementen). Wenn du ihnen keinen Namen verpasst, sind alle angesprochen.

Doch wie übertragen wir das Prinzip „Max, komm“, „Hasso, hier“ von Hunden auf HTML?

### Namen zuweisen per Eigenschaft `class`

Benenne deine HTML-Elemente. Dafür verwendest du die `class`-Eigenschaft. Im Beispiel wollen wir einen Absatz mit – nein, nicht mit Hasso – sondern mit `rahmen` benennen.

1. Gehe in das Tag, welches du benennen möchtest. Suche dir im Beispiel einen Absatz aus.
2. Setze den Cursor vor die schließende spitze Klammer und tippe ein Leerzeichen.
3. Tippe nun eine Eigenschaft mit der Syntax `class="name"`, im Beispiel schreibst du also `class="rahmen"`.  
`<p class="rahmen">`

### Gebundene und freie Klassen

Und nun? Was jetzt? Die Klasse scheint von ihrer Existenz wohl noch völlig unbeeindruckt zu sein! Weiterhin siehst du überall Rahmen!

Richtig, bis jetzt hast du im Style Sheet auch noch gar nicht auf diese Klasse verwiesen. Es ist so als ob du rufen würdest: „Hunde, kommt“. Dann kommt natürlich auch Hasso angewackelt (wenn er nicht gerade einer heißen Hündin hinterherläuft ☺).

### Notation für gebundene Klassen

Gehe also in den Style-Sheet-Bereich im HEAD. Schaue zum Selektor `p`. Notiere hinter dem `p` ohne Leerzeichen einen Punkt. Schreibe danach ohne Leerzeichen den Klassennamen.

`p.rahmen`

Fantastisch! Schon wird nur ein einziger Absatz mit einem Rahmeneffekt versehen.

Auch diese Punktnotation stammt von Programmiersprachen. Man wandert von links nach rechts; vom Allgemeinen (Absatz `p` bzw. Hund) zum Konkreten (`rahmen` bzw. Hasso).

Klassennamen kannst du dir im Prinzip frei ausdenken. Verzichte auf Leerzeichen und Umlaute, beginne nicht mit einer Ziffer oder einem Bindestrich und wähle die Namen nicht zu lang. Ich empfehle durchweg Kleinschreibung.

### Notation für ungebundene Klassen

Nun ist unsere mühsam erstellte Rahmenanweisung an das Element `<p>` gebunden. Nur Absätze lassen sich also so gestalten.

Eine ungebundene Klasse kannst du dagegen auf alle Elemente anwenden.

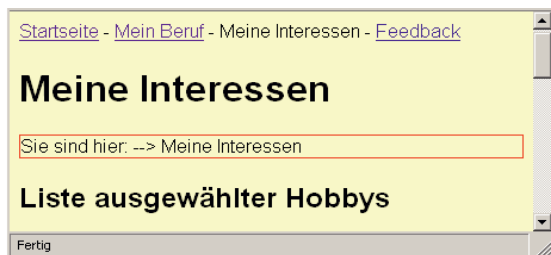
Um aus unserer gebundenen Klasse eine „freie“ zu machen, lässt du das `p` einfach weg. Schreibe nur den Punkt mit Klassennamen. Hier die gesamte derart umgeformte Regel im Blick:

```
.rahmen {
  border-style: solid;
  border-width: 1px;
  border-color: red;
}
```

Damit sind wir bei `kapitel4/stand2` angelangt. Wenn du willst, kannst du mir der `hobby.html` aus diesem Ordner vergleichen.

## Exaktes Layout: Ränder, Kasten und Innenrand

Zurück zu unseren Rahmen. Die Rahmenstile hast du inzwischen kennen gelernt. Doch normalerweise zieht sich so ein Rahmen per Voreinstellung über die gesamte Breite des übergeordneten Elements. Im Beispiel ist das `body` und damit ist der Rahmen automatisch so breit wie das Browserfenster.



**Der Rahmen zieht sich über die gesamte Fensterbreite.**

Außerdem klebt die Rahmenlinie viel zu dicht am Text. Und überhaupt: Könnte man nicht die Farbe der Rahmenlinie ändern?

Den linken oder rechten Rand? Nur die Dicke der oberen Linie? Nur den Innenabstand von oben? Und wie wäre es mit einer individuellen Breite? Einer Hintergrundfarbe? Einer Grafik?

Halt! Alle Wünsche können wir erfüllen, doch der Reihe nach. Ich stelle dir nun unzählige weitere wichtige Eigenschaften von CSS vor.

Wir lassen uns dabei nicht die Laune von Oldies wie dem Internet Explorer 5 verderben. Denn dieser alte Browser hatte so seine eigenen Vorstellungen von Breiten, Rahmen und Rändern und war nur bedingt „rand- und rahmentauglich“. Doch mit Aussterben von Windows 98 geht auch die Ära dieses alten Internet Explorers zu Ende.

### ■ Übung macht den Meister!

Die folgenden Eigenschaften gelten nicht nur für Rahmen! Wir experimentieren deshalb zuerst mit einem Absatz: Erstelle eine ungebundene Klasse namens `test`, schreibe `.test` als Selektor. Weise einem Absatz in der `hobby.html` nun über `class="test"` diese Klasse zu. Und nun manipulieren wir die Eigenschaften dieses Absatzes.

### Randeinstellungen mit margin

Zuerst geht es um den äußeren Rand. Dafür ist das Attribut `margin` verantwortlich. Als Maßeinheit empfehle ich Pixel.

Wenn du `margin` weglässt, gilt ein vom Browser gewählter kleiner „Anstandsrand“. Dieser stammt aus dem erwähnten „eingebauten Style Sheet“ des Browsers.

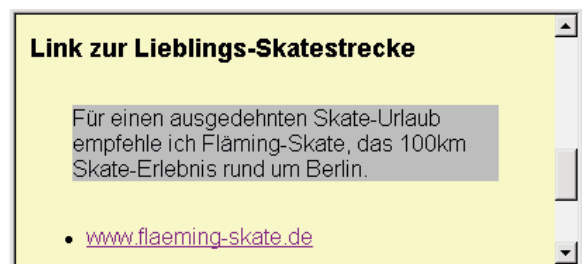
Weise deinem Test-Absatz zur Probe einen äußeren Rand von 30 Pixeln zu:

```
.test {
    margin: 30px;
}
```

Der Text schafft sich also rundherum einen Raum von 30 Pixeln. Von allen vier Seiten!

### Beispiel: Attraktiver „Farbkasten“

Füge für unser Experiment noch eine Hintergrundfarbe ein, z.B. `silver`.



**Grauer Kasten: Der Rand beträgt rundherum 30 Pixel.**

So erzeugst du einen attraktiven „Farbkasten“.

```
.test {
    margin: 30px;
    background-color: silver;
}
```

### Rand nur auf einer Seite

Soll der Rand nur auf einer Seite gelten? Und/oder für jede Seite unterschiedlich sein? Klar, geht. Du brauchst die folgenden vier Wörter für die „vier Himmelsrichtungen“:

- `top` (oben)
- `right` (rechts)
- `bottom` (unten)
- `left` (links)

Für einen linken Rand von 10 Pixeln notierst du z.B. `margin-left: 10px`;! Ein rechter Rand wird durch `margin-right` erzeugt usw. usw.

### Malerprinzip: Die letzte Farbe deckt!

Bleiben wir noch eine Weile beim Rand. Denn gerade an der Stelle wird es hochinteressant! Notiere

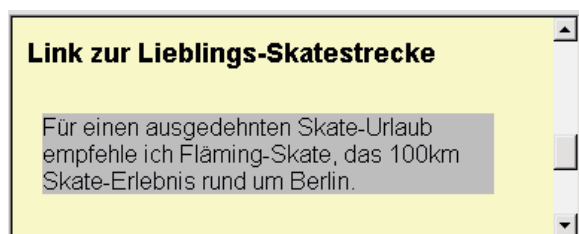
dieses eben besprochene Eigenschafts-Werte-Paar für den linken Rand, und zwar so:

```
.test {
  margin-left: 10px;
  margin: 30px;
  background-color: silver;
}
```

Speichere das Dokument und aktualisiere die Ansicht im Browser. Es passiert ... nichts!

Die Reihenfolge spielt eine wichtige Rolle.

Der Browser liest von oben nach unten. Hier gibt es zwei Eigenschaften, die sich um die Vorherrschaft prügeln, einmal 10 Pixel (nur für den linken Rand) und einmal 30 Pixel für alle Ränder (und damit auch für links).



**So klappt's: Der linke Rand ist nur 10 Pixel breit.**

Wer gewinnt? Die am tiefsten liegende Eigenschaft übernimmt im Beispiel die Vorherrschaft! Warum? Weil sie „am dichtesten dran ist“. Wie beim Malern: Die zuletzt gestrichene Farbe deckt!

Drehe die Reihenfolge einfach um:

```
.test {
  margin: 30px;
  margin-left: 10px;
  background-color: silver;
}
```

Nun betrachtet der Browser den Gesamtrand als Voreinstellung, als „Default“. Da er danach aber für `margin-left` eine andere Eigenschaft findet, überschreibt diese Eigenschaft die Voreinstellung von oben! Das Begreifen dieser Gesetzmäßigkeit ist wichtig für das Verständnis von CSS!

Natürlich kannst du `margin` auch weglassen oder alle vier Ränder individuell gestalten. Dann spielt die Reihenfolge keine Rolle.

### Seitenrand für das Dokument

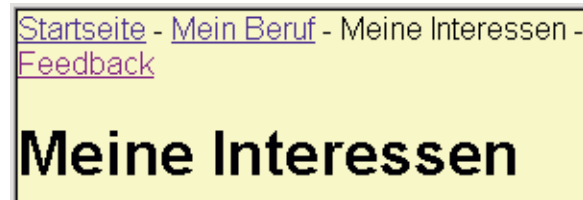
Schweifen wir kurz ab! Du möchtest die Randbreite des *gesamten* Dokuments bestimmen? Dann notierst du die Eigenschaft `margin` einfach direkt im Selektor `body`, und zwar so – siehe nächste Spalte oben:

```
body {
  margin: 10px;
}
```

Wenn du diese Angabe weglässt, gibt es aber trotzdem einen kleinen Rand: Der Browser weist diesen anhand seines eingebauten Style Sheets zu.

### ■ Probeweise Gesamtrand unterdrücken

Unterdrücke einmal – zur Probe – den eingebauten „Anstands-Rand“ für dein gesamtes Dokument.



**Sieht nicht gut aus: Mit `margin: 0`; klebt der Text am Browserrand.**

Das gelingt mit `margin: 0`; im `body`-Tag. Warum eigentlich nicht `margin: 0px`? Weil 0 gleich 0 ist und keine Maßeinheit braucht!

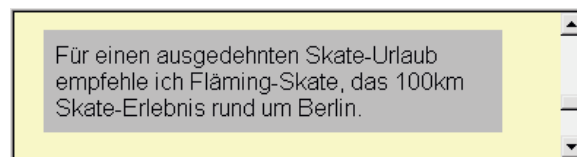
Und nun lösche die Randangaben wieder, wir geben uns mit dem „eingebauten Rand“ zufrieden.

### Mehr Luft bitte: Innenrand

Zurück zu unserem „Farbkasten“. Die Hintergrundfarbe ist ja sehr nett. Trotzdem „klebt“ der Text *im* Kasten viel zu dicht am Rahmen. Kein Problem! Definiere einfach zusätzlich mit der Eigenschaft `padding` einen Innenrand als Polster!

Padding steht für Füllung, Polsterung:

```
.test {
  margin: 30px;
  margin-left: 10px;
  padding: 8px;
  background-color: silver;
}
```



**Mehr Luft durch Innenrand mit `padding: 8px`;**

Auch hier kannst du jede Seite individuell mit `padding-left`, `padding-top` usw. gestalten.

Wichtig zu wissen: Die *eigentliche* Breite des Inhalts (`width`) verändert sich dadurch nicht. Nur das „Drumherum“ wird natürlich breiter!

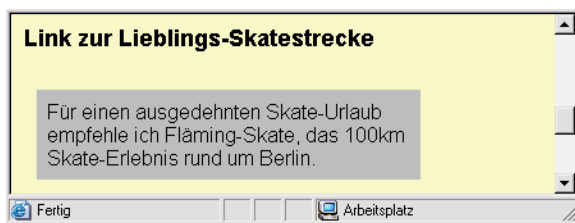
Wir sind jetzt bei `kapitel4/stand3` angelangt.

## Absolut oder relativ? Breitenangaben justieren

### Breite des Elements einstellen

Wie stellst du die Breite deiner Rahmen, Absätze und Farbkästen ein? Nutze die schon bekannte Eigenschaft `width` für die Angabe der Breite.

Du kannst sowohl absolute als auch relative Werte notieren. Mit der relativen Angabe `width: 50%`; füllt dein Element rund die Hälfte der Fensterbreite aus. Oder die Hälfte des übergeordneten Elements. Für exaktes Layout ist solch eine relative Angabe kaum brauchbar. Besser sind deshalb stets absolute Angaben, beispielsweise in Pixeln. Die nehmen wir auch!



**Absolute Angabe: Der Kasten ist genau 300 Pixel breit.**

Bleiben wir bei unserem Beispiel „Farkasten“, dem Absatz mit der Klasse `test`. Notiere zusätzlich eine Breite von 300 Pixeln.

```
.test {
    margin: 30px;
    margin-left: 10px;
    padding: 8px;
    background-color: silver;
    color: black;
    width: 300px;
}
```

Wunderst du dich, warum ich zusätzlich `color: black`; eingefügt habe? Wo die Schriftfarbe doch normalerweise sowieso Schwarz ist? Das wird von den Machern der Style-Sheet-Sprache so empfohlen. Wenn Hintergrundfarbe, dann auch Vordergrundfarbe. Aber eigentlich ist es nicht nötig und ich mache es hier nur, um überkorrekt zu sein.

Die Breite `width` bezieht sich übrigens immer nur auf die Breite des Textbereichs. Eigenschaften wie `padding`, `border` oder `margin` werden dazuaddiert!

### Individuelle Breiten mit width

Lassen wir unserer Fantasie freien Lauf. Fügen wir zusätzlich eine Umrandung ein. Diese soll zwei Pixel breit sein und rot eingefärbt werden.

### Link zur Lieblings-Skatestrecke

Für einen ausgedehnten Skate-Urlaub  
empfehle ich Fläming-Skate, das  
100km Skate-Erlebnis rund um Berlin.

Fertig

Arbeitsplatz

**Schick: Individuelle Breite mit `border-left-width`.**

Als Clou wählen wir für den linken Bereich jedoch eine abweichende Breite von 20 Pixeln. Das gelingt durch `border-left-width`.

```
.test {
    (6 Zeilen: siehe Nachbarspalte)
    border-style: solid;
    border-width: 2px;
    border-left-width: 20px;
    border-color: red;
}
```

Auch hier hilft wieder das „Malermeister-Prinzip“. Wir notieren `border-left-width` unterhalb von `border-width`, damit diese Eigenschaft das letzte Wort bekommt.

### ■ Die Berechnung der Breite

Im Beispiel ist unser Kasten nun nicht nur 300 Pixel breit, sondern 300 Pixel (`width`) + 16 Pixel Innenrand (`padding`, je 8 Pixel für links und rechts) + 20 Pixel linken Rand (`border-left-width`) + 2 Pixel rechten Rand (`border`). Macht summa summarum also 338 Pixel! Den äußeren Rand, also `margin`, habe ich dabei noch gar nicht mitgezählt. Auf die Breite bezogen kommen links 10 Pixel (`margin-left`) und rechts 30 Pixel (`margin`) „Luft“ hinzu.

Die Hintergrundfarbe zieht sich übrigens bis über diese gesamte Breite von 338 Pixeln. Sie erstreckt sich unterhalb des Innenrand (`padding`) und selbst unterhalb der Rahmenlinie (`border`). Da `border` jedoch selber eine Farbe besitzt – im Beispiel rot – überschreibt die Rahmenfarbe die Hintergrundfarbe. Übrigens stellt der Internet Explorer die Breite nicht immer richtig dar. Mehr über dieses ärgerliche Problem erfährst du gleich auf der nächsten Seite.

### ■ Übung macht den Meister!

Zurück zum ersten Beispiel, zur Umrandung. Stelle den Innenrand des Elements (`.rahmen`) auf 3 Pixel ein und die Rahmendicke auf 2 Pixel. Damit sind wir bei `kapitel4/stand4` angelangt.



## Alles im Rahmen: Hintergrundinfos zum Boxmodell

Wichtig zu wissen: Alle Elemente einer HTML-Seite stecken in einem Rahmen bzw. einer „Kiste“. Bei Elementen auf Zeichenebene wie `<b></b>` oder `<strong></strong>` ist der Rahmen so breit wie das damit formatierte Zeichen.

### ■ Der Rahmen der Blockelemente

Bei so genannten Blockelementen wie Überschriften, Absätzen, Listenelementen, Formularbereichen usw. dagegen erstreckt sich dieser Rahmen – falls nicht anders festgelegt – über die gesamte Breite des übergeordneten Elements. Dieses übergeordnete Element ist im Beispiel `<body>`, also das gesamte Browserfenster. Es kann aber auch ein übergeordnetes DIV oder ein anderes Blockelement sein.

In der folgenden Abbildung habe ich alle normalerweise unsichtbaren Rahmen sichtbar gemacht. Das gelingt beispielsweise mit der Web Developer Toolbar im Firefox. (siehe Seite 77). Wähle dort **HERVORHEBEN | BLOCK-LEVEL-ELEMENTE HERVORHEBEN**.



Selbst Listenelemente `<li>` stecken in einem Rahmen. Bis auf den roten von uns mit einer festgelegten Breite ausgestatteten Rahmen namens „test“ erstrecken sich alle anderen Rahmen über die gesamte Fensterbreite.

### Padding, Border und Margin

Und wie du bisher weiterhin erfahren hast, kannst du die Breite eines Rahmens auch individuell festlegen: mit `width`. Und du kannst außerdem den Innenrand (`padding`), die Rahmenlinie (`border`) und den Außenrand (`margin`) wunschgemäß bestimmen. Doch genau an dieser Stelle fängt das Drama an. Denn ältere Ausgaben des Internet Explorers fangen hier fürchterlich an zu patzen.

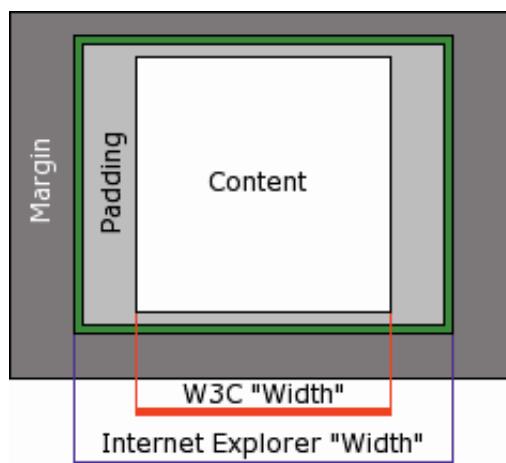
### Boxmodell-Fehler vom Explorer

Laut CSS-Spezifikation gilt: Zusätzliche Abstände wie `padding`, `border` oder `margin` werden stets zur Breite (bzw. Höhe) eines Rahmens bzw. einer Box dazuaddiert. So haben wir das besprochen – das ist das korrekte Boxmodell von CSS. Beispiel gefällig?

Zur Erinnerung: Unser auf den Vorseiten erstellter Rahmen namens „test“ ist 300 Pixel breit. Der Innenrand `padding` beträgt 8px. Die Rahmenbreite `border-width` beträgt 2px, nur der linke Rahmen `border-left-width` hat 20 Pixel. Das alles wird dazuaddiert:  $300 + 8 + 8 + 2 + 20 = 338$ . So beträgt die sichtbare Gesamtbreite bis zur äußeren Kante des roten Rahmens im Beispiel also 338 Pixel. (Der unsichtbare Außenrand `margin` kommt natürlich auch noch hinzu, er wahrt die Abstände nach allen Seiten.)

Der Internet Explorer bis einschließlich Version 5.5 dagegen zieht die Werte für `padding` und `border-width` von den 300 Pixeln Breite ab und bietet dem Inhalt somit nur noch  $300 - 8 - 8 - 2 - 20 = 262$  Pixel Platz. Das mag konsequent und sinnvoll erscheinen, entspricht aber nicht dem Standard.

Die folgende Abbildung zeigt das Problem: Breite ist nicht gleich Breite.



Die eigentliche Breite (`width`) gilt per W3C-Standard nur für den Content, den Inhalt (rote Maßlinie). Der alte Internet Explorer (blaue Maßlinie) jedoch misst die Breite bis über die äußere Kante der Rahmenlinie (`border`). Bild: Wikipedia.

Der Internet Explorer 6 (und alle übrigen aktuellen Browser) interpretieren das Box-Modell allerdings korrekt. Der Internet Explorer 6 macht das aber nur dann, wenn du eine korrekte DTD verwendest.

### ■ Der Quirks-Modus

Trotzdem wiegt das Erbe schwer: Im guten Glauben auf ältere Browser zugeschnittene Seiten würden in aktuellen Browsern keine gute Figur mehr machen. Deshalb kennen neuere Browser den so genannten „Kompatibilitätsmodus“, auch als „Quirks-Modus“ bezeichnet. Die Browser simulieren dabei weitgehend das fehlerhafte Verhalten der älteren Browsergeneration und machen die „Marotten“ mit. (Quirks heißt übersetzt schließlich „Eigenheiten“ bzw. „Marotten“.) Allerdings zeigt nur der Internet Explorer im Quirks-Modus die falschen Breiten an. Firefox interpretiert die Rahmen in allen Fällen korrekt.

### ■ Der Standard-Modus

Das Gegenstück dazu ist der Standard-Modus. Erst in diesem Modus wird der Box-Modell-Fehler beim Internet Explorer nicht mehr simuliert.

Im Firefox kannst du den gewählten Modus ganz einfach feststellen: Rechtsklicke in die gewünschte Seite und wähle **SEITENINFORMATIONEN ANZEIGEN**. Schau ins Register *Allgemein* und lies den Anzeigemodus ab. In anderen Browsern wie dem Internet Explorer, Chrome oder in Opera tippst du dagegen

```
javascript:alert(document.compatMode)
```

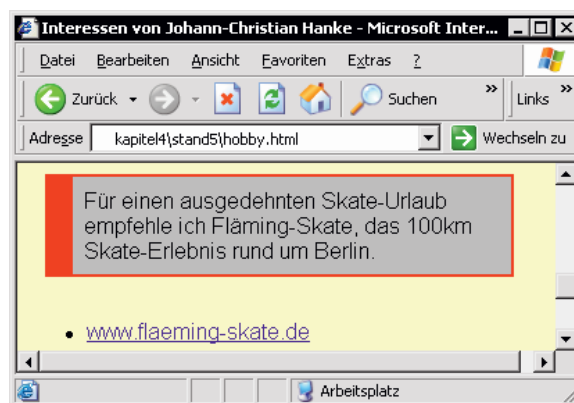
in die Adresszeile ein. Nach Druck auf **Enter** informiert dich eine JavaScript-Box über das Ergebnis. *backCompat* steht für den besagten Kompatibilitäts-Modus, *CSS1Compat* dagegen für die standardgerechte Anzeige. Dank des Kompatibilitäts-Modus simuliert zumindest der Internet Explorer weiterhin das Fehlverhalten seiner Vorgänger einschließlich des Box-Modell-Fehlers.

Du als Seitenbetreiber kannst und solltest stets den Standard-Modus erzwingen: Gib einfach die korrekte Dokumenttyp-Deklaration für deine HTML-Version an. So, wie wir es im ganzen Heft schon gemacht haben. Du bevorzugst eine verkürzte DTD wie diese hier für HTML 4.01? So wie auch in älteren Auflagen dieses Titels?

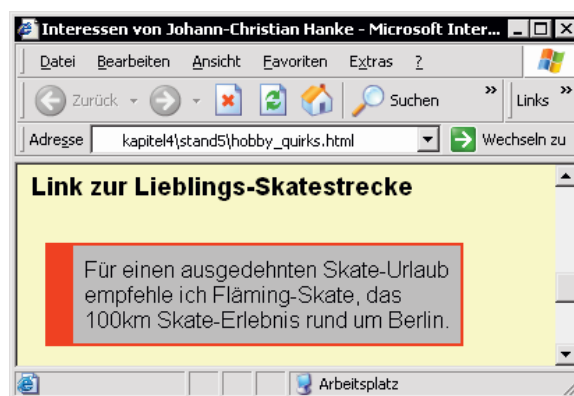
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Das ist ja keineswegs falsch. Trotzdem schalten dann einige Browser wieder in den Kompatibilitätsmodus – auch der Internet Explorer. Und schon stimmen die Maße deiner DIV-Container nicht mehr! Nervig? Jawohl! Das nervt total!

Aber auch ein „überkorrekter“ Beginn kann Probleme bereiten. Denn eigentlich beginnen XHTML-Dokumente mit einem XML-Prolog. Der DTD müsste stets folgende Zeile vorangestellt werden: `<?xml version="1.0"?>` Aber auch das bewegt zumindest den Internet Explorer 6 dazu, statt des Standard-Modus den Kompatibilitäts-Modus zu wählen. Aus diesem Grund solltest du den Prolog weglassen und aus genau diesem Grund wähle ich im Heft auch HTML 4 und nicht XHTML. (Erst der Internet Explorer 7 stößt sich nicht mehr an diesem XML-Prolog.)



**So ist es richtig: Der Rahmen wird im Internet Explorer mit korrekter Breite (hier 338 Pixel) dargestellt. Zusätzliche Angaben wie padding, border oder margin werden zur Breite width dazuaddiert.**



**Gleiche Seite im Quirks-Modus: Der Platz für den Text wird viel kleiner – zumindest im Internet Explorer – er beträgt nur noch 262 Pixel. Vergleiche mit der Datei „hobby\_quirks.html“ im Ordner „stand5“.**

Wenn du dich für alle Details dieser Problematik interessierst, empfehle ich dir den sehr interessanten Beitrag „Der DOCTYPE-Switch und seine Auswirkungen“ von Carsten Protsch auf [www.carsten-protsch.de](http://www.carsten-protsch.de).

## Zusatzwissen für Profis: Mehr zu Hintergrundeffekten

An dieser Stelle also wieder der Profi-Bereich. Du willst mehr zu Hintergrundeffekten wissen? Füge ein Hintergrundbild ein. Kachele oder positioniere es exakt an einem bestimmten Fleck.

Nutze die Kurznotationen für bestimmte Eigenschaften und Werte. Die meisten Effekte funktionieren im Internet Explorer, Opera und natürlich Firefox.

### Tapete: Hintergrundbild einfügen

Das Zuweisen von Hintergrundfarben kennst du. Wähle `background-color`. Doch vielleicht liebst du es eher grafisch? Egal ob gesamtes Dokument (body), Überschrift, Absatz oder DIV-Container: Du kannst viele HTML-Elemente mit einer Hintergrundgrafik versehen.

Dazu dient die Eigenschaft `background-image` in folgender Syntax:

```
background-image: url(Pfad/Name.end);
```

Du wünschst eine Hintergrundgrafik für dein gesamtes Dokument? Sie heißt `textur.jpg` und liegt im gleichen Ordner wie das HTML-Dokument? Schreibe

```
body {
    background-image: url(textur.jpg);
}
```

Für einen Tapeteneffekt sollte die Grafik so ausgewählt werden, dass sie nahtlos an sich selbst anschließt (Textur). Vergleiche mit der Datei `tapete.html` unter `kapitel4/profi`.

### Wiederholverhalten einstellen

Die Eigenschaft `background-repeat` wiederum legt fest, wie die Grafik ausgerichtet werden soll. Per Voreinstellung wird die Grafik gekachelt, also so oft wiederholt, bis der gesamte Hintergrund ausgefüllt ist. Das ergibt den eben erwähnten Textur-Tapeten-Effekt.

Interessant sind auch die anderen Varianten:

Eigen-schaft	Erläuterung
repeat	Voreinstellung: wiederholen, kacheln
repeat-x	Wiederholung nur horizontal, „auf der x-Achse“
repeat-y	Wiederholung nur vertikal, „auf der y-Achse“
no-repeat	keine Wiederholung

### Soll die Grafik mitrollen?

Per Voreinstellung wird eine Hintergrundgrafik gerollt. Bei Texturen ist es egal, man erkennt es nicht. Doch gerade große Grafiken solltest du mit der Eigenschaft `background-attachment` fixieren. Die Eigenschaft besitzt folgende Werte:

Eigenschaft	Erläuterung
scroll	Voreinstellung, Browser bewegt Hintergrundgrafik mit
fixed	Hintergrundgrafik bleibt feststehend

### ■ Beispiel: Grafik am Rand fixiert

Fixieren wir eine Grafik am linken Rand!



Es darf gerollt werden: Die Hintergrundgrafik ist fixiert.

Hinter dieser Abbildung verbirgt sich folgende Regel:

```
body {
    margin-left: 120px;
    font: 12pt Arial, sans-serif;
    background-image: url(blute.gif);
    background-repeat: no-repeat;
    background-attachment: fixed;
}
```

Wichtig ist der zusätzliche linke Rand für den Text. (Sonst wäre die Grafik ein hinter dem Text liegendes „Wasserzeichen“.)

Du findest das Beispiel unter `kapitel4/profi` unter dem Namen `fixed.html`.

### Hintergrundbild exakt platzieren

Du möchtest ein Logo exakt an einer bestimmten Stelle im Hintergrund platzieren? Wähle die Eigenschaft `background-position`.

Notiere erst den Abstand von links, dann von oben. Und zwar in folgender Syntax, ohne Komma:

```
background-position: links oben;
```

Bezugspunkt ist die linke obere Ecke des Browserfensters. Du möchtest eine Grafik 20 Pixel von links und 50 von oben ausrichten? Notiere:

```
background-position: 20px 50px;
```

Alle neueren Browser kommen mit dieser Anweisung zurecht.



Kein Problem in Opera 5/6: Exaktes Positionieren.

Das Beispiel findest du unter `kapitel4/profi` unter dem Namen `exakt.html`.

### Hintergrund-Steno mit background:

Erinnerst du dich noch an die `font:-`Kurzform? Auch für Hintergrundeffekte gibt es solch eine Steno-Variante.

Diesmal ist jedoch alles ganz einfach. Schreibe `background:` Notiere dahinter einen, mehrere oder alle Eigenschafts-Werte von

- `background-color` (Hintergrundfarbe)
- `background-image` (Hintergrundbild)
- `background-repeat` (Wdhlg. des Bildes)
- `background-attachment` (Fixierung)
- `background-position` (exakte Position)

Trenne die Werte durch Leerzeichen. Die Reihenfolge ist diesmal egal. Hier zwei Beispiele:

```
background: silver;
```

```
background: url(bild.gif) no-repeat fixed;
```

### Rahmen-Steno mit border:

Die Kurzform für Ränder ist ebenfalls ganz praktisch. Die Syntax lautet:

```
border: Wert1 Wert2 Wert3;
```

Gib einfach die Werte folgender Eigenschaften an:

- `border-width` (Randbreite wie 1px)
- `border-style` (Rahmenstil wie solid)
- `color` (Farbe wie red)

Die Reihenfolge und Anzahl ist egal. Eine zwei Pixel dicke rote Umrandung erreichst du so:

```
border: 2px solid red;
```

Allerdings kann man mit „border-Steno“ leider nur alle vier Wände gleichzeitig ansprechen. Ganz anders als bei unseren nächsten beiden „Steno-Kandidaten“.

### Rand-Steno mit margin:

Die „Rand-Steno“-Variante kennst du im Prinzip schon. Mit `margin: 20px;` sprichst du alle vier Seiten gleichzeitig an. Es gibt jedoch eine Möglichkeit, jede Seite separat anzugeben und sich dabei trotzdem kurz zu fassen.

Man geht vor wie bei einer gefälschten Schweizer Präzisionsuhr aus Bangkok: Oben beginnend im Uhrzeigersinn. Und zwar so:

```
margin: top right bottom left;
```

Zuerst gibst du also den Wert für den oberen, dann für den rechten, für den unteren und schließlich für den linken Rand an.

```
margin: 0 10px 0 20px;
```

Die 0 braucht hier keine Einheit, denn 0 ist 0!

Ein Beispiel findest du in der Datei `margin.html`.

### Innenrand-Steno mit padding:

Auch für den Innenrand gibt es solch eine praktische Kurzschreibweise. Entweder du notierst das schon bekannte `padding` mit einem einzigen Wert (der dann für alle Kanten gleichzeitig gilt). Oder du gibst die einzelnen Werte wieder reihum im Uhrzeigersinn an:

```
padding: top right bottom left;
```

Zu allem Überfluss kannst du statt vier auch nur zwei Werte verwenden:

```
padding: top_bottom left_right;
```

Dabei steht der erste für oben und unten und der zweite für links und rechts. Folgende Zeile

```
padding: 20px 5px;
```

definiert also einen oberen und unteren Rand von 20 Pixeln und einen linken und rechten von 5 Pixeln. Steno für Profis heißt also: Fasse dich kurz!

## ✓ Übungsteil D: Rahmen, Ränder, Effekte

Du weißt jetzt, wie man:

- mit Umrandungen und Farben Rahmeneffekte erzielt
- separate Stilklassen nutzt
- Außen- und Innenrand einstellt
- Hintergrundgrafiken positioniert



Hast du die Übungen C3a bis C3d bzw. C3e von Seite 38/39 gemacht? Sehr gut! Dann kopiere den Inhalt vom Ordner c3 in einen neuen Ordner namens d1.

### ■ Übung D1a: Ungebundene Klasse erstellen

Füge den Absatz *Wir sorgen für guten Stil* der Klasse *kasten* hinzu. Es soll eine ungebundene Klasse werden. Erstelle im Style Sheet einen entsprechenden Selektor für eine ungebundene Klasse *kasten*.

### ■ Übung D1b: Rahmen mit „Schaltflächen-Effekt“

Erstelle für diesen Absatz *kasten* rundherum einen Rahmen mit Schaltflächeneffekt (outset). Die Rahmenbreite soll 2 Pixel betragen. Wähle als Rahmenfarbe Blau.

### ■ Übung D2a: Rahmenbreite und Innenrand einstellen

Kopiere die Datei *stil.html* aus dem Ordner d1 in einen neuen Ordner d2. Sorge dafür, dass der Rahmen 300 Pixel breit ist und einen Innenrand von 3 Pixeln besitzt.

### ■ Übung D2b: Hintergrundfarbe einstellen

Weise dem Rahmen als Hintergrundfarbe ein zartes Gelb zu.

### ■ Übung D3: Hellgrauen Farbkasten mit Rahmen einrichten

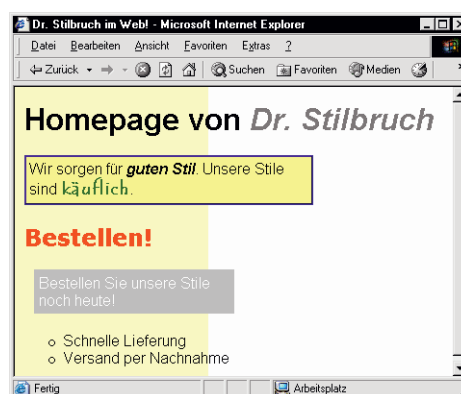
Kopiere die Datei *stil.html* aus dem Ordner d2 in einen neuen Ordner d3. Erstelle eine weitere ungebundene Klasse namens *hellgrau*. Weise sie dem letzten Absatz zu, der Passage *Bestellen Sie unsere Stile noch heute*. Richte einen hellgrauen Farbkasten ein (hellgrau = *silver*), der von links und rechts einen Rand von 10 Pixeln besitzt. Der Innenrand soll 5 Pixel betragen. Die Breite des Contents soll 50% des Browserfensters betragen. Als Schriftfarbe wählst du Weiß.

### ■ Übung D4 (Profi): Farbverlaufseffekt erstellen

Erstelle mit deiner Bildbearbeitung eine Grafik, die mindestens 1800 Pixel lang ist. Die Höhe kannst du so klein wählen wie du willst, ich empfehle 10 Pixel. Die ersten 200 Pixel sollen mit einem zarten Gelb gefüllt werden. Der Rest bleibt weiß.

Speichere die Grafik unter dem Namen *verlauf.gif* im neuen Ordner d4. Kopiere die Datei *stil.html* aus dem Ordner d3 in den Ordner d4. Binde die Grafik *verlauf.gif* als Hintergrundgrafik ein.

Du brauchst die Hintergrundfarbe dafür nicht zu entfernen. Hauptsache du notierst die Anweisung für die Hintergrundgrafik etwas tiefer!



Toller Effekt: Die Grafik wird stets wiederholt.



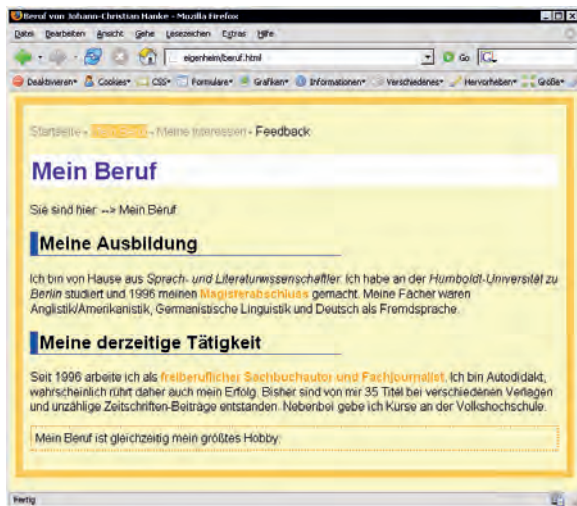
## Kapitel 5: CSS professionell! DIV, SPAN, hover und Co.

Feierabend! Schluss mit der Experimentiererei! Zurück zu unserem Projekt Feierabendheim; pardon: eigenheim.

Jetzt bist du mit dem nötigen Rüstzeug ausgestattet, um das Projekt „Selbstdarstellung“ zu einem guten Ende zu führen. Die paar „Kleinigkeiten“ die noch nötig sind, zeige ich dir auf dem Weg! Mach mit!

Lassen wir die Experimente aus dem Ordner stilübung hinter uns und wagen uns wieder in den Ordner eigenheim.

Hier liegen unsere drei HTML-Dokumente noch jungfräulich und völlig unberührt. Wie wollen wir diese gestalten? Folgendermaßen:



Hier die Seite „beruf.html“ in der Endfassung.

Wenn du die anderen beiden Seiten sehen willst: Blättere zur Seite 5 und zur Seite 50. Dort siehst du den Rest des „Modells Personenkult“ im Endlayout.

### Mehr Freiheit: <div> und <span>

So schön HTML auch ist: Leider „besitzen“ die meisten Tags bestimmte eingebaute Eigenschaften wie Absatzabstand, Schriftgröße, Formate wie fett, kursiv usw. usf. Oder vielmehr: Die Browser sorgen dafür, dass diese Eigenschaften angezeigt werden.

### Block- und Zeichenebene

Deshalb wurden die Tags <div></div> und <span></span> entwickelt. DIV ist eine Art „freier Container“ auf Absatzebene, eine Box für Text, Absätze, Überschriften und Grafiken.

Es handelt sich um ein Blockelement ohne eingebaute Eigenschaften.

Innerhalb von <div></div> kannst du kurze Zeilen oder umfangreiche Passagen unterbringen. Die einzige „Eigenschaft“ ist der Zeilenumbruch. DIV-Elemente lassen sich sogar ineinander verschachteln. DIV ist ideal zum Layouten mit CSS und zum Positionieren von Elementen.

Absätze dagegen darfst du weder ineinander verschachteln noch kannst du Überschriften einfügen. DIV dagegen bedeutet Freiheit auf Blockebene!

SPAN wiederum ist ein Inline-Element wie <b></b> oder <i></i>. Es besitzt im Gegensatz zu B oder I jedoch keine eigenen Eigenschaften, die man erst mühevoll abschalten müsste.

Selbstverständlich lassen sich auch <div>- oder <span>-Container mit Klassen versehen.

### ■ Übung macht den Meister!

Ran an das Projekt! Wofür brauchen wir überhaupt DIV und SPAN? Zuerst sollen unsere drei Seiten je einen dicken Außenrand erhalten. Dazu müssen wir einen DIV-Container einrichten, der den gesamten Inhalt umspannt.

BODY taugt hierfür nicht, da die Browser zu unterschiedlichen Anzeigergebnissen kommen!

Bau also in alle drei Dokumente direkt unter <body> je ein Einschalt-DIV und haarscharf über </body> das korrespondierende Ausschalt-DIV ein. Ordne das Einschalt-DIV per class-Eigenschaft der Klasse rand zu. Notiere:

```
<div class="rand">Seiteninhalt</div>
```

(Bitte noch keine Stil-Regel mit class-Selektor einrichten, das machen wir gleich gemeinsam.)

Nun zur vorbereiteten Linkleiste! Hier gibt es auf jeder Seite eine Textstelle, die die aktive Seite hervorhebt. So wird auf der index.html der Text *Startseite* natürlich nicht verlinkt. Wickle diesen Text in SPAN ein. Wähle als Klasse aktiv. Notiere auf der index.html:

```
<span class="aktiv">Startseite</span>
```

Später heben wir dieses Wort per CSS hervor!



## Volle CSS-Power: Style Sheet-Bereich extern auslagern

Du hast deine drei HTML-Seiten im Ordner *eigenheim* mit DIV und SPAN ausgestattet? Und noch keine Stil-Notierungen vorgenommen? Gut! Steigen wir nun in die CSS-Bundesliga auf!

Auf Dauer ist die bisherige Vorgehensweise, also das Notieren der Stile im HEAD, nicht sehr ökonomisch. Denn im Prinzip müsstest du deine Regeln immer wieder neu schreiben, Seite für Seite. Denke einmal an umfangreiche Sites mit hunderten von Dateien. Was für ein Riesen-Verwaltungsaufwand! Welches Fehlerpotenzial!

CSS wäre nicht CSS, wenn es nicht eine Lösung für dieses Problem gäbe.

Lagere deine Stilanweisungen extern aus!

Das Zauberwort lautet Stil-Recycling: Wir verwenden eine einzige „Regel-Sammlung“ und nutzen diese in drei, zehn oder hundert Dokumenten gleichzeitig.

Wenn du dann das Aussehen einer Passage ändern willst, nimmst du diese Änderungen nur ein einziges Mal in der Stildatei vor. Auf einen Schlag ändert sich das Aussehen in allen hundert HTML-Dateien. Ist das nicht fantastisch?

Setzen wir das doch gleich in die Praxis um!

### Externe CSS-Datei erstellen

Unsere externe Stildatei soll *standard.css* heißen. Wir legen sie im gleichen Ordner ab, also im Ordner *eigenheim*.

CSS-Dateien sind simple Textdateien mit der Endung *css*.

Und so legst du eine CSS-Datei an:

1. Gehe im Windows-Explorer (oder Arbeitsplatz) in deinen Projektordner, im Beispiel also in den Ordner *eigenheim*.
2. Klicke mit der rechten Maustaste auf einen freien Bereich in diesem Ordner. Wähle (übrigens genau wie beim Anlegen einer HTML-Datei) den Befehl **NEU**. Entscheide dich für den Unterbefehl **TEXTDATEI**.

3. Überschreibe den Platzhalter-Namen mit dem Namen *standard.css*.
4. Bestätige deine Einstellungen mit **OK**. Auch die Warnung, dass die Datei unbrauchbar würde, kannst du gelassen mit **JA** wegdrücken.
5. Was passiert, wenn du auf diese Datei doppelt klickst? Warum öffnet sich der Editor nicht?

Falls sich bei dir der Editor jetzt nicht öffnet, besteht auf deinem PC keine Verknüpfung der Endung *css* mit dem Editor (Notepad). (Siehe auch mein Windows-Heft 166)



**Ideal: Verknüpfe die Endung *css* mit dem Editor.**

6. Wenn nun die *Öffnen mit*-Dialogbox erscheint, wählst du den **EDITOR** (Notepad) aus. Hake unbedingt *Diesen Dateityp immer mit diesem Programm öffnen* ab und wähle **OK**.

Jetzt öffnet sich der Editor, du kannst beginnen!

### ■ Weitere Tricks zum Bearbeiten

**Editor schneller öffnen:** Wähle **START | AUSFÜHREN**. Tippe Notepad und drücke **[Enter]**.

**CSS-Datei schneller öffnen:** Ziehe das CSS-Dateisymbol bei gedrückter linker Maustaste in das geöffnete Editor-Fenster.

**Weaverslave verwenden:** Oder verwende auch für CSS den schon erwähnten Weaverslave. Der zeigt dir immerhin eine tolle Farbhervorhebung! Ziehe die CSS-Datei einfach in das geöffnete Weaver-slave-Fenster oder öffne sie über den Dateibrowser.

## Quelltext der externen CSS-Datei im Überblick

Das Editor-Fenster ist geöffnet? Tippe einfach los!  
Ich empfehle dir, zuerst meine Fassung komplett zu übernehmen. Später kannst du gerne Änderungen vornehmen.

Die Selektoren habe ich der Übersichtlichkeit halber besonders hervorgehoben

```
body {
    background-color: #ffffcc;
    color: black;
}

div.rand {
    font-family: Arial, Helvetica, sans-serif;
    width: 700px;
    padding: 10px;
    border-style: solid;
    border-width: 10px;
    border-color: #ffcc66;
}

h1 {
    background-color: white;
    color: #0000cc;
    padding: 2px;
}

h2 {
    border-left-style: solid;
    border-left-width: 10px;
    border-bottom-style: solid;
    border-bottom-width: 1px;
    border-color: #0066cc;
    width: 400px;
    padding-left: 2px;
}

.dotbox {
    border-style: dotted;
    border-width: 2px;
    border-color: #ff9900;
    padding: 5px;
}

span.aktiv {
    background-color: #ffcc66;
    color: white;
}
```

```
b {
    color: #ff9900;
}

a {
    text-decoration: none;
    color: black;
}

a:visited {
    color: gray;
}

a:hover {
    background-color: #ffcc33;
    color: white;
}

ul {
    list-style-image: url(kreis.gif);
}
```

Wie du siehst, sind in dieser externen Stildatei keinerlei `<style>`-Tags nötig!

### Link zur Style Sheet-Datei

Gleich besprechen wir das Wichtigste im Einzelnen! Denn da gibt es durchaus Neues zu entdecken. Doch vorher verrate ich dir noch, wie du diese externe CSS-Datei in eine HTML-Datei einbindest.

Füge auf jeder der drei HTML-Seiten einen Link zur externen Style Sheet-Datei ein.

Diesen Link platzierst du in einer eigenen Zeile unter `</title>` und über dem abschaltenden `</head>`. Die Syntax sieht so aus (alles in einer Zeile):

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

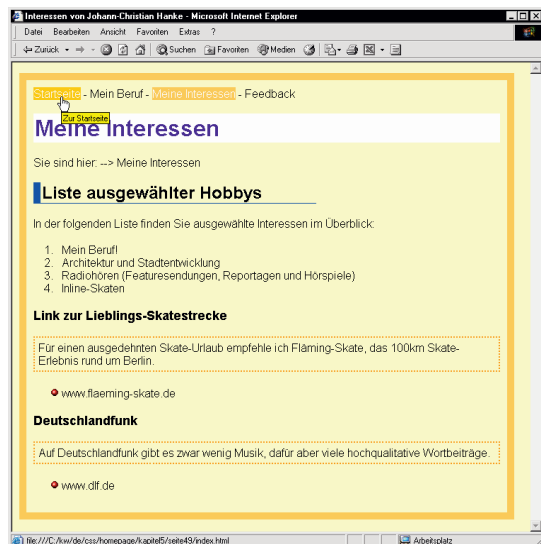
Um die Datei `standard.css` einzubinden, notierst du also in allen drei HTML-Dateien:

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

Die Eigenschaft `href` steht wieder für *hyper reference*, Querverweis. Du findest den Stand dieses Kapitels unter `kapitel5/stand2`. Und jetzt erkläre ich dir, was sich dahinter verbirgt.

## So erzeugst du dynamische Querverweise mit hover

Schau dir dein Dokument einmal an. Hier zeige ich dir als Beispiel die Seite `hobby.html`:



**Sieht ganz gut aus: „hobby.html“ mit externem Style Sheet.**

Besprechen wir nun die Stil-Befehle im einzelnen! Die Anweisungen in BODY und DIV sind sicher verständlich. Wie der dicke blass-orange Rahmen erzeugt wird, ist nach den Experimenten der Vorseiten sicher ebenfalls kein Rätsel mehr für dich.

Dass ich die Schriftart diesmal im DIV angebe und nicht im BODY, ist nur eine Rücksichtnahme auf den Uraltbrowser Netscape 4. Dieser „Trauer-Browser“ würde Arial nicht zuweisen, weil er die Vererbung von body auf div und untergeordnete Elemente nicht erkennt.

Auch die Stile für h1 und h2 müssen wir nicht weiter besprechen, das haben wir schon ausgiebig im vorigen Kapitel getan.

### Punktierten Rahmen erzeugen

Für den punktierten Rahmen habe ich mir den Klassennamen `dotbox` ausgesucht. Es handelt sich im Beispiel um eine ungebundene Klasse.

Selbstverständlich darfst du nicht vergessen, den zu „punktierenden“ Absätzen mit

```
<p class="dotbox"> ... </p>
```

dieses Format auch zuzuweisen. Tue es jetzt! Auf meiner `index.html` handelt es sich um einen Absatz, auf meiner `hobby.html` sind es zwei und auf der `beruf.html` ist es einer.

### Aktuelle Seite hervorheben

Ganz oben in der rechten Spalte habe ich über den Selektor `b` alle fett formatierten Passagen mit dem gleichen Orange-Ton gefärbt, der für den Rahmen verwendet wurde.

Doch schaue einmal zum letzten Selektor in der linken Spalte, zu `span.aktiv`. Dieser Befehl sorgt, wie du weißt, für die Hervorhebung der „aktuellen Seite“. Vergleiche ruhig z.B. mit der Linkleiste auf der Hobby-Seite. Die über


```
<span class="aktiv">Meine Interessen</span>
```

markierte „aktuelle Seite“ wird ähnlich hervorgehoben wie ein aktiver Link. Und da sind wir schon beim Stichwort, bei den „hover-Links“.

### Fünf Pseudo-Klassen für Links

Während man mit einfachem HTML nur die Farben der Links bestimmen kann, ist mit CSS noch viel mehr möglich! Zur Gestaltung von Links nimmst du den Selektor `a`.

Zuerst habe ich über eine allgemeine `a`-Regel die Unterstreichung abgeschaltet (`text-decoration: none;`) und die Links schwarz gefärbt. Das gilt dank unserer Vererbung für alle Links. Doch es gibt schließlich vier Link-Zustände. Hier siehst du alle vier dieser so genannten Pseudo-Klassen:

- `a:link` ist der noch unbesuchte Hyperlink
- `a:visited` ist der besuchte Hyperlink
- `a:hover` ist der Hyperlink während des Darüberfahrens mit der Maus
- `a:active` ist der aktive, gerade angeklickte Link
- `a:focus` ist der Link beim Durchsteppen mit der -Taste

Wichtig ist genau diese Reihenfolge: Halte sie ein, damit es keine Anzeigeprobleme gibt!

Pseudo-Klassen heißen sie, weil diese Klassen nur im Zusammenhang mit Links (Selektor `a`) funktionieren. Schreibe `a:` und dann den Klassennamen.

Im Beispiel habe ich die „hover-Links“ mit der Hintergrundfarbe Orange und Textfarbe Weiß versehen, die besuchten Links färbte ich grau. Die anderen habe ich unberücksichtigt gelassen.

## Aufzählungspunkte mit Bildchen und Liste als Menü

### Individuelles Aufzählungszeichen

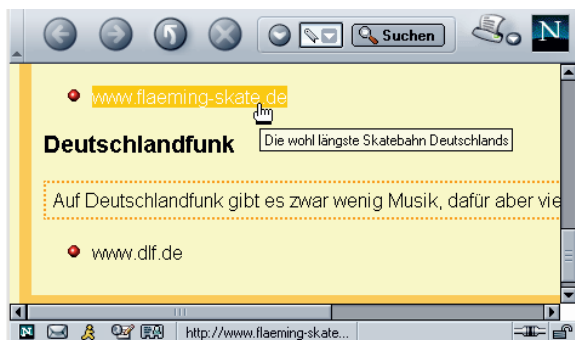
Nun haben wir die wichtigsten Stilanweisungen besprochen. Fehlt noch die *unordered list*, denn hier gibt es eine Besonderheit zu besprechen. Baue ein individuelles Aufzählungszeichen ein!

```
ul {
  list-style-image: url(kreis.gif);
}
```

Die generelle Syntax sieht so aus:

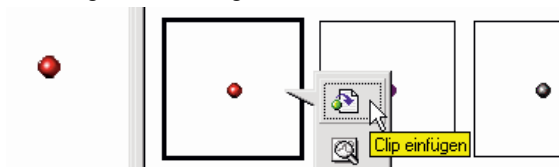
```
list-style-image: url(Pfad/Name.end);
```

Im Beispiel dient eine kleine GIF-Grafik für diesen Zweck. Ich habe sie *kreis.gif* genannt und im gleichen Ordner abgelegt.



### Individuelles Aufzählungszeichen mit `list-style-image`.

Doch woher bekommst du so eine winzige Grafik? Ich habe sie im Beispiel als Microsoft Word bezogen, aus der ClipArt-Gallery. Wähle in Word **EINFÜGEN | GRAFIK | CLIPART**. Die so genannte „ClipArt Gallery“ erscheint. Tippe in die Suchleiste (Bereich *Clips suchen*) das Wort *web* und drücke **Enter**. Wie du siehst, hat Word ein Herz für Web-Designer und zeigt dir unzählige Buttons und Trennlinien an!



### Die „Gallery“ von Word bietet unzählige Web-Grafiken.

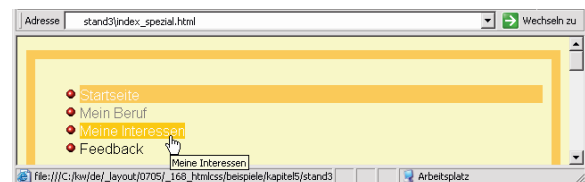
Du hast deinen Clip in Word eingefügt? Das Dokument auf die gewünschte Größe gezoomt? Dann drücke die Taste **Druck**, um ein Bildschirmfoto des Schirms in die Zwischenablage zu legen (= Screenshot anfertigen!) Rufe jetzt dein bevorzugtes Bildbearbeitungsprogramm auf, füge den Inhalt der Zwischenablage über **BEARBEITEN | EINFÜGEN** bzw. **Strg + V** ein und schneide das Bildchen aus. Jetzt brauchst du es nur noch als *kreis.gif* zu sichern.

### Auch das geht: Menüs als Liste

Wollen wir einmal etwas ganz Spannendes wagen? Ein Experiment? Wir fügen das Menü in eine Liste ein. Dabei ersetzen wir das alte Menü wie folgt:

```
<ul class="menu">
  <li class="aktiv">Startseite</li>
  <li><a href="beruf.html" title="Mein Beruf">Mein Beruf</a></li>
  <li><a href="hobby.html" title="Meine Interessen">Meine Interessen</a></li>
  <li><a href="mailto:css@lexi.de" title="Schreiben Sie mir!">Feedback</a></li>
</ul>
```

Das ist durchaus sinnvoll, da ein Menü vom Logischen her sowieso eine Liste ist! Ich empfehle es dir!



**Das sieht aber noch nicht perfekt aus! Die Liste erbt die kleine Aufzählungsgrafik und alles steht untereinander.**

Doch wie schaffen wir es, dass die Listenpunkte nicht untereinander, sondern nebeneinander dargestellt werden. Und bitte ohne Aufzählungszeichen? Schau dir das wie folgt ergänzte Style Sheet an:

```
.menu {
  padding: 0; margin: 0;
}

.menu li {
  display: inline;
  list-style-type: none;
  margin-right: 5px;
}
```

Das Geheimnis steckt in der unteren Regel. Sie wirkt dank des Selektors `.menu li` erst einmal auf alle `li`-Elementen innerhalb von `<ul class="menu">`. Die Deklaration `display: inline;` schaltet dann die Blockdarstellung der Listenelemente aus und ordnet die Menüpunkte auf diese Weise nebeneinander an. Und mit `list-style-type: none;` wird das Aufzählungszeichen deaktiviert. Das ist der Trick! Der Rest ist Kosmetik!



**Zum Vergleichen: Diese Fassung findest du unter dem Pfad `kapitel5/stand3` in der Datei `index_spezial.html` bzw. in der CSS-Datei `standard_spezial.css`.**

## Profi-Wissen: Hinweise zu externen Style Sheets

Hier einige Profi-Hinweise zum Einbinden und Kombinieren von internen und externen Stilanweisungen und zum Verstecken von Elementen.

### CSS extern und intern parallel?

Kann man interne und externe CSS-Anweisungen parallel betreiben? Na klar! Nimm ein zentrales Style Sheet für deine ganze Site. Wünschst du auf einer speziellen HTML-Seite explizit andere Stilanweisungen? Dann überschreibst du die Stile aus der übergeordneten externen Stilanweisung einfach, und zwar per Stilbereich im HEAD. Oder du definierst zusätzlich neue Stile für die Elemente, die speziell in dieser HTML-Datei Verwendung finden.

Beachte auch hier das Kaskadenprinzip. Setze zuerst den Link zur CSS-Datei. Die lokalen Stilanweisungen werden dagegen in der herkömmlichen Art und Weise darunter notiert.

Die Syntax sieht folgendermaßen aus:

```
<link rel="stylesheet"
type="text/css" href="Pfad/Name.css">
<style type="text/css">
lokale Stilanweisungen
</style>
```

Selbstverständlich steht es dir auch hier frei, mit Inline-CSS zu arbeiten. Formatiere die Tags meinetwegen auch direkt per style-Eigenschaft.

### Mehrere Style Sheets einbinden

Niemand hindert dich daran, mehrere CSS-Dateien einzubinden. In einer legst du alle Schrifteigenschaften, in der anderen dagegen die Breiten und Rahmeneffekte fest. Notiere sie einfach untereinander.

```
<link rel="stylesheet" type="text/css"
href="Pfad/Name1.css">
<link rel="stylesheet" type="text/css"
href="Pfad/Name2.css">
```

Auch hier solltest du die richtige Reihenfolge beachten. Wenn sich zwei Regeln um die Vorherrschaft streiten, gewinnt die, die zuletzt aufgerufen wurde.

### Druck-Style-Sheet einbinden

Noch ein sinnvoller Trick: Binde ein Style Sheet für die Anzeige am Bildschirm und ein separates für den Ausdruck auf Papier ein. Das geht ganz einfach, ergänze lediglich das Attribut `media`. Vergib dann je nach Wunsch den Wert `print` (für das

Druck-Style-Sheet) oder `screen` (für die Anzeige am Bildschirm). Nun gelten diese Style-Sheet-Dateien nur für ihren jeweiligen Einsatzzweck:

```
<link rel="stylesheet" type="text/css"
href="druck.css" media="print">
<link rel="stylesheet" type="text/css"
href="layout.css" media="screen">
```

Ein Beispiel dafür findest du auf [www.cmbasic.de](http://www.cmbasic.de).

### ■ Elemente ausblenden

Rufe mal die Druckvorschau bei [www.cmbasic.de](http://www.cmbasic.de) auf. Und tatsächlich – dank des Druck-Style-Sheets blende ich das Logo und die Menüs weg. Denn die stören im Ausdruck. Das kannst du auch! Zum Ausblenden von Elementen verwendest du einfach nur die CSS-Eigenschaft `visibility`.

Eigenschaft	Wert	Erläuterung
<code>visibility</code>		Sichtbarkeit
	<b>visible</b>	Voreinstellung, Element sichtbar
	<b>hidden</b>	Element wird versteckt

Du möchtest beispielsweise ein DIV ausblenden, welches der Klasse `menu` angehört? Notiere einfach im speziellen Druck-Style-Sheet (und nur da!):

```
div.menu {
    visibility: hidden;
}
```

### Die @import-Syntax für CSS

Es gibt übrigens noch eine andere Methode, externe CSS-Dateien einzubinden. Und zwar direkt mit den Sprachmitteln von CSS. Nimm den Befehl `@import url(Pfad/Name.css);`

Die komplette Syntax mit `<style>`-Tags sieht folgendermaßen aus:

```
<style type="text/css">
@import url(Pfad/Name.css);
</style>
```

Früher hat man diese Methode verwendet, um Style Sheets vor dem Browser Netscape 4 zu verstecken. Denn dieser verstand die `@import`-Syntax nicht. Man hat einfach zuerst ein Style Sheet auf die normale Art und Weise eingebunden – mit allerlei Tricks für den alten Netscape-Browser. Danach baute man das „richtige“ Style Sheet über `@import` ein. Dort wurden all die trickreichen Regeln überschrieben. Für den Netscape-Browser jedoch blieb dieses normale Style Sheet unsichtbar.

## ✓ Übungsteil E: Übungen zu externen Style Sheets

Du weißt jetzt, wie man:

- DIV- und SPAN-Container nutzt
- hover-Effekte für Hyperlinks einsetzt
- externe Style Sheet-Dateien erstellt und in HTML-Seiten einbindet



### ■ Übung E1: Externe CSS-Datei erstellen

Gehe in den separaten Ordner `aufgaben`. Hast du die einfache Übung B1 gemacht? Sehr gut! (Falls nicht, blätterst du schnell zur Seite 24 und holst es nach!) Erstelle von den drei Dateien im Ordner `b1` eine Kopie. Lege diese in einen neuen Ordner namens `e` (einfach nur `e` ohne Zahl). Erstelle im `e`-Ordner eine externe CSS-Datei namens `leben.css`. Gib als einzige Anweisung für alle Elemente die Schriftarten Arial, Helvetica bzw. eine allgemeine serifenlose Schriftart an.

### ■ Übung E2: Link auf externe CSS-Datei und SPAN-Container

Binde diese CSS-Datei nun per Link auf allen drei Seiten (`frage.html`, `ja.html`, `nein.html`) ein. Gehe in die `ja.html`. Vergrößere das Wort *schön* per SPAN und Inline-CSS auf 25 Punkt!

### ■ Übung E3: Link-Stile mit hover-Effekte versehen

Erweitere deine CSS-Datei. Baue folgende Anweisungen für hover-Links ein:

- Die normalen (`a:link`) und aktiven Links (`a:active`) sollen grün und *nicht unterstrichen* dargestellt werden
- Nur während des Darüberfahrens (`a:hover`) soll der Link unterstrichen erscheinen (`text-decoration: underline;`), mehr nicht.
- Die besuchten Links sollen rot und wiederum nicht unterstrichen erscheinen.

Das Leben ist **schön!**  
[ [Zurück zur Startseite](#) ]

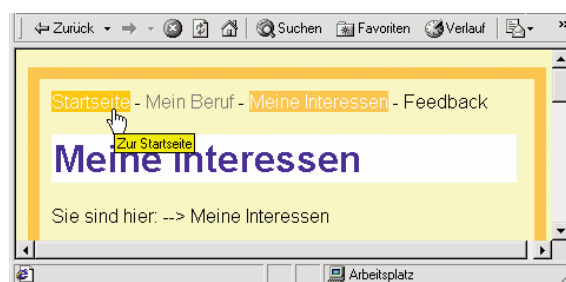
### ■ Übung E4: Probleme mit hover-Links lösen

Probiere die Links aus. Was stellst du fest, nachdem du einen Link angeklickt hattest? Klappt alles? Glückwunsch, dann hast du die richtige Reihenfolge beachtet. Bei dir wird die Unterstreichung unterdrückt, egal ob du „hoverst“ oder nicht? Denke an das Kaskadenprinzip! Richte die Reihenfolge der Link-Stile so ein, dass auch besuchte Links nur bei „hover“ mit einer Unterstreichung versehen werden. Es gibt nur eine Reihenfolge, die hier keine Probleme bereitet.

### ■ Übung E5: Relative Breite mit `width: auto`

In der Gemeinde der Webgestalter gibt es einen Streit: Absolutes, pixelgenaues Layout versus Layout mit relativer Breite. Während die meisten Gestalter feste Breiten bevorzugen, warnen andere davor, das „Browserfenster“ zu vergewaltigen.

Zurück zum Projekt im Ordner `eigenheim`. Bisher hatten wir im Selektor `div.rand` mit der Angabe `width: 700px;` eine Breite von genau 700 Pixeln erzwungen. Stelle zur Probe eine relative Breite ein.



Die Seite passt sich der Breite des Browserfensters an

Die Eigenschaft `width` kennt auch eine relative Breitenangabe. Gib statt eines Pixel-Wertes einfach den Eigenschafts-Wert `auto` an. Oder wähle eine Angabe wie z.B. `95%`. Wichtig: Teste es unbedingt mit dem Internet Explorer und mit Firefox. Beide Browser kommen bei Prozentangaben zu unterschiedlichen Interpretationen.



## Kapitel 6: Das Projekt – Spaltensatz mit Kopf- und Fußzeile

Schön, dass du bis hierher durchgehalten hast. Nun kannst du dich schon als angehenden Stil-Experten betrachten! Jetzt kommen wir zum interessantesten Teil dieses Hefts. Es dreht sich um attraktives Layout und vor allem um Spaltensatz ohne Tabellen.

Bis vor kurzem galt: Attraktive, mehrspaltige Seiten lassen sich nur mit unsichtbaren Tabellen aufbauen. Und tatsächlich folgen die meisten Webportale immer noch diesem Grundsatz. Egal ob [spiegel.de](http://spiegel.de) oder [amazon.de](http://amazon.de) – überall findest du die Tabelle – häufig sind sogar mehrere Tabellen ineinander verschachtelt. Schluss mit dieser Zweckentfremdung! Tabellen wurden nicht geschaffen, um als unsichtbare Platzhalter missbraucht zu werden. Immer mehr Webmaster erkennen die Vorteile von barrierearmen, tabellenfreien Seiten, die von Style Sheets in Form gehalten werden. Der Quellcode wird viel schlanker, die Seite lädt schneller. Sie kann auch von behinderten Menschen besser gelesen werden, da die Struktur klarer ist. Außerdem ist es viel einfacher, das Layout zu verändern – eine neue CSS-Datei genügt! Alle aktuellen Browser machen's mit, also worauf wartest du noch?

### ■ Dave Shea's CSS Zen Garden

Du möchtest Schönheit und Vorzüge von CSS live erleben? Besuche [www.csszengarden.com](http://www.csszengarden.com) – ein Projekt des Webdesigners Dave Shea aus Canada.



Wähle rechts das gewünschte Design und schon ändert sich die Seite komplett. Die Änderung wird einzig und allein durch eine andere CSS-Datei bewirkt! Achtung: Die Layouts sind geschützt und dürfen nicht einfach 1:1 kopiert werden! Abgucken ist aber erlaubt.

### Das Projekt „Ferienhaus“

Und jetzt ran an unser Projekt. Im Beispiel geht es um die Vermarktung eines Produkts (hier: Ferienwohnungen). Der Besucher soll sich über Produkt, Preis und Verfügbarkeit informieren und wenn sie oder er will ... auch gleich bestellen (bzw. buchen) können. Selbstverständlich gibt es eine Feedback-Möglichkeit für Fragen, diesmal sogar in Form eines Formulars.

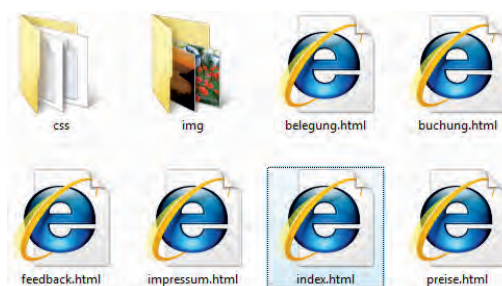
Damit du das Beispiel auch dann nutzen kannst, wenn du zufällig gerade keine kleine Firma bist, habe ich der Musterseite einen „Vereins-Anstrich“ verpasst und die Textmenge (= Schreibarbeit) auf das Allernötigste reduziert. Auch das Layout kannst du – wenn du das Prinzip einmal begriffen hast – problemlos abändern.

### ■ Die Ordnerstruktur

Fangen wir mit der Planung der Ordner- und Dateistruktur an! Das Projekt besteht aus sechs HTML-Seiten. Die Startseite ist die HTML-Datei `index.html`, das ist klar. Die übrigen Seiten sollen `belegung.html`, `preise.html`, `buchung.html`, `feedback.html` und `impressum.html` heißen.

Alle HTML-Seiten legen wir in einen Ordner namens `ferienhaus`. Richte dir diesen Ordner (unter deinem „Hauptordner“ `homepage`) schon ein!

Außerdem sehen wir je einen weiteren Unterordner namens `img` für die Grafiken und einen Ordner namens `css` für die CSS-Datei vor.



### Diese Ordner und Dateien planen wir.

Das Bild zeigt dir alle Dateien im Stammordner `ferienhaus`. Unter `css` lagern wir neben der CSS-Datei auch eine „Aufzählungszeichen-Grafik“.

Im `img`-Ordner legen wir die übrigen „Haupt-Grafiken“ ab. Ich habe sie in der Größe optimiert. Faustregel: Kein Bild größer als 35–55 kByte!

## Das Layout im Überblick

Und hier zeige ich dir das fertige Projekt schon einmal vorab. Es ist eine stabile und bewährte CSS-Struktur, die du beliebig anpassen kannst! Ich habe übrigens dafür gesorgt, dass die Seite auch im Internet Explorer 5 eine gute Figur macht.



**Zugegeben – so schön wie der CSS Zen Garden sieht es nicht aus. Aber für den Anfang ist es doch schon ganz brauchbar!**

### ■ Pixelgenaues Layout für 800 x 600

Die Seite besteht im Beispiel aus einer orangen Kopfzeile, einer hellgrauen Fußzeile und zwei Spalten. Sie ist zentriert ausgerichtet und „schwebt“ in der Mitte des Browserfensters. Der Hintergrund ist dunkelgrau, damit sich die weiße Seite besonders gut abhebt. Im linken Bereich wird die schon besprochene Linkleiste mit Link-Buttons vorgesehen. Diese besitzt eine Breite von 160 Pixeln und wird durch eine Liste erzeugt. Insgesamt ist die Seite 780 Pixel breit. Die Höhe ist nicht so entscheidend, schließlich kann der Betrachter die Seite rollen.

Unser Projekt zielt damit auf pixelgenaues Layout für Bildschirmauflösungen der Größe 800 x 600! Und das ist ein guter Kompromiss.

Plane auf gar keinen Fall Seiten, die nur unter einer Bildschirmauflösung von 1024 x 768 Pixeln funktionieren. Vergiss nicht, dass es immer noch Betrachter mit alten Monitoren und kleinen Anzeigegegeräten (Handheld-PCs, TV-Set-top-Boxen) gibt.

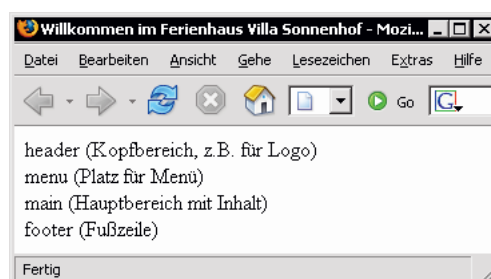
## Schritt für Schritt zum Erfolg

Zuerst bauen wir die HTML-Datei auf. Schreibe den Quellcode ab – und zwar mitsamt aller Platzhaltertexte wie *header* (Kopfbereich, z.B. für Logo) usw. Diese Platzhalter ersetzen wir später durch die Inhalte – aber erst, nachdem wir die Grundstruktur der Seite aufgebaut haben. Speichere das Ganze unter *index.html* im Ordner *ferienhaus*:

```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1">
<title>Willkommen im Ferienhaus Villa
    Sonnenhof</title>
<link href="css/struktur.css"
    rel="stylesheet" type="text/css">
<link href="css/layout.css"
    rel="stylesheet" type="text/css">
</head>
<body>

<div id="mother">
    <div id="header">
        header (Kopfbereich, z.B. für Logo)
    </div>
    <div id="menu">
        menu (Platz für Menü)
    </div>
    <div id="main">
        main (Hauptbereich mit Inhalt)
    </div>
    <div id="footer">
        footer (Fußzeile)
    </div>
</div>

</body>
</html>
```



**Bisher sieht die Seite so aus. Zum Vergleichen: Du findest den aktuellen Stand auch unter dem Namen „index\_ohne\_css.html“ im Ordner „kapitel6/ferienhaus“. Die CSS-Links habe ich dort deaktiviert.**

### Links auf Style-Sheet-Dateien

Im Kopfbereich habe ich schon die Links auf zwei noch zu erstellende CSS-Dateien eingefügt. Es sind zum einen die `struktur.css` und zum anderen die `layout.css`. Beide Dateien liegen im Unterordner `css`. Die `struktur.css` kümmert sich um die Anordnung, Ausrichtung, Größe und Einfärbung der DIV-Container. Sie legt also die Grundstruktur fest. Die entsprechenden Schriftformate werde ich in der `layout.css` festlegen.

Das ist zwar nicht zwingend erforderlich. Auf diese Weise Sorge ich jedoch für mehr Übersicht und Klarheit!

### ■ Link zum Unterordner

Im Gegensatz zu Pfaden im Windows-Dateisystem arbeitest du im Web mit dem Schrägstrich, dem einfachen Slash (/). Das weißt du. Außerdem verwenden wir so genannte „relative Pfadangaben“. Schließlich kennst du ja den exakten Speicherort deiner Datei auf dem Server nicht.

Gehe stets von deinem Stammordner aus!

Wenn du vom Stammordner aus auf die Datei `struktur.css` im Unterordner `css` verweist, sieht die Pfadangabe also so aus: `css/struktur.css`!

### DIV-Container mit Eigenschaft `id`

Im Beispiel benötigen wir fünf DIV-Container. Jeden DIV-Container habe ich durch die Eigenschaft `id` (für *identifizieren*) eindeutig benannt. Was ist der Unterschied zwischen `class` und `id`? Könntest du nicht auch schreiben `<div class="mother">...</div>` oder `<div class="header">...</div>`? Klar, auch das wäre möglich. Doch eine `id` hat Vorteile. Im Gegensatz zu einer Klasse darf sie nur ein einziges Mal vorkommen und ist daher bestens für Elemente geeignet, die ebenfalls nur ein einziges Mal vorkommen:

```
<div id="mother">...</div>
<div id="header">...</div>
```

Im Style Sheet wiederum verweist du folgendermaßen auf solche „id-DIV-Container“ – wichtig ist die Raute:

```
#id { Stilanweisungen; }
```

Schreibe also: `#mother {}` bzw. `#header {}`.

Soweit ein paar Neuerungen schon vorab. Der Rest folgt später.

### ■ Die Aufgaben der fünf DIV-Container

Der erste unserer fünf Struktur-Container bekommt im Beispiel die `id mother`.

```
<div id="mother">
```

Dieser `mother`-Container wickelt sich um alle anderen drumherum. Wir benötigen ihn im Beispiel, um die exakte Breite festzulegen und um die Seite in die Mitte des Browserfensters zu rücken.

Die anderen vier Container werden den Inhalt enthalten. Ich habe sie sinnvoll benannt. Mit `header` beziehe ich mich auf den Kopfbereich, `menu` kennzeichnet das Menü, `main` ist die Hauptspalte und `footer` verweist auf die Fußzeile. Die Namen sind natürlich nur Vorschläge, du kannst sie fast beliebig anpassen. (Verzichte jedoch auf Umlaute, Leer- und Sonderzeichen.)

#### Vergib logische Namen für deine Container!

Ich rate davon ab, Namen wie `linkeSpalte` oder `rechteSpalte` zu vergeben. Damit engst du dich zu sehr ein. Denn du weißt ja nicht, ob du das Menü nicht später mal von der linken auf die rechte Seite schieben möchtest. Mit CSS ist das doch kinderleicht! Und es wäre unlogisch, wenn diese rechte Spalte dann weiterhin `linkeSpalte` hieße. Allgemeine Bezeichnungen wie `main`, `content`, `inhalt`, `menu`, `menueleiste`, usw. sind da meist sinnvoller.

### Die CSS-Datei „struktur.css“

Als Nächstes präsentiere ich dir den Quellcode der ersten CSS-Datei. Mach mit! Erstelle eine leere Datei namens `struktur.css` und lege sie im Unterordner `css` ab. Auf den nächsten Seiten nehmen wir diese Datei dann haarklein auseinander:

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

body {
    font: 100% Verdana, Arial, Helvetica, sans-serif;
    text-align: center;
    background-color: gray;
    margin-top: 10px;
}

#mother {
    width: 780px;
    text-align: left;
    background-color: white;
    margin: 0 auto;
```

```

border: 5px solid white;
}

#header {
  height: 80px;
  background-color: #ff9900;
}

#menu {
  float: left; /* schwebt links */
  width: 160px;
  background-color: #ffff99;
  padding: 20px 5px;
}

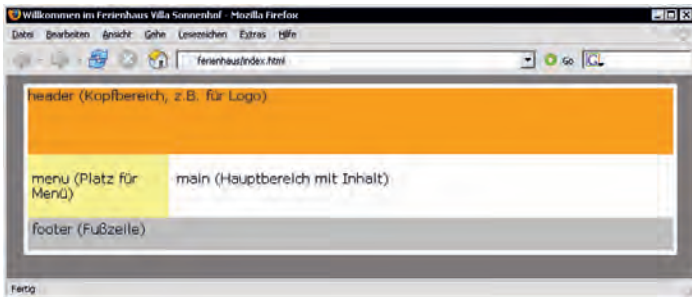
#main {
  background-color: white;
  padding: 20px 10px;
  margin: 0 0 0 170px;
}

#footer {
  clear: both;
  height: 30px;
  padding: 5px;
  background-color: silver;
}

```

### Die CSS-Datei im Überblick

Alles abgeschrieben und gespeichert? Dann sieht deine Seite namens `index.html` erst einmal so aus:



**Die Grundstruktur ist fertig: Alle Container sind korrekt angeordnet und eingefärbt.**

Damit ist der Seitenaufbau schon fertig! Du hast eine attraktive und schicke Seite mit Rahmen- und Farbeffekten erzeugt.

Doch nun werfen wir einen Blick auf die CSS-Datei und klären das, was dir noch unklar ist.

### ■ Rahmen und Ränder unterdrücken per Asterisk \*

Du staunst über die erste Regel mit dem komischen Sternchen? Dieser Selektor heißt Asterisk und wählt alle Elemente aus:

```

* {
  padding: 0;
  margin: 0;
}

```

Und bei allen Elementen entferne ich erst einmal die Innen- und Außenränder. Wozu?

Die gängigen Browser verwenden, wie du längst weißt, ihr eigenes, eingebautes Style Sheet. Sie stellen, falls nicht anders angegeben, Überschriften, Absätze oder auch Listen nicht nur mit ihren eigenen Schriftgrößen, sondern auch mit voreingestellten Randabständen dar. Und das kann ganz schön stören! Mit dieser Stilregel sorgen wir erst einmal dafür, dass überall die voreingestellten Innen- und Außenränder abgeschaltet werden. Und zwar tatsächlich bei jedem Element. Das ist sehr praktisch, denn jetzt können wir viel ungestörter unsere individuellen Maße verwenden.

Natürlich könnte man diese Innen- und Außenränder auch in jedem Element separat einstellen. Doch die Sternchen-Syntax spart viel Schreibarbeit.

### ■ Die Regel für body

Als Nächstes schauen wir uns die `body`-Regel an:

```

body {
  font: 100% Verdana, Arial, Helvetica, sans-serif;
  text-align: center;
  background-color: gray;
  margin-top: 10px;
}

```

Die erste Zeile legt die Schriftgröße und die Schriftart fest. Wir benutzen – fortgeschritten wie wir nun mal sind – CSS-Steno. Dir ist die Syntax nicht geläufig? Dann blättere schnell noch einmal zur Seite 34. Als Einheit empfehle ich diesmal Prozent – also eine relative Einheit. Das ist besser, da sich Schriften mit relativen Schriftgrößen auch im Internet Explorer 5 und 6 skalieren lassen. Und zwar mit dem Mausekranz bei gedrückter `[Strg]`-Taste.

Wozu dient `text-align: center;` in Zeile zwei? Das ist ein Zugeständnis an den Internet Explorer 5. Der schafft es sonst nicht, die Seite in die Mitte zu rücken. Weiter unten müssen wir dann daran denken, `text-align` wieder auf `left` zu schalten.

Das Festlegen der Hintergrundfarbe ist klar? In Zeile drei färben wir das gesamte Browserfenster erst einmal grau ein!

Besonders pfiffig ist die letzte Zeile. Hier füge ich einen Rand von 10 Pixeln zwischen oberer Fensterkante und dem Seitenbereich ein. Dann klebt der Inhalt nicht so am Browserfenster – das wirkt luftiger. Würdest du diese Zeile weglassen, würde die Seite direkt ohne Luftspalt am oberen Rand des Browserfensters beginnen.

### ■ Der Container „mother“

Die ersten drei Zeilen der *mother*-Regel sind dir sicher klar. Wir legen die Breite exakt fest auf 780 Pixel, setzen die Textausrichtung wieder auf links und entscheiden uns für die Hintergrundfarbe weiß. Doch was macht diese Zeile?

```
margin: 0 auto;
```

Sie setzt den oberen und unteren Rand auf 0. Der linke und rechte Rand jedoch wird automatisch bestimmt. Da die Seite 780 Pixel breit ist, rutscht sie dadurch genau in die Mitte der Seite. Das ist also der Trick, um ein DIV in die Mitte zu rücken. Da der Internet Explorer 5 diesen Trick jedoch nicht kennt, hatten wir in *body* zusätzlich die Regel *text-align: center* gesetzt.

Und was bewirkt *border: 5px solid white;*? Das ist Rahmensteno (siehe Seite 45). Der Container erhält rundherum einen durchgezogenen, 5 Pixel dicken weißen Rand. Sieht doch gut aus, oder?

### ■ Regel für „header“

Der Header bereitet keinerlei Probleme. Wir ziehen die Höhe auf 80 Pixel und legen als Hintergrundfarbe *orange* (*#ff9900*) fest. Irgendwelche Innenabstände definieren wir diesmal nicht. Denn wenn du ein Logo in diesem Container platzieren willst, sollte dieses gleich an der linken und oberen Kante beginnen.

Im Beispiel werden wir jedoch später eine Überschrift in den Header setzen. Diese wird dann separat mit Abständen versehen, damit sie nicht so an der Kante klebt.

### Das schwebende Menü

Als Nächstes folgt das Menü. Tragen wir einmal zusammen, was wir schon verstehen:

```
#menu {
  float: left; /* schwebt links */
  width: 160px;
  background-color: #ffff99;
  padding: 20px 5px;
}
```

Der DIV-Container für das Menü ist 160 Pixel breit und verwendet einen Gelbton als Hintergrundfarbe. Der obere und untere Innenrand beträgt 20 Pixel, der linke und rechte nur 5 Pixel. Damit sorgen wir dafür, dass der Inhalt nicht so dicht an den Rand geklatscht wird. Vor allem die 20 Pixel Luft von oben sind sinnvoll. Das werden wir in der Hauptspalte auch so machen!

### ■ float: left

So weit so gut. Doch was macht *float: left*? Dieses Attribut-Werte-Paar bringt den DIV-Container zum schweben. Schließlich heißt *to float* schweben. Der Container schwebt wie ein Ballon so weit nach links (und oben), wie es in der ihn umgebenden Box *mother* möglich ist. Er wird also linksbündig ausgerichtet und stößt oben an den *header*-Container. Und da der *menu*-Container selber nur 170 Pixel (zur Erinnerung: 160 Pixel *width* + 2 x 5 Pixel *padding*) breit ist, haben wir daneben noch genug Platz für einen weiteren Container mit der Hauptspalte! Genial, nicht wahr?

Die Höhe des Containers legen wir nicht fest. Sie ergibt sich aus der Höhe des enthaltenen Menüs.

Übrigens kannst du *float* auch verwenden, um Grafiken links- oder rechtsbündig auszurichten. Dann passiert genau das gleiche wie früher mit *align="right"* bzw. *align="left"*. Nur eben mit modernerer Syntax in CSS.

### ■ float und clear

Du möchtest, dass eine folgende Box nicht mehr neben der gefloateten, sondern unter dieser beginnt? Dann musst du das Attribut *clear* verwenden. Das haben wir – rein zur Sicherheit – im *footer* berücksichtigt: *clear: both;*

Damit sorgen wir dafür, dass diese Fußzeile auf jeden Fall unterhalb des Menüs (und unterhalb der Hauptspalte) beginnt.



Hier ein Überblick über das Attribut `float` und das Gegenstück `clear`.

Eigenschaft	Wert	Erläuterung
<code>float</code>		Textumfluss (Schwebezustand)
	<code>none</code>	Voreinstellung, kein Schweben und kein Textumfluss
	<code>left</code>	Element steht links, darunter notierter Inhalt fließt rechts vorbei
	<code>right</code>	Element steht rechts, darunter notierter Inhalt fließt links vorbei
<code>clear</code>		Beendigung des „float“-Textflusses
	<code>left</code>	Element steht unterhalb der mit <code>clear: left</code> gefloateten Boxen
	<code>right</code>	Element steht unterhalb der mit <code>clear: right</code> gefloateten Boxen
	<code>both</code>	Element beginnt unter dem umflossenen Element, egal ob dieses rechts oder links schwebt

### Der Trick mit der Hauptspalte

Schauen wir uns nun den Code der Hauptspalte an.

```
#main {
  background-color: white;
  padding: 20px 10px;
  margin: 0 0 0 170px;
}
```

Die ersten zwei Zeilen sind sicher kein Problem: Hintergrundfarbe weiß? Das ist klar! Auch der obere und untere bzw. linke und rechte Innenrand erschließt sich dir sofort. Wir wollen, dass der Inhalt nicht so dicht an der Wand klebt – genau wie im DIV für das Menü.

Aber warum so ein großer linker Rand? Warum wird dieses DIV nicht auch „gefloated“? Beispielsweise mit `float: right`? Tatsächlich – das wäre

möglich! Dann müssten wir die Breite zusätzlich auf ca. 600 Pixel beschränken und es würde prima funktionieren. Der „menu-Ballon“ bleibt weiterhin links, der „main-Ballon“ schwebt dagegen nach rechts und zwischen beiden ist ein wenig Luft.

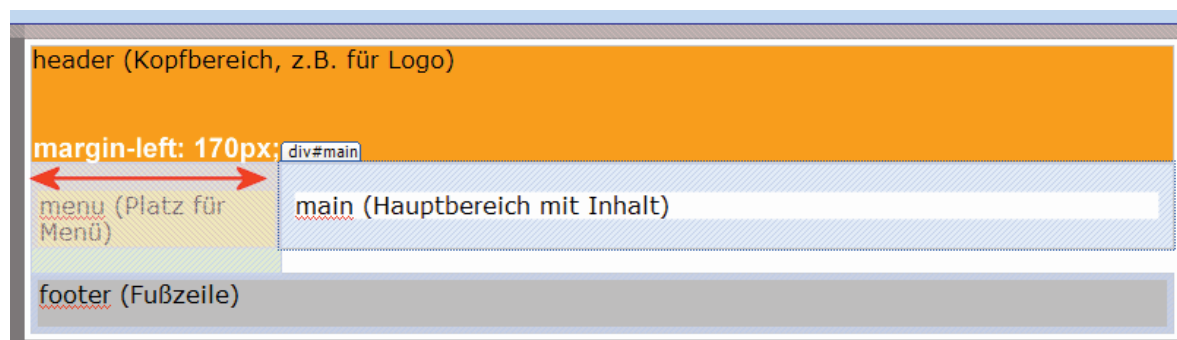
Leider bereitet diese Variante Probleme im Internet Explorer. Wenn der Inhalt in der Hauptspalte zu breit wird – beispielsweise durch eine breite Tabelle oder ein Bild mit Überbreite – rutscht der rechte Bereich häufig nach unten weg. Neben dem Menü ist dann einfach nicht mehr genug Platz.

Der Trick: Wir verzichten auf den Textfluss mit `float` und geben lediglich einen großen linken Rand von 170 Pixel an: `margin: 0 0 0 170px`; Und das sind genau die 170 Pixel, die unser Menü an Platz benötigt. So haben wir für den Hauptinhalt einen großen Container, der sich über die gesamte Breite des übergeordneten Elements zieht. Dank des linken Rands von 170 Pixeln bleibt jedoch genügend Freiraum für das schwebende Menü.

#### ■ Vorsicht mit `width: 100%`

Einige Rahmen, beispielsweise der header, der footer oder der main-Container, ziehen sich über die gesamte Breite des übergeordneten DIVs namens `mother`. Warum lassen wir die Breite weg? Warum geben wir ihnen nicht die Eigenschaft `width: 100%`; mit auf den Weg? Weil du nie vergessen darfst, dass zu `width` ggf. auch `margin`, `border` und äußerer Rand dazuaddiert werden.

Im Zweifelsfalle würden diese Container sonst das gesamte Layout sprengen, da diese Werte zusammengezählt meist mehr als 100% ergeben. Glaube mir, ich spreche da aus Erfahrung!



Die Abbildung zeigt, dass sich der main-Container eigentlich über die gesamte Breite des mother-DIVs zieht. Allerdings haben wir links einen Rand von 170 Pixeln gelassen. Platz genug, dass sich das Menü entfalten kann. Aber im Prinzip schwebt das Menü über dem main-Container. Genauer gesagt: über dem Rand des main-DIVs.



## Kapitel 7: Exakt positioniert: Dreispalter mit fixierten DIVs

Im nächsten Kapitel geht es mit dem Ferienhausprojekt weiter. Vorher will ich dir jedoch eine weitere Positionierungstechnik vorstellen: das absolute Positionieren. Denn bisher haben wir unsere DIVs relativ zueinander angeordnet. Sie laufen im Textfluss mit. Nun jedoch wollen wir eine linke und eine rechte Spalte ganz exakt in Bezug auf das Browserfenster positionieren. Wir nutzen es für einen Dreispalter, der sich über die ganze Seite zieht.

### CSS-P für exaktes Positionieren

Nehmen wir ein Beispiel. Du willst einen DIV-Container mit einer ganz bestimmten id (hier box10) exakt 50 Pixel von oben und 100 Pixel von links ausrichten? Die Breite soll 200 Pixel betragen? Dann schreibe im Style Sheet:

```
#box10 {
    position: absolute;
    top: 50px;
    left: 100px;
    width: 200px;
}
```

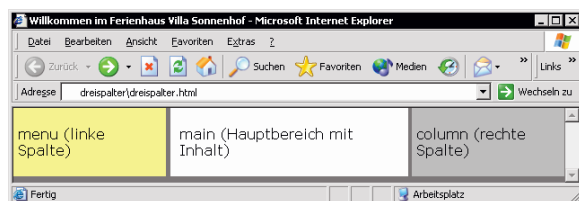
Im HTML-Dokument notierst du diesen Container so: `<div id="box10">Inhalt</div>`

Du kannst mit `height` auch eine Höhe notieren. Ansonsten passt sich die Höhe automatisch an! Außerdem gibt es noch einen Wert `right` für den Abstand von rechts, den wir hier nicht verwenden.

### Eigenschaften und Werte von CSS-P

Mit diesem Beispiel weißt du schon das Wichtigste zum Thema absolute Positionierung. Mit der eben gezeigten Syntax (`position: absolute;`) kann am wenigsten schief gehen.

Die Platzierung bezieht sich dabei auf das jeweils übergeordnete Element. Da das im Beispiel BODY ist, handelt es sich um die obere und linke Browserkante!



**Unser Projekt besteht aus drei Spalten. Die linke und die rechte werden absolut positioniert, die mittlere passt sich in ihrer Breite an.**

### Der HTML-Quellcode

Zuerst zeige ich dir den HTML-Quellcode, der Bereich zwischen `<body></body>` genügt.

```
<body>

    <div id="menu">
        menu (linke Spalte)
    </div>

    <div id="main">
        main (Hauptbereich mit Inhalt)
    </div>

    <div id="column">
        column (rechte Spalte)
    </div>

</body>
```

### Die CSS-Datei

Und hier folgt die CSS-Datei. Ich zeige dir nur die drei Regeln für die Container `menu`, `main` und `column`:

```
#menu {
    position: absolute;
    top: 0;
    left: 0;
    width: 160px;
    background-color: #ffff99;
    padding: 20px 5px;
}

#main {
    background-color: white;
    padding: 20px 10px;
    margin: 0 175px 0 175px;
}

#column {
    position: absolute;
    top: 0;
    right: 0;
    width: 160px;
    background-color: silver;
    padding: 20px 5px;
}
```

### ■ Die Container menu und column

Unsere beiden absolut positionierten DIVs sind der Star des Layouts. Während sich menu ganz eng an den linken und oberen Fensterrand kuschelt, rutscht column ganz nach rechts. Natürlich könntest du die Werte auch variieren. Schreibe beispielsweise top: 80px; wenn du den Container erst 80 Pixel von oben beginnen lassen möchtest.

### ■ Der Haupt-Container #main

Was gibt es zur Hauptspalte Besonderes zu sagen? Nicht viel: Der linke und der rechte Rand betragen je 175 Pixel. Ansonsten klebt der Container zwischen den beiden absolut positionierten DIVs und passt sich in der Breite an.

Du findest dieses Layout im Ordner kapitel7/dreispalter.

## Dreispalter mit Header

Wie wäre es mit einem zusätzlichen Header? Das nächste Layoutbeispiel soll folgendermaßen aussehen – der Header besitzt eine Höhe von 80 Pixeln:



**Dieses Layout besitzt zusätzlich eine Kopfzeile mit 80 Pixeln Höhe. Darunter erst beginnen die beiden absolut positionierten DIVs menu und column und der relativ positionierte main-Container.**

Füge im HTML-Quellcode einfach zusätzlich ein header-DIV ein, und zwar oberhalb der schon vorhandenen DIVs:

```
<div id="header">
  header (Kopfbereich)
</div>
```

### ■ CSS-Datei im Überblick:

Die dazu passende CSS-Datei muss folgendermaßen aussehen. Ich zeige dir auch hier nur die wichtigen Passagen, die sich auf die zu positionierenden Container beziehen. Die Änderungen habe ich hervorgehoben. Erkennst du das Prinzip?

```
#header {
  height: 80px;
  background-color: #ff9900;
}
```

```
#menu {
  position: absolute;
  top: 80px;
  left: 0;
  width: 160px;
  background-color: #ffff99;
  padding: 20px 5px;
}
```

```
#main {
  background-color: white;
  padding: 20px 10px;
  margin: 0 175px 0 175px;
}
```

```
#column {
  position: absolute;
  top: 80px;
  right: 0;
  width: 160px;
  background-color: silver;
  padding: 20px 5px;
}
```

## CSS-P: Übersicht der Eigenschaften

Fassen wir an dieser Stelle die wichtigsten Eigenschaften und Werte von CSS-P als Überblick zusammen.

### ■ Eigenschaft position

Die wichtigste Eigenschaft ist position. Sie regelt die genaue Positionierung auf der Seite.

Wert	Erläuterung
static	Voreinstellung, norm. Positionierung „im Fluss“
absolute	Positionierung in Bezug auf die linke obere Ecke des <b>übergeordneten</b> Elements (in der Regel ist dies das Element BODY = Browserfenster). Setzt du ein DIV in ein anderes absolut positioniertes DIV, ist jedoch dieses DIV der Ausgangspunkt!
relative	Positionierung relativ zum vorherigen Element (Fehlerhafte Interpretation in NN 4.x!)

### ■ top, left, bottom, right

Mit top bzw. left gibst du den Abstand vom übergeordneten Element (in der Regel oberer bzw. linker Rand) an, am besten in Pixeln. Aktuelle Browser (der Internet Explorer sogar ab Version 5) kennen zusätzlich bottom (Abstand vom unteren Browsererrand) und right (rechte Fensterkante).

## ✓ Übungsteil F: Übungen zum Spaltensatz mit CSS

Du weißt jetzt, wie man:

- attraktive Projekte plant und vorbereitet
- DIV-Container relativ und absolut auf der Seite ausgerichtet
- interessante mehrspaltige Layouts mit Kopf- und Fußzeile erstellt

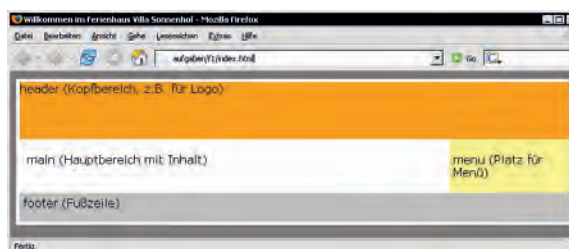


In den folgenden Übungen experimentierst du etwas mit den eben kennengelernten Layouts herum! Die Profi-Übungen sind besonders schwierig und daher freiwillig. (Und im nächsten Kapitel geht es dann mit dem attraktiven Ferienhaus-Projekt weiter.)

### ■ Übung F1: Menü von links nach rechts schieben

Richte dir einen Übungsordner ein und kopiere das Beispiel von Kapitel 6 dort hinein.

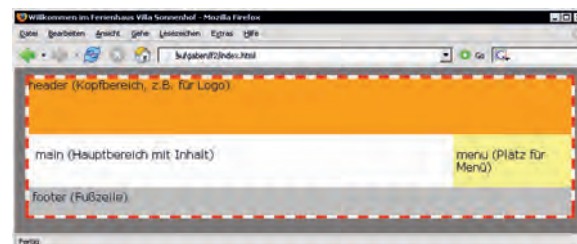
Ändere das Beispiel. Schiebe das Menü von der linken auf die rechte Seite. Tipp: Es genügen zwei kleine Eingriffe in der entsprechenden CSS-Datei.



### ■ Übung F2: Rahmenlinie rot einfärben und mit Strichelleffekt versehen

Färbe die äußere Rahmenlinie rot ein. Wähle eine Dicke von vier Pixeln. Verpasse der Linie außerdem einen attraktiven Strichelleffekt (siehe Abbildung).

Schau dir bei dieser Gelegenheit an, wie Internet Explorer und Firefox diesen Linientyp darstellen. Gibt es Unterschiede?

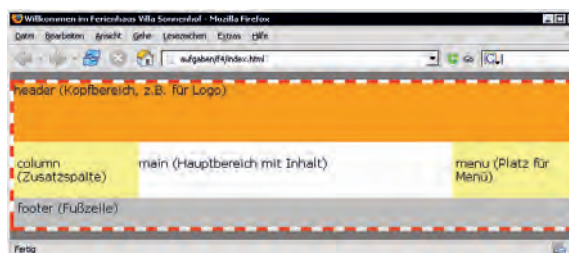


### ■ Übung F3: Relative Breite einstellen

Sorge nun dafür, dass sich die Breite des mother-Containers der Breite des Browserfensters anpasst. Nur die Breite des menu-Containers soll unverändert bleiben.

### ■ Übung F4 (Profi): Dritte Spalte einfügen

Füge außerdem eine dritte Spalte in die Seite ein. Da sich das Menü ja jetzt rechts befindet, soll diese Spalte links angeordnet werden. Nenne den DIV-Container für die Spalte `column`. Verpasse der neuen Spalte die gleiche Breite wie der Menü-Spalte. Nun hast du eine dreispaltige Seite mit relativer Positionierung.



### ■ Übung F5 (Profi): Beispiel mit absoluter Positionierung: Fußzeile ergänzen

Kopiere das zweite Beispiel aus Kapitel 7, also das Beispiel mit Kopfzeile und den drei Spalten, die tlw. durch absolute Positionierung entstanden sind. Ergänze eine Fußzeile, die sich über die gesamte Breite des Browserfensters erstreckt. „Verlängere“ nun die linke oder auch die rechte Spalte – beispielsweise durch Hinzufügen von Inhalt. Mache sie länger als den Hauptbereich. Was stellst du fest? Soviel an dieser Stelle: Gegen diesen Effekt ist leider kein Kraut gewachsen. Aus diesem Grund rate ich persönlich inzwischen auch eher zu Layouts mit relativer Positionierung.

## Kapitel 8: Die Feinheiten – Menü und Inhalt chic gestalten

Zurück zu unserem Ferienhaus-Projekt! Die Grundstruktur steht, wir haben die Seite in die gewünschten Bereiche eingeteilt. Nun werden wir die Inhalte einfügen und in Form bringen. Also das Menü, die Kopfzeile und den Hauptbereich. Fangen wir mit den entsprechenden Stilen an!

### CSS-Datei „layout.css“

Zuerst zeige ich dir die dazu nötige CSS-Datei layout.css. Damit gestalten wir alle Textpassagen und auch das Menü. Zur Erinnerung: Ich binde diese Datei unterhalb der eben besprochenen struktur.css ein! Schreibe diese Datei ab und lege sie ebenfalls in den Ordner css.

```
/* Überschrift im header-DIV */
#header h1 {
    color: white;
    padding: 5px
}

/* Menüleiste (menu-DIV) */
#menu a {
    text-decoration: none;
    font-weight: bold;
    color: black;
    display: block;
    padding: 3px;
}

#menu a:hover {
    color: red;
}

#menu li {
    border: 1px solid #ffff99; /* Trick */
}

#menu #high {
    font-weight: bold;
    color: white;
    background-color: #ff9900;
    display: block;
    padding: 3px;
}

#menu ul {
    line-height: 1.6;
    list-style-type: none;
}
```

```
#menu ul ul {
    line-height: 1.4;
    list-style-image: url(reddot.gif);
    padding-left: 20px;
}
/* Ende Menüleiste */

/* ab hier: main-DIV */
/* Hauptüberschrift: */
#main h1 {
    font-size: 1.3em;
    font-weight: bold;
    color: #333399;
    padding-bottom: 14px;
}

/* Unterüberschrift: */
#main h2 {
    font-size: 1em;
    line-height: 1.4;
    margin-bottom: 6px;
}

/* normaler Fließtext: */
#main p {
    margin-bottom: 12px;
    line-height: 1.3;
}

/* Kastenformat */
#main .kasten {
    text-align: center;
    padding: 5px;
    background-color: #ffff99;
    border: solid 1px black;
    width: 500px;
}

/* Minigrafik rechts ausrichten */
#main .floatR130 {
    float: right;
    margin-left: 5px;
    margin-right: 130px;
}

/* UL mit grafischem Aufz.zeichen: */
#main ul {
    list-style-image: url(reddot.gif);
    margin-left: 30px;
    margin-bottom: 10px;
    line-height: 1.5;
}
```

```
/* STRONG undefinieren */
#main strong {
    font-family: "Century Gothic", ☞
    Verdana, Arial, Helvetica, sans-serif;
    color: red;
}
/* Ende main-DIV */
```

## Die Inhalte formatieren

Und nun schauen wir uns die einzelnen Container und deren Inhalt ganz genau an. Beginnen wir mit der Kopfzeile.

### Die Kopfzeile

Hier wird einzig und allein eine Überschrift 1 platziert. Mehr nicht. (Du kannst hier natürlich auch eine Grafik ablegen!)

```
<div id="header">
    <h1>Ferienhaus &raquo;Sonnenblick☞
    &laquo;</h1>
</div>
```

Dazu passt gleich die erste CSS-Regel auf der Vorseite. Mit der Schreibweise `#header h1` beziehe ich mich dabei nur auf H1-Tags im header-DIV. Diese Vorgehensweise empfehle ich sehr. Ich verwende sie auch für die nächsten Tags!

Eine Regel wird dadurch, dass sie speziell für ein einziges DIV geschaffen wurde, sehr spezifisch. Du kannst damit die Eigenschaften der Elemente dieses DIVs ganz gezielt steuern. Das ist eleganter als wenn du der Überschrift oder den Absätzen einzelne Klassen zuweisen würdest! Der Quellcode bleibt schlanker! Außerdem sinkt die Gefahr, dass sich widersprechende Eigenschaften für das gleiche Element ins Gehege kommen. Denn an einer Stelle benötigst du vielleicht Aufzählungszeichen für eine Liste, an anderer Stelle wiederum nicht.

### ■ Grafik in der Kopfzeile?

Du möchtest eine Grafik in die Kopfzeile setzen, die sich als Hintergrund über die gesamte Breite des header-DIVs erstreckt? Beispielsweise, um einen Farbverlauf zu erzeugen? Gehe in die CSS-Datei `struktur.css` und bearbeite die Regel `#header`. Füge eine Zeile nach folgendem Muster ein:

```
background-image: url(bildname.end);
```

„Unsere“ Grafik heißt `verlauf2.jpg` und liegt im Ordner `css`. Daher sieht diese CSS-Zeile so aus:

```
background-image: url(verlauf2.jpg);
```

Mehr zum Einbinden von Hintergrundgrafiken hatte ich dir ja auf Seite 44 verraten.

### ■ Aufbau der Verlaufs-Grafik

Es handelt sich bei dieser `verlauf2.jpg` um eine 1200 Pixel lange Farbverlaufs-Grafik, die nur 1 Pixel hoch ist. Sie wird nach unten stets wiederholt und so ergibt sich der Verlauffeffekt:



Diese Grafik verläuft von einem hellen in einen dunklen Orangeton.

Vergleiche mit den Dateien aus dem Pfad `kapitel8/ferienhaus_verlauf`. Im Hauptbeispiel jedoch verzichten wir auf einen derartigen Effekt.

### Die Menüleiste im Überblick

Als Nächstes folgt unser kleines Meisterwerk, die Menüleiste. Und so sieht diese im Beispiel als HTML-Quellcode aus:

```
<div id="menu">
    <ul>
        <li><span id="high">Home</span>
            <ul>
                <li><a href="belegung.html">Belegung</a></li>
                <li><a href="preise.html">Preise</a></li>
            </ul>
        </li>
        <li><a href="feedback.html">Feedback</a></li>
        <li><a href="buchung.html">Buchung</a></li>
        <li><a href="impressum.html">Impressum</a></li>
    </ul>
</div>
```

### ■ Richtig verschachteln!

Es handelt sich um eine verschachtelte Liste, eine UL. Jeder Menüpunkt wird innerhalb von `<li>` `</li>` notiert. Außerdem gibt es natürlich Hyperlinks.

Ich empfehle dringend, Menüs als Listen aufzubauen. Das entspricht der logischen Struktur von Menüs – es sind schließlich Listen.

Beachte übrigens die korrekte Verschachtelung. Du findest innerhalb dieser UL eine weitere UL für die Unterebene. Diese untergeordnete UL muss dabei in einen Listenpunkt der übergeordneten Ebene eingebunden werden. Und zwar genau so, wie ich es

oben gezeigt habe. Das schließende `</li>` darf erst dann gesetzt werden, wenn die untergeordnete Liste geschlossen wurde! Falsch wäre folgende Schreibweise:

```
<li><span id="high">Home</span></li>
<ul>
  <li><a href="belegung.html">Belegung</a></li>
  <li><a href="preise.html">Preise</a></li>
</ul>
```

### Listeneigenschaften der Menüleiste

Wirf jetzt einen Blick auf den CSS-Quellcode für das Menü. Ich habe versucht, den Code so schlank wie möglich zu halten. In meiner Lösung stecken viel Erfahrung und viel Probiererei! Und ich habe mir außerdem Mühe gegeben, damit es auch im Internet Explorer 5 ganz leidlich aussieht.

Fangen wir mit der Erklärung an! In der Regel `#menu a` formatieren wir erst einmal die Hyperlinks des Menüs. Wir entfernen u. a. die Unterstreichung und weisen das Schriftformat fett zu.

Die Direktive `display: block;` sorgt dafür, dass die Links als Block dargestellt werden. Sie werden zum Blockelement aufgewertet.

Auf diese Weise zieht sich der anklickbare Bereich über die gesamte Breite der übergeordneten DIVs.

Dieser `a`-Block bekommt außerdem einen Innenrand von 3 Pixeln, damit die Links nicht so gequetscht wirken.

### Der Hover-Effekt

Mit folgender Regel `#menu a:hover { color: red; }` lege ich den Mouseover-Effekt fest: Bei Mausberührung färben sich die Links rot – das ist alles. Wenn du willst, kannst du weitere Eigenschaften notieren, beispielsweise für eine veränderte Hintergrundfarbe und/oder zusätzliche Rahmenlinien.

### Ein kleiner Trick

Nun zur nächsten Regel: `#menu li`. Sie ist nur dazu da, damit der Internet Explorer 6 die Abstände korrekt anzeigt. Sonst würde er viel zu große Abstände unterhalb der Aufzählungspunkte lassen. Warum das so ist, weiß ich nicht!

Hier also der Trick: Ich ziehe um die Listenelemente eine Rahmenlinie. Damit diese aber nicht sichtbar ist, habe ich sie ebenfalls orange eingefärbt.

### Menüpunkt hervorheben

Nun heben wir den aktiven Menüpunkt hervor. Das gelingt mit dem Tagpaar `<span id="high">Home</span>` im HTML-Quellcode. Beachte die `id` namens `high`. Im CSS-Quellcode Sorge ich dann dafür, dass diese `id` durch einen orangen Rahmen hervorgehoben wird.

Schau in die CSS-Datei: Der Pfad `#menu #high` bezieht sich nur auf dieses Tag mit dieser `id`. Auch hier mache ich das `span`-Tag mit `display: block;` erst einmal zum Blockelement. Auch hier vergebe ich einen Innenrand von 3 Pixeln – genau wie bei den Hyperlinks.



### Der Kasten zieht sich über die untergeordnete Liste

Ich habe auch probiert, dieses SPAN-Element wegzulassen und die `id high` direkt im LI-Element zu notieren. Das geht natürlich auch, aber dann sieht das Ergebnis so aus wie in der Abbildung. Der Grund ist einfach: Das erste LI-Element beherbergt auch die untergeordnete UL. Und die wird dadurch in den Effekt mit einbezogen. Damit das nicht geschieht, brauchen wir SPAN als zusätzlichen Container!

### Listenstil und Zeilenhöhe

Fehlen noch die letzten beiden Regeln.

In `#menu ul` setze ich erst einmal den Zeilenabstand auf 1.6. Wenn du magst, kannst du diesen Wert wunschgemäß variieren. Und danach schalte ich die voreingestellten Listenstile ab: `list-style-type: none;` Denn irgendwelche Aufzählungspunkte kann ich zumindest bei den Haupteinträgen keineswegs gebrauchen!

In der untergeordneten Liste `#menu ul ul` wähle ich einen engeren Zeilenabstand von 1.4 und einen roten Punkt als Aufzählungszeichen.

```
list-style-image: url(reddot.gif);
```

Letzteres gelingt natürlich nur, wenn eine Abbildung namens `reddot.gif` im `css`-Ordner liegt.

Gefällt dir das Menü?



## Gestaltung des main-Containers

Werfen wir jetzt einen Blick auf den HTML-Quellcode, den ich im main-DIV eingetragen habe:

```
<div id="main">
<h1>Blick in den Sonnenuntergang</h1>

<p>Bei uns blicken Sie von Ihrer Ferienwohnung direkt aufs Meer und erleben so den
<strong>Sonnenuntergang</strong>.
</p>

<p></p>

<h2>Das bieten wir Ihnen:</h2>
<ul>
<li>modern eingerichtete Zimmer</li>
<li>ruhige Lage, direkt am Wasser</li>
<li>alle Zimmer mit Blick aufs Meer</li>
<li>Einbauküche mit Herd und Geschirrspüler</li>
<li>Festsaal und Gemeinschaftsraum</li>
</ul>

<p class="kasten">Unser Haus ist ein Ort der Begegnung, hier wird gefeiert: Jeden
Samstag halten wir unsere <strong>Tafelrunden</strong> ab.</p>
</div>
```

### ■ Die CSS-Formate

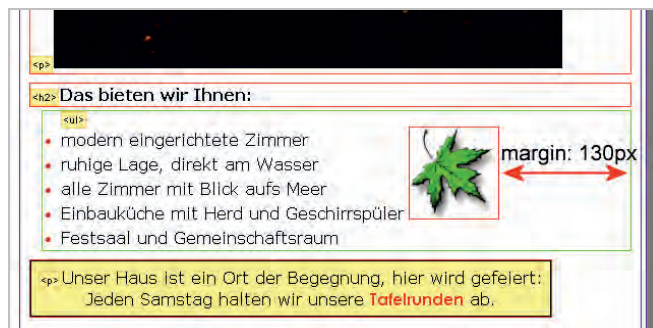
Schau dir den hier gezeigten HTML-Quellcode an. Blättere dabei immer wieder zurück zur CSS-Datei layout.css auf Seite 63. Du merkst: Die Formatierung ist unspektakulär. Mit dem vorhandenen Wissen müsstest du dir alle Stilregeln erschließen können. Auch diesmal beziehe ich mich wieder per Pfad #id element auf das spezielle HTML-Element. Die Regel für die H1 beginnt also so: #main h1, die für die H2 so: #main h2.

Interessant ist vielleicht wieder der Listeneffekt. Auch hier binde ich erneut eine individuelle Grafik als Aufzählungszeichen ein, und zwar die schon für das Menü verwendete Kreisgrafik.

Doch was macht diese Regel?

```
/* Minigrafik rechts ausrichten */
#main .floatR130 {
    float: right;
    margin-left: 5px;
    margin-right: 130px;
}
```

Sie richtet die kleine Grafik rechtsbündig aus. Das Besondere ist der große rechte Außenrand. Der sorgt dafür, dass die Grafik bis in die Aufzählung „hineinreicht“.



**Die Grafik verschafft sich einen rechten Außenrand von 130 Pixeln. Dadurch schwebt sie bis in die Aufzählung hinein.**

Und margin statt padding habe ich nur deshalb gewählt, damit auch der Internet Explorer 5 einen ordentlichen Abstand anzeigt.

#### Wiedererkennbare Klassennamen wählen!

In diesem Fall habe ich einen Klassennamen gewählt, der auch schon etwas über die Eigenschaft aussagt. Mit floatR beziehe ich mich darauf, dass das Objekt nach rechts schwebt. Und die 130 weist auf die 130 Pixel für den rechten Rand hin.

## Kapitel 9: Tabellen de luxe mit HTML und CSS

Tabellen in HTML? Das ist nicht schwer! Ich zeige dir zuerst das komplette Grundgerüst einer Tabelle. Die Tabelle enthält einen Kopf- und einen Fußbereich, zwei Spalten und insgesamt vier Zeilen:

```
<table border="1">
  <thead>
    <tr>
      <th>links oben in der Kopfzeile</th>
      <th>rechts oben in d. Kopfzeile</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>links unten in der Fußzeile</td>
      <td>rechts unten in d. Fußzeile</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>1. normale Zeile, links</td>
      <td>1. normale Zeile, rechts</td>
    </tr>
    <tr>
      <td>2. normale Zeile, links</td>
      <td>2. normale Zeile, rechts</td>
    </tr>
  </tbody>
</table>
```

Uns so sieht diese Tabelle dann aus:

links oben in der Kopfzeile	rechts oben in d. Kopfzeile
1. normale Zeile, links	1. normale Zeile, rechts
2. normale Zeile, links	2. normale Zeile, rechts
links unten in der Fußzeile	rechts unten in d. Fußzeile

Die Rahmenlinie wird nur angezeigt, weil im einleitenden „table“-Tag das Attribut `border` steht. Die Zeile „border=“1“>“ erzeugt eine 1 Pixel breite Rahmenlinie.

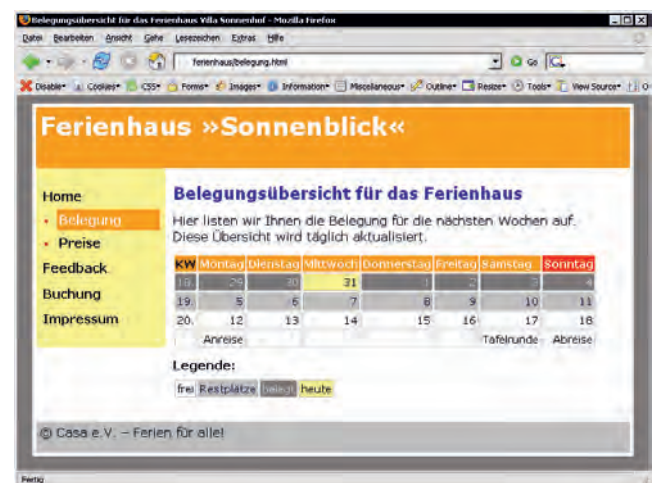
Die gesamte Tabelle steht zwischen den Tagpaaren `<table></table>`. Die Tags `<thead></thead>` und `<tfoot></tfoot>` hüllen den Kopf- bzw. Fußbereich ein. Wichtig: Der Fußbereich wird direkt unterhalb des Kopfbereichs notiert – auch wenn die Inhalte später in die letzte Zeile rutschen. Das Tagpaar `<tbody></tbody>` umschließt dann den Tabellenkörper, also den Hauptinhalt.

Du benötigst keine Kopf- und Fußzeile? Lasse sie weg. Die Tags `<thead></thead>` bzw. `<tbody></tbody>` und alles, was dazwischen steht, sind freiwillig. Wenn du sie weglässt, hat die Tabelle ganz automatisch nur einen „Body“ – selbst dann, wenn du kein `<tbody></tbody>`-Tagpaar schreibst! Denn auch dieses Tagpaar ist optional.

Zeilen wiederum werden durch `<tr></tr>` definiert. Die einzelnen Zellen umschließt du mit `<th></th>` (Zellen der Kopfzeile) bzw. `<td></td>` (alle übrigen Zellen). Du findest dieses Beispiel unter dem Pfad `kapitel9/mustertabelle` in der Datei `tabelle.html`.

### Tabelle für Belegungsübersicht

Zu viel Theorie auf einmal? Fangen wir ganz einfach an! Und zwar mit unserem Beispiel. Schließlich fehlt noch die Belegungsübersicht für unser Ferienhaus. Und so soll die entsprechende Seite namens `belegung.html` aussehen.



Tabellarische Belegungsübersicht mit Kopfzeile, Fußzeile und Angabe der Kalenderwoche.

### Deine Aufgabe: Dokumente vorbereiten

Bereite das Dokument `belegung.html` vor. Tippe in den Hauptcontainer `#main` die Hauptüberschrift „Belegungsübersicht für das Ferienhaus“ und einen ersten Textabsatz: „Hier listen wir Ihnen ...“. Denke auch an das Menü – hier muss nun der Eintrag *Belegung* hervorgehoben werden. Bereite auf diese Weise auch die übrigen noch fehlenden vier Seiten vor, hier genügt jeweils eine Platzhalter-H1 wie „Ausstattung und Preise“ (`preise.html`) oder „Impressum“ (`impressum.html`) usw.

**Schritt für Schritt zur Tabelle ...**

Und nun erstellst du Schritt für Schritt diese attraktive Tabelle. Hier ein paar Tipps zur Vorgehensweise.

1. Tippe das Tag `<table>`. Setze einige Zeilen tiefer gleich das entsprechende Ausschalt-Tag `</table>`.
2. Schreibe nun die Tagpaare für den THEAD, den TFOOT und den TBODY. Damit hast du erst einmal alle wichtigen Haupt-Container „aufgestellt“.

```
<table>
  <thead>
  </thead>
  ... usw ...
</table>
```

3. Definiere die entsprechenden Zeilen – insgesamt fünf. Fange an im THEAD. Dort und TFOOT gibt es nur eine Zeile `<tr></tr>`. (TR steht für *table row*, Tabellenzeile.) Im TBODY dagegen benötigen wir drei Zeilen. Ich zeige dir die Zeilendefinition im THEAD:

```
<thead>
  <tr>
  </tr>
</thead>
```

4. Notiere nun alle Zellen. Eine Zelle der Kopfzeile durch `<th></th>` erzeugt. (TH steht für *table head*, Tabellenkopf.) Die übrigen Zellen werden mit `<td></td>` generiert (*table data*, Tabellendaten).
5. Trage den Inhalt in die Zellen ein.

Ich zeige dir hier die gesamte Tabelle in Komplettersicht und in der von mir bevorzugten, eingerückten Schreibweise. Bei dieser Gelegenheit siehst du gleich vorab, welche CSS-Klassen ich mir zum Gestalten ausgesucht habe. Sie sind fett formatiert.

```
<table class="beleg">
  <thead>
    <tr>
      <th class="kw">KW</th>
      <th>Montag</th>
      <th>Dienstag</th>
      <th>Mittwoch</th>
      <th>Donnerstag</th>
      <th>Freitag</th>
      <th>Samstag</th>
      <th class="so">Sonntag</th>
    </tr>
  </thead>
  <tfoot>
```

```
<tr>
  <td>&nbsp;</td>
  <td>Anreise</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>Tafelrunde</td>
  <td>Abreise</td>
</tr>
</tfoot>
<tbody>
  <tr class="b">
    <td>18.</td>
    <td>29</td>
    <td>30</td>
    <td class="a">31</td>
    <td>1</td>
    <td>2</td>
    <td>3</td>
    <td>4</td>
  </tr>
  <tr class="r">
    <td>19.</td>
    <td>5</td>
    <td>6</td>
    <td>7</td>
    <td>8</td>
    <td>9</td>
    <td>10</td>
    <td>11</td>
  </tr>
  <tr>
    <td>20.</td>
    <td>12</td>
    <td>13</td>
    <td>14</td>
    <td>15</td>
    <td>16</td>
    <td>17</td>
    <td>18</td>
  </tr>
</tbody>
</table>
```

Zum Vergleichen: Den bisherigen Stand findest du unter dem Pfad `kapitel9/ferienhaus_stand1`.

Du lässt Zellen leer wie im Beispiel? Füge stets ein geschütztes Leerzeichen als Platzhalter ein, ein `&nbsp;` – da ältere Browser sonst „hässliche Fehlstellen“ anzeigen. Im Beispiel habe ich es bei leeren Zellen der Fußzeile so gemacht!

## Tabelle mit CSS formatieren

Bisher fehlt der Pepp! Deshalb werfen wir einen Blick auf die Gestaltung! Erstelle eine separate CSS-Datei namens `tabellen.css`. Lege sie ebenfalls im `css`-Ordner ab. Diese Datei bindest du dann nur auf der Seite ein, wo du sie wirklich brauchst: im Beispiel auf der `belegung.css`. Füge dort unterhalb der schon bestehenden zwei CSS-Links einen weiteren Link ein:

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

Und nun lege los! Schreibe in die neue CSS-Datei `tabellen.css` folgende Stilregeln:

```
/* Schriftgröße, 20px Rand unter
Tabelle */
table.beleg {
    font-size: 0.9em;
    margin-bottom: 10px;
}

/* Innenabstand für alle Zellen */
table.beleg td, th {
    padding: 2px;
}

/* alle Zellen außer th */
table.beleg td {
    text-align: right;
    border: 1px solid silver;
}

/* nur th-Zellen */
table.beleg thead {
    color: white;
    background-color: #ff9900;
}

/* Sonntag, Hintergrundfarbe rot */
.so {
    color: white;
    background-color: #ff0000;
}

/* Kalenderwoche: */
.kw {
    color: black;
}

/* Restplätze: */
.r {
    color: #000033;
    background-color: silver;
}
```

```
/* belegt */
.b {
    color: white;
    background-color: gray;
}

/* aktiver Tag, Hintergr. gelb: */
.a {
    background-color: #ffff99;
    color: black;
}
```

### ■ Auch hier: Gestaltung mit Klassen!

Wie du dem Style Sheet entnehmen kannst, habe ich mit Klassen gearbeitet. Die Tabelle selbst gehört zur Klasse `beleg`. Dadurch bin ich nicht auf ein einziges Layout festgelegt! Mit `table.beleg thead` (Oberselektor Unterselektor) erreiche ich, dass nur Zellen innerhalb des Tabellenkopfes gestaltet werden! Vielleicht wunderst du dich, warum ich als Ausrichtung für die Tabellenzellen die Eigenschaft rechtsbündig (`text-align: right;`) gewählt habe? Grund: Wir wollen die Tageszahlen in unserer „Kalendertabelle“ rechtsbündig setzen!

Sieht chic aus, oder? Vergleiche mit dem Beispiel unter `kapitel9/ferienhaus_stand2`.

## Zusatzwissen zu Tabellen

### ■ Breite und Höhe der Zellen

Wie justierst du eigentlich Breite und Höhe der Zellen und damit ggf. auch der Zeile? Ganz einfach! Nutze die CSS-Eigenschaften `width` und `height` und gib den gewünschten Wert an – beispielsweise in px oder em. Damit kannst du das Layout der Tabellen exakt bestimmen. Ansonsten passen sich Höhe und Breite der Zellen dem enthaltenen Inhalt an.

### ■ Vertikalausrichtung justieren

Sobald eine Zelle höher ist als der Zelleninhalt, wird der Zelleninhalt vertikal mittig ausgerichtet. Nutze deshalb ggf. die CSS-Eigenschaft `vertical-align`, um eine obenbündige (oder untenbündige) Ausrichtung zu erzwingen.

Wert	Erläuterung
top	Zelleninhalt wird bündig an der oberen Zellkante ausgerichtet
middle	Voreinstellung, Element „schwebt“ in der Mitte
bottom	Element wird untenbündig ausgerichtet

Damit ein Element genau am oberen Zellrand beginnt, schreibe z.B.: `vertical-align: top;`

## ✓ Übungsteil G: Übungen zu Tabellen und zum CSS-Layout

Du weißt jetzt, wie man:

- einfache Tabellen erstellt
- Tabellen mit CSS attraktiv layoutet
- Höhe, Breite und Vertikalausrichtung für pixelgenaues Layout einsetzt



In den folgenden Übungen arbeitest du weiter am Projekt! Dabei ergänzt du auch eine noch fehlende Seite.

### ■ Übung G1: Tabelle für die Legende ergänzen

Übung macht den Meister! Im Beispiel fehlt noch die Tabelle für die Legende. Erstelle sie. Diese Tabelle benötigt weder THEAD noch TFOOT. Selbst die `<tbody>` `</tbody>`-Tags kannst du weglassen. Achte jedoch auf die Gestaltung mit CSS. Die Tabelle soll zur gleichen Klasse gehören wie die große Tabelle. Auch die Zellen selbst werden mit den entsprechenden Stilklassen gestaltet.

Feedback	KW	Montag	Dienstag	Mittwoch	Donnerstag	Freitag
Buchung	18.	29	30	31	1	2
Impressum	19.	5	6	7	8	9
	20.	12	13	14	15	16
	Anreise					

Legende:

frei	Restplätze	belegt	heute
------	------------	--------	-------

### ■ Übung G2: Höhe der Zeilen bestimmen

Die Tabellen wirken zwar schon sehr attraktiv, doch du brauchst im Beispiel noch etwas mehr Luft! Verändere die Höhe der Zeilen! Nimm im Beispiel eine Höhe von 2.3em für jede Zeile. Erstelle also eine Stilregel, die allen Zeilen diese Höhe zuweist. Was stellst du fest, wenn du dir den Inhalt nun anschaut? Wie ist die voreingestellte vertikale Ausrichtung in Tabellenzellen?

	KW	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
18.	29	30	31	1	2	3	4	5
19.	5	6	7	8	9	10	11	12
20.	12	13	14	15	16	17	18	19
	Anreise					Tafelrunde	Abreise	

### ■ Übung G3: Vertikalausrichtung auf „untenbündig“ stellen

Richte es so ein, dass der Zellinhalt bündig am unteren Zellrand ausgerichtet wird. Vergleiche mit der Abbildung, hier zeige ich dir einen Ausschnitt aus der Tabelle:

19.	5	6	7
20.	12	13	14

### ■ Übung G4: Weitere Zeile einfügen

Füge in die Tabelle eine weitere Zeile mit der 21. Kalenderwoche ein. Sie soll als *belegt* angezeigt werden!

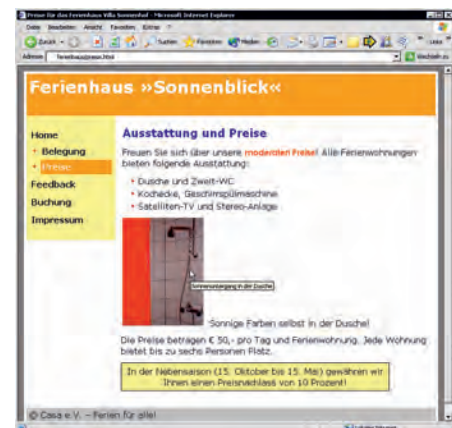
### ■ Übung G5 (Profi): Gestalten durch Hintergrundgrafik mit runden Ecken

Wähle einen abweichenden Hintergrund für die Belegungsseite: Binde meine Hintergrundgrafik `eckenlogo.gif` im header ein. Sie besitzt links und rechts oben abgerundete Ecken. Du findest die Abbildung unter `kapitel9/logografiken`. (Dort gibt es noch andere Logografiken zum Ausprobieren!)

### ■ Übung G6: Ausfüllen der Unterseite „preise.html“

Nimm dir die nächste Seite vor, im Beispiel `preise.html`. Wenn du bisher fleißig mitgemacht hast, hast du zumindest schon eine Platzhalterseite mit dem Menü und der Überschrift. Erstelle den Rest! Orientiere dich dabei an der nebenstehenden Abbildung. Ich hoffe, dass du die Texte lesen kannst. Die Abbildung heißt übrigens `usche.jpg`. Du findest sie bei den Beispieldateien, und zwar unter dem Pfad `kapitel9/projektabbildungen`.

Denke an den alternativen Text für die Grafik! Versuche, die Formatierung mit den bisherigen Stilen hinzubekommen.



## Kapitel 10: Bitte bestellen! Attraktive Formulare dank CSS

Was ist schöner als der neue PC, den du elegant über ein Bestellformular orderst? Oder das seltene Buch, die Reise ... ? Und da sind wir auch schon beim nächsten Beispiel, dem Bestellformular!

**Schickes Formular mit HTML: Ohne CSS sieht es jedoch so lahm aus wie in der rechten Abbildung.**

Im Beispiel probieren wir ein Formular mit allen Schikanen: Mit `<label></label>` für die Beschriftung, mit `<fieldset></fieldset>` und mit `<legend></legend>` zum „Einrahmen“ und Beschriften einzelner Formularblöcke (Persönliche Daten bzw. Ihr Buchungswunsch). Das Attribut `accesskey` gibt es außerdem – zum Festlegen eines Buchstabens für den Tastaturzugriff.

Wir erzeugen auf diese Weise ein tolles, barrierefreies Formular, das nicht mit einer unsichtbaren Platzhaltertable in Form gehalten, sondern mit CSS ausgerichtet wird. Als kleines Highlight gibt es noch einen Hover-Effekt für die Formularfelder.

Mach mit! Nimm dir die `buchung.html` vor und fülle den `#main`-Abschnitt mit „Formular-Leben“. Schreibe einfach den untenstehenden Quellcode ab. Folge mir dann auf den nächsten Seiten durch den „Formulardschungel“.

Und wundere dich nicht, wenn dein Formular am Anfang noch recht kümmerlich aussieht. Die CSS-Formatierung meistern wir ab Seite 75!

```
<form action="http://www.jchanke.de/php/test.php" method="post">

  <fieldset><legend>Persönliche Daten</legend>
    <label for="Name"><u>N</u>ame:</label>
    <input type="text" name="Name" id="Name" accesskey="n">
    <label for="Mail"><u>E</u>-Mail:</label>
    <input type="text" name="Mail" id="Mail" accesskey="e">
    <label for="Telefon"><u>T</u>elefon:</label>
    <input type="text" name="Telefon" id="Telefon" accesskey="t">
  </fieldset>

  <fieldset><legend>Ihr Buchungswunsch</legend>
    <label for="Anreise"><u>A</u>nreisewoche (KW):</label>
    <input type="text" name="Anreise" id="Anreise" size="5" maxlength="2" accesskey="a">
    <label for="Dauer"><u>D</u>auer (in Wochen):</label>
    <input type="text" name="Dauer" id="Dauer" size="5" maxlength="1" accesskey="d">
    <label for="Personen"><u>P</u>ersonen:</label>
    <input type="text" name="Personen" id="Personen" size="5" maxlength="1" accesskey="p">
  </fieldset>

  <label for="Kommentar"><u>A</u>n<u>m</u>erkungen</label>
  <textarea cols="60" rows="5" name="Kommentar" id="Kommentar" accesskey="m"></textarea>
  <p><input type="reset" value="Formularinhalt löschen" class="knopf">
  <input type="submit" value="Jetzt abschicken!" class="knopf" title="Jetzt senden"></p>

</form>
```



### FORM-Tag: Wohin gehen die Daten?

Besprechen wir nun einige wichtige Felder! Fangen wir bei der äußeren Box an: Rund um den Formularbereich wickeln wir die FORM-Tags. Und zwar in folgender Grundsyntax:

```
<form action="Pfad/zu/Auswerteprogramm"
method="post">Formularbereich</form>
```

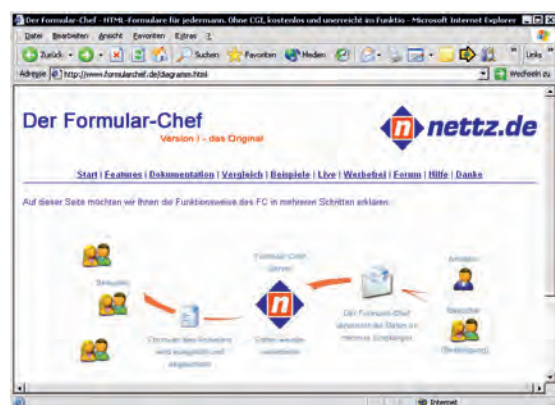
Das Attribut `method` hat mit der Übergabe der Daten zu tun. Schreibe stets `method="post"` (und nicht `method="get"`). Dadurch werden die Daten im Hintergrund an das Auswerteprogramm übermittelt.

#### ■ Das Attribut `action`

Das Attribut `action` kümmert sich um das Weiterleiten der Formulardaten. Für eine wirklich professionelle Formulareauswertung sollten die Daten von einem Programm auf dem Server verarbeitet werden! Trage als Wert dieses Attributs also den Pfad zu diesem Programm ein. Im Beispiel reicht ein einfacher Formmailer in PHP. Der nimmt die Daten entgegen und schickt sie dir dann automatisch als E-Mail zu. Wie das genau geht, führt an dieser Stelle zu weit. Alle Details zur Formularverarbeitung mit PHP erfährst du jedoch in „PHP 5 leicht & verständlich“.

Bietet dein Hoster PHP-Unterstützung? Dann verwende die Datei `unimailer.php`, die ich im Pfad `kapitel10/formmailer` abgelegt habe. Dort findest du auch eine kleine Einbauanleitung.

Kein PHP? Um einen kostenlosen FormMail-Anbieter zu finden, suche auf [Google.de](http://Google.de) nach `formmailer service kostenlos`!



**Formularversendeservice [www.formularchef.de](http://www.formularchef.de) – ein eigener PHP-Formmailer wie der von mir mitgelieferte ist aber besser!**

Im Beispiel habe ich bei `action` erst einmal den Pfad `www.jchanke.de/php/test.php` eingesetzt. Dahinter verbirgt sich mein „Form-Tester“. Dieser

zeigt dir die Daten immerhin schon einmal zu Prüfzwecken an, ohne sie aber zu verschicken.

### Her mit den Daten: Die INPUT-Felder

Die wichtigsten Felder im Formular sind die so genannten INPUT-Felder. Im Beispiel gibt es allein sechs vom Typ „text“. Dort kann der Benutzer etwas „hineintun“, z. B. Name, E-Mail-Adresse oder Telefon. Weiterhin haben wir einmal den Typ „reset“ und einmal den Typ „submit“ – dahinter verbergen sich der Reset- und Submit-Button.

Hier eine Übersicht, welche Werte das `type`-Attribut annehmen kann!

Wert	Erklärung
text	erzeugt ein einfaches Texteingabefeld
password	Feld zur Passworteingabe (zeigt Sternchen an)
radio	anklickbarer Radioknopf (für Einfachauswahl bei mehreren Optionen)
checkbox	abhakbare Checkbox (Mehrfachauswahl)
reset	Reset-Button (Löschen des Formulars)
submit	Submit-Button (Abschicken des Formulars)
button	Schaltfläche (u.a. interessant für JavaScript)
hidden	verstecktes Formularfeld (für CGI, JavaScript usw., solch ein Feld sieht der Nutzer nicht!)

Wir beschränken uns im Beispiel auf das einfache Texteingabefeld, den Reset- und Submit-Button.

Ein End-Tag ist bei INPUT-Felder übrigens nicht nötig!

#### ■ Einfaches Texteingabefeld

Am häufigsten ist mit Sicherheit das einfache Texteingabefeld vom Typ „text“. Schau dir als Beispiel das erste Feld an – das Feld für den Namen. Eigentlich genügt folgende Angabe:

```
<input type="text" name="Name">
```

Was `type="text"` bewirkt, hatten wir eben geklärt – das Einblenden eines Textfelds. Das Attribut `name` sorgt dafür, dass du das Feld später wiederer kennst. Den Wert (hier Name) denkst du dir frei aus, er beschreibt das Feld. Das was der Benutzer direkt ins Feld einträgt, wird als `value` bezeichnet, als Wert.

Jedes Feld gibt ein `name-value`-Paar mit den jeweiligen Daten zurück!

Also: Wenn der Benutzer *Hallmeyer* heißt, schickst dein Formular Folgendes an deine Datenbank bzw. dein Formmailer-Programm: `Name=Hallmeyer`. Und der Formmailer schickt dir diese Infos dann zu.

Im Beispiel sieht das Formularfeld jedoch so aus:

```
<input type="text" name="Name" id="Name"
accesskey="n">
```

### ■ Das LABEL-Tag für die Beschriftung

Hinzugekommen ist zum einen die `id`. Damit gebe ich dem Feld eine eindeutige Kennung. Die ist deshalb nötig, damit unser LABEL-Tag ein Verweisziel bekommt. Denn insgesamt sieht die Zeile so aus:

```
<label for="Name"><u>N</u>ame:</label>
<input type="text" name="Name" id="Name"
accesskey="n">
```

LABEL ist ein Tag für die Beschriftung. Das Attribut `for` bezieht sich dabei auf die `id` des zu „belabelnden“ Feldes. Der Effekt? Wenn der Benutzer auf die Beschriftung klickt, springt der Cursor automatisch in das dazugehörige Feld. Probiere es aus!



Nach Klick auf das „Label“ springt der Cursor direkt in das dazugehörige Texteingabefeld.

### Accesskey festlegen

Und was steckt hinter dem ebenfalls freiwilligen Attribut `accesskey`? Hat das etwas mit den Schlüsselwörtern im Datenbankprogramm Access zu tun?

Mitnichten, damit legst du einen Buchstaben fest, mit dem der Nutzer das entsprechende Formularfeld direkt anspringen kann. In aller Regel muss dieser Buchstabe als Teil einer Tastenkombination getippt werden. Im Internet Explorer und im Firefox unter Windows drückst du zuerst die `Alt`-Taste.

Probiere es aus: Mit `Alt` + `N` springst du ins Name-Feld, mit `Alt` + `P` ins Feld für die Personenzahl und mit `Alt` + `M` in das große Anmerkungsfeld.

Damit der Nutzer diesen „Accesskey“ auch erkennen kann, habe ich den entsprechenden Buchstaben mit `<u></u>` unterstrichen. Ganz genau so wie bei den Menüs in vielen Windows-Programmen!

Die Verwendung des LABEL-Tagpars und des Accesskeys ist keine Pflicht, aber eine wichtige Voraussetzung für barrierefreie Formulare.

### Weitere Attribute für Texteingabefelder

Zurück zum „Input“. Gerade diese Texteingabefelder sind doch wirklich das Größte! Kann man da nicht noch mehr machen? Einen Vorgabewert in das Feld eintragen? Die Länge genau bestimmen? Dafür sorgen, dass der Cursor nach der Eingabe von fünf Zeichen blockiert (ideal für die Eingabe von Postleitzahlen)?

Na klar kann man das! Nutze einfach die entsprechenden zusätzlichen Attribute.

### ■ Big oder small? Attribut size

Das Attribut `size` sorgt für die optische (sichtbare) Länge des Felds. Im Beispiel haben wir das bei den letzten drei Texteingabefeldern gemacht, der Wert für `size` beträgt hier 5. Die tatsächlich mögliche Anzahl an Zeichen ist davon nicht berührt.

### ■ Mehr nicht: Attribut maxlength

Mit `maxlength` bestimmst du die tatsächlich mögliche Anzahl an Zeichen. Schreibe `maxlength="2"` und der Cursor blockiert nach Eingabe von zwei Zeichen. So sorgen wir dafür, dass niemand Kalenderwochen mit drei Stellen eingeben kann. Bei der Personenzahl heißt es gar `maxlength="1"`, denn für mehr als neun Personen sind die Ferienwohnungen schlicht zu klein.

### ■ Gib den value schon vor

Richtig, normalerweise wird der `value` des Formularfelds aus dem Texteingabefeld ausgelesen. Ihn schon vorzugeben macht eigentlich wenig Sinn, oder doch?

Gib den `value` dann an, wenn die Wahrscheinlichkeit der Übereinstimmung groß ist.

Du weißt mit großer Sicherheit, dass die meisten Besucher aus dem PLZ-Bezirk 12345 kommen? Dann gib diesen `value` zur Sicherheit vor. Schreibe `value="12345"`! Dieser Wert erscheint direkt im Formularfeld, kann aber vom Nutzer überschrieben werden! Im Beispiel haben wir allerdings keinen derartigen Fall.

### TEXTAREA: Mehr Feedback!

Diese einfachen Textfelder sind durchaus nett – für kurze Eingaben. Wenn der Besucher dagegen „ganze Romane“ tippen will, wird das schmale Feld kaum reichen.

Deshalb wurde `<textarea></textarea>` erfunden, ein sympathisches Tag-Paar, das einen größeren

Eingabebereich erzeugt. Wie groß, kannst du selbst bestimmen.

Attribut	Erklärung
name	ist eigentlich logisch, jedes Formularelement musst du benennen, z.B. name="Kommentar"
cols	eigentlich „Spalten“, gemeint ist die Breite in Zeichen, z.B. cols="60"
rows	Anzahl der Zeilen, z.B. rows="5"

Auch hier gibt es im Beispiel zusätzlich wieder eine id und das accesskey-Attribut.

### FIELDSET und LEGEND

Du möchtest mehrere inhaltlich zusammengehörige Formularfelder gruppieren? So, wie wir es im Beispiel gemacht haben? Dann hilft das Tagpaar `<fieldset></fieldset>`. Dazwischen befindliche Formularelemente werden automatisch gruppiert – erkennbar durch einen Rahmen. Du wünschst zusätzlich eine Beschriftung für diesen gruppierten Bereich? Notiere diese innerhalb der Tags `<legend></legend>`. Platziere diese LEGEND-Tags unterhalb des einleitenden FIELDSET-Tags.

### Submit- und Reset-Button

Fehlen noch die letzten beiden INPUT-Felder, unsere Buttons! Kaum verpasst du dem type-Attribut den Wert submit, verwandelt sich das Feld in eine attraktive Absende-Schaltfläche.

Der Submit-Button dient zum Senden des Formularinhalts!

Auch hier ist ein value-Attribut zu empfehlen, damit du das Formularfeld individuell benennen kannst: Er „prangt“ direkt auf der Schaltfläche, quasi als Beschriftung! Wenn du die Eingaben im Formular dagegen löschen möchtest, notierst du type="reset" wie „Zurücksetzen“.

Und damit haben wir alle Formularfelder des Formulars abgearbeitet! Es gibt aber noch ein paar weitere Typen, die ich dir hier kurz vorstelle.

### Mann oder Frau? Pull-down-Menü!

Wie wäre es mit einem schicken Pull-down-Menü, auch als Klapplistenfeld bezeichnet? Der Quellcode sieht so aus, im Beispiel heißt das Feld Anrede.

```
<select name="Anrede">
  <option>Herr</option>
  <option>Frau</option>
</select>
```

Du möchtest ein Feld voreinstellen? Tippe im gewünschten OPTION-Tag das Attribut selected.

```
<option selected>Herr</option>
```

Dieses Musterformular findest du unter dem Pfad „kapitel10/musterformular/bestellen.html“.

### Nur einer gewinnt: Radioknöpfe

Als Alternative zu Klapplistenfeldern kannst du auch die Radiobuttons verwenden. Auch hier kann nur eine von mehreren Optionen gewählt werden:

```
<input type="radio" name="Farbe" value="rot">rot
<input type="radio" name="Farbe" value="blau">blau
<input type="radio" name="Farbe" value="lila">lila
```

Wichtig: Alle zusammengehörigen Felder müssen unbedingt den gleichen Namen erhalten, im Beispiel lautet dieser *Farbe*.

### Bitte wählen Sie per Checkbox

Du möchtest den Nutzer wählen lassen? Sie oder er soll etwas „abhaken“? Das gelingt mit einer Checkbox.

```
<input type="checkbox" name="Newsletterabo" value="ja">Newsletter abonnieren?
```

Die Besonderheit: Der Wert gehört diesmal per Attribut value direkt ins INPUT-Tag!

Probiere mein mitgeliefertes Musterformular aus und verschicke die Daten an meinen Formtester.

## Die optische Gestaltung des Formulars mit CSS

Zurück zu unserem Hauptprojekt! Die größte Herausforderung ist nicht das Formular selber, sondern die Ausrichtung per CSS. Soviel vorweg: Mit einer unsichtbaren Platzhaltertabelle wäre es schneller gegangen. Hier zeige ich dir die CSS-Datei für das Formular. Vergleiche mit meiner Lösung aus dem Ordner `kapitel10/ferienhaus_stand2`.

### Der CSS-Code für das Formular

Auch hier fahren wir wieder die gleiche Strategie. Wir lagern den CSS-Quelltext in einer extra Datei aus! Ergänze folgende Zeile im Kopfbereich der `buchung.html` – und zwar unterhalb der anderen beiden CSS-Links:

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

Und so sieht der Quelltext der `formular.css` aus:

```
form {
    padding: 20px;
    margin: 10px 0;
    background-color: #ffffcc;
}

label {
    margin: 5px 0;
    display: block;
}

input, textarea {
    border: 1px solid #ff0000;
    background-color: #ffff99;
}

input:focus, input:hover, textarea:focus,
textarea:hover {
    background-color: #ffff00;
}

input.knopf {
    padding: 3px;
    margin-top: 10px;
    background-color: white;
    border-color: gray;
}

legend {
    color: red;
}
```

```
fieldset {
    display: block;
    width: 60%;
    padding: 10px;
    border: 1px solid silver;
    margin-right: 5px;
    margin-bottom: 10px;
}
```

Vieles kannst du dir sicher inzwischen alleine erschließen. Besprechen wir nun ein paar Details:

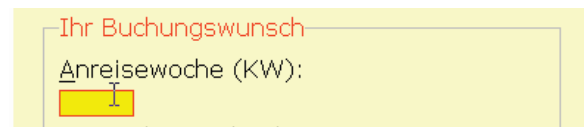
#### Formularfelder untereinander ausrichten

Wie habe ich es geschafft, die Formularfelder untereinander anzuordnen? Indem ich das `LABEL`-Tag mit `display: block;` zum Blockelement aufgewertet habe. Denn Blockelemente verfügen automatisch über einen Zeilenumbruch.

#### Dünne Rahmenlinien für Formularfelder

Die dünnen Rahmenlinien für die Formularfelder erzeuge ich mit der Regel `input, textarea` – als Farbe habe ich rot gewählt. Für die beiden Buttons wünsche ich auch dünne Rahmenlinien, allerdings in grau. Deshalb überschreibe ich diese Eigenschaft weiter unten in der Regel `input.knopf`.

#### Hover- und Focus-Effekt



#### Hover-Effekt beim Darüberfahren und Hineinklicken.

Der schicke Hover-Effekt wird über die Pseudo-Klassen `:hover` (bei Mausover) und `:focus` (bei Hineinklicken in das Feld) erzeugt. Um sowohl die `INPUT`-Felder als auch `TEXTAREA` zu erreichen, schreibe ich folgenden langen Selektor:

```
input:focus, input:hover, textarea:focus,
textarea:hover
```

#### Probieren geht über Studieren!

Den Rest des Formulars habe ich mit der Methode „Probieren geht über Studieren“ ausgerichtet. Denn Internet Explorer und Firefox kommen tlw. zu leicht abweichenden Ergebnissen. So habe ich die Zeile `display: block;` in der `fieldset`-Regel nur mit Rücksicht auf den Internet Explorer eingefügt. Sonst würde der Explorer den mit `margin-bottom: 10px;` eingestellten unteren Rand unterdrücken.

## ✓ Übungsteil H: Übungen zu Formularen und zu CSS

Du weißt jetzt, wie man:

- einfache und „erweiterte“ Texteingabefelder erstellt
- Pull-down-Menüs und Checkboxes erzeugt
- Formulare auswertet und verschickt



In diesen Übungen arbeitest du weiter am Ferienhaus-Projekt!

### ■ Übung H1: Seite feedback.html erstellen

Übung macht den Meister! Weiter geht es in Sachen Formular. Es fehlt noch das Feedback-Formular auf der Seite feedback.html. Erstelle diese Seite, orientiere dich an nebenstehendem Beispiel. Nimm auch hier zum Gestalten die CSS-Datei formular.css.

Neu hinzugekommen ist das Anrede-Feld. Verwende ein Pull-down-Menü. Arbeite auch hier mit dem LABEL-Tag. Ein Accesskey ist nicht nötig, da er in Pull-down-Menüs laut Standard nicht verwendet werden darf. Warum, weiß ich nicht.

Falls du noch keinen Formmailer hast, binde den via <http://www.jchanke.de/php/test.php> erreichbaren Form-Tester ein! Damit kannst du das Formular immerhin überprüfen.

### ■ Übung H2: Länge der Texteingabefelder festlegen

Lege die optische Länge der beiden kleineren Texteingabefelder fest. Sie soll je 25 Zeichen betragen! Sorge dafür, dass eine Maximallänge von 50 Zeichen nicht überschritten werden kann.

### ■ Übung H3: Impressum erzeugen

Erstelle nun die Seite für das Impressum. Baue dort deine Pflichtangaben ein. Mehr zum Thema Impressum kannst du auch hier nachlesen: [www.digi-info.de/de/netlaw/webimpressum/benutzerguide.php](http://www.digi-info.de/de/netlaw/webimpressum/benutzerguide.php).

### ■ Übung H4 (Profi): Auf allen Seiten „nach oben“-Links einbauen

Baue ganz unten auf allen Seiten einen Link ein, der zum Anfang der Seite zurückführt. So, wie ich es dir auf Seite 20f. gezeigt habe. An dieser Stelle jedoch noch ein Trick: Du musst keinen internen Anker erstellen, wenn du schon Elemente mit einer eindeutigen id in deiner Seite hast. Auch ein DIV mit einer id kann als Ziel fungieren, wobei die id praktisch der Name des Ankers ist. Verweise einfach auf das übergeordnete DIV, auf den mother-Container. Alles klar?



### ■ Übung H5 (Profi): Style Sheet für Druckansicht erstellen

Erstelle eine CSS-Datei speziell für den Ausdruck. Dabei sollen der Kopfbereich (header) und das Menü (menu) weggeblendet werden. Vergleiche mit meinem Hinweisen auf Seite 52.

### ■ Übung H6: HTML- und CSS-Quellcode auf Fehler überprüfen

Überprüfe den HTML- und CSS-Quellcode auf Fehler. Wie das geht, verrate ich dir übrigens auf der nächsten Seite! Gräme dich nicht, wenn du lauter CSS-Warnungen wegen irgendwelcher Farben kassierst. Das ist kein Fehler! Das sind lediglich Hinweise, die dir zeigen, wo du ggf. aufpassen solltest. Nicht damit du aus Versehen für Schrift und Hintergrund die gleiche Farbe wählst.



## Nützliche Tools, Links und Sites für HTML und CSS

An dieser Stelle liste ich dir im Schnellverfahren einige nützliche Links und Tools auf.

### HTML und CSS prüfen

#### ■ HTML-Quellcode testen

Du möchtest den HTML-Quellcode auf Fehler testen? Das gelingt über den W3C-Prüfdienst <http://validator.w3.org>. Gib eine Webadresse an, lade die Seite hoch oder kopiere den Quellcode in ein Formularfeld. Dich stört, dass der Dienst auf Englisch „antwortet“? Versuche [www.validome.de](http://www.validome.de), einen sehr strengen deutschsprachigen Prüfdienst.

#### ■ CSS-Code prüfen

Auch für den CSS-Quellcode gibt es einen Prüfdienst. Surfe zu <http://jigsaw.w3.org/css-validator> (ohne www!). Sei aber nicht frustriert wegen der vielen Warnungen. Der Prüfer will partout zu jeder Hintergrundfarbe auch eine Vordergrundfarbe sehen – obwohl das natürlich übertrieben ist.

Zumindest die Vordergrundfarbe wird von übergeordneten an untergeordnete Elemente vererbt – so sparst du Schreibaufwand. Ignoriere diese Warnungen einfach!

### HTML- und CSS-Referenzen

Die wohl beste HTML-Referenz findest du in der SELFHTML unter <http://de.selfhtml.org>. Die SELFHTML ist auch eine prima CSS-Referenz.

#### ■ Spezielle CSS-Referenzen

Wenn du weitere CSS-Referenzen suchst, schau unter [www.css4you.de](http://www.css4you.de) nach. Hier findest du eine ausführliche deutschsprachige CSS-Referenz mit Einsteiger-Workshop. Etwas Ähnliches gibt es auch unter [www.thestyleworks.de](http://www.thestyleworks.de).

#### ■ CSS-Workshops

Interessante Workshops und Lesestoff, u.a. zum Boxmodellfehler des Internet Explorers, findest du auf Tom Stichs Seite [www.stichpunkt.de](http://www.stichpunkt.de). Unbedingt lesenswert ist ebenfalls <http://jendryschik.de>.

### Editoren

Einer meiner Lieblingseditoren ist und bleibt **Weaverslave**, den du unter [www.weaverslave.ws](http://www.weaverslave.ws) bekommst. Viele mögen auch **Topstyle**, von dem zumindest die Lite-Version kostenfrei ist: [www.bradsoft.com/topstyle/tslite](http://www.bradsoft.com/topstyle/tslite). Mir gefällt die

Lite-Version nicht, da sie keine Rückgängig-Funktion besitzt! (Und die brauche ich oft!) Ich bevorzuge den Editor **Qwined** von der finnischen Firma Neoxen. Dieses schmale Freeware-Tool gibt es hier:

[www.neoxen.com/neoxen/products/qwined](http://www.neoxen.com/neoxen/products/qwined)

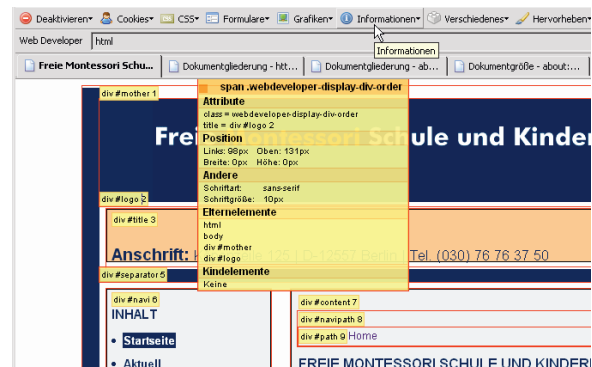
Es klinkt sich in das Kontextmenü des Windows Explorers ein und hebt CSS-Code farbig hervor.

### Browser-Tools

Für den Firefox gibt es viele nützliche Add-ins, u.a. für CSS. Du findest sie häufig sogar in deutschsprachigen Versionen unter [www.erweiterungen.de](http://www.erweiterungen.de).

#### ■ Web Developer

Das wohl beste und bekannteste Add-in ist die Web Developer Toolbar von Chris Pederick. Du findest sie an letzter Stelle in der Rubrik *Werkzeuge für Entwickler*. Installiere diese Erweiterung, starte den Firefox neu und staune: In dieser Symbolleiste findest du Tools zum Einblenden von Stileigenschaften so wohl als auch Befehle zum probeweise Deaktivieren von CSS oder gar zum Testen des Quellcodes.



Die Web Developer Toolbar von Chris Pederec zeigt dir auf Wunsch alle CSS-Eigenschaften ausgewählter Elemente.

Die CSS-Anzeigetools findest du unter dem Button **INFORMATIONEN**. Das Prüfen der Websites gelingt über das Menü **EXTRAS**. Dort findest du aber auch einen Befehl zum Prüfen deiner Links und zum Testen auf Barrierefreiheit.

#### ■ Internet Explorer Developer Toolbar

Für den Internet Explorer gibt es dagegen die Internet Explorer Developer Toolbar, die nicht annähernd den Funktionsumfang bietet wie das Vorbild von Chris Pederick. Du findest diese Toolbar über eine Suche auf [www.microsoft.de](http://www.microsoft.de).



## Stichwortverzeichnis

- @import-Syntax 52
- \_blank 20
- \_parent 20
- \_self 20
- \_top 20
- A 16, 20
  - hover 50
- Absätze 11
- action 72
- ALIGN 12
- ALT 12
- Alternativschriften 25
- Alternativtext 12
- Anker, interne 20
- AOL 18
- Attribute, für HTML 13
- Aufzählung 15
- Ausrichtung 27
- Auswertung des Formulars 72
- background-color 28
- background-image 44
- Blokebene 47
- BODY 6
- border-style 37
- Boxmodell-Fehler des IE 42
- BR 11
- Breite 69
  - in CSS 41
  - nur für eine Seite 41
  - von Grafiken 12
- Cascading Style Sheets 24
- CGI für Formelarauswertung 72
- Checkbox 74
- class 38
- color 28
- CSS
  - Breitenangabe 41
  - die wichtigsten Attribute 33
  - Einführung 24
  - externe CSS-Datei 48
  - Farben 28
  - fett 26
  - Gesetzmäßigkeiten 29
  - Hierarchie 29
  - Innenrand 40
  - Kaskadenprinzip 24, 29
  - Klassen 38
  - Kommentare 32
  - Kompaktschreibweise 32
  - kursiv 26
  - Link zur externen CSS-Datei 49
  - Maßeinheiten 27
  - Rahmeneffekte 37
  - Ränder 39
  - Schriftart 24
  - Schriftgröße 26
  - Style Sheets zentral notieren 30
  - Textausrichtung 27
  - Tricks zu externen Style Sheets 52
  - Unterstreichung 26
  - Zeilenabstand 27
- CSS-Steno
  - für Schriftattribute 34
- Hintergrund 45
- Innenrand 45
- Rahmen 45
- Ränder 45
- DIV
  - mit id 56
- Dokumenttyp-Deklaration 6, 15
- DTD 6, 15
- Editor
  - in Windows 4
  - Tipps und Tricks 48
- Eigenschaften, für CSS 26
- E-Mail-Link
  - mit mailto: 18
  - Nachteile 18
- Entitäten 7
- Erstzeileneinzug 27
- Externe CSS-Datei 48
  - Link setzen mit normaler Syntax 49
  - Link setzen per @import-Syntax 52
- Externe Style Sheets
  - @import-Syntax 52
- Farben
  - Attribut color 28
  - Browser-sichere Palette 28
  - CSS und HTML 28
  - für Hintergrund 28
- Farbverlauf 46
- Feedback
  - durch E-Mail-Link 18
  - durch Formulare 72
- Fett 26
- Firmenhomepage 54
- font 34
- font-family 24
- font-style 26
- font-variant 26
- font-weight 26
- Formmailer 72
- Formularauswertung
  - mit CGI 72
- Formulare
  - abschicken 74
  - Auswertung 72
  - Checkbox 74
  - INPUT-Felder 72
  - Musterformular 71
- FTP-Programm 4
- Füllung 40
- GIF 8
- GMX 18
- Grafiken 8
  - als Aufzählungszeichen 51
  - Alternativtext 12
  - ausrichten 12
  - Breite und Höhe 12
  - für Hintergrund 44
  - Größe in Pixeln 8
  - in HTML einfügen 12
- H 11
- HEAD 6
  - im Detail 11
  - Meta-Tags 11
  - Style Sheets notieren 30
- Headings 11
- Hierarchie 29
- Hintergrundbild 44
- Hintergrundeffekte 44
- Hochladen 4
- Höhe
  - von Grafiken 12
  - von Tabellenzellen 69
- Homepage 13
- hover-Links
  - für dynamische Links 50
  - Probleme 53
  - Pseudo-Klassen 50
- HR 14
- HTML
  - Browservorschau 10
  - Definition 13
  - erstes Beispiel 9
  - Farben 28
  - Kommentare 13
  - Start 5
  - Tabellen 67
- Hyperlinks
  - auf internen Anker 21
  - auf Media-Dateien 21
  - auf PDF-Dateien 21
  - dynamisch mit hover 50
  - E-Mail-Link 18
  - externe 19
  - Grafik als Link 22
  - Grundsyntax 16
  - intern 16
  - interne Anker und Verweise 20
  - mit Beschreibung 20
  - neue Seite aufrufen 20
- ID 56
- IMG 12
- Importieren, externe CSS-
  - Datei 49, 52
- index.html
  - erstellen 5
  - Warum wichtig? 5, 13
- Innenrand 40
- INPUT 72
- JPEG 8
- Kapitälchen 26
- Kaskadenprinzip 24, 29
  - hover-Links 53
- Klassen 38
  - freie 38
  - gebundene 38
  - Pseudo-Klassen für Hyperlinks 50
  - zur Tabellengestaltung 69
- Kommentare
  - in CSS 32
  - in HTML 13
- Kompatibilitätsmodus 6
- Kopf 6
- kursiv 26
- Layout
  - pixelgenau 55
- letter-spacing 33
- LI 15, 33
- Linie 14
- Links *Siehe* Hyperlinks
- list item 15
- Listen
  - attraktiv gestalten 33
  - in HTML 15
  - individuelle Aufzählungszeichen 51
- list-style-image 51
- mailto: 18
- margin 39
- Maskierung von Umlauten 7
- Maßeinheiten in CSS 27
- maxlength 73
- Meta-Tag für Zeichensatz 6
- Meta-Tags
  - im HEAD notieren 11
- Midi 23
- MP3 21, 23
- Multimedia 21
- Musik 21
- Musik auf der Homepage 23
- Navigation
  - durch Hyperlinks 16
- Notepad 4
- Nummerierung 15
- OL 15
- Ordner
  - erstellen 5
  - für das Projekt 5, 13
- P 11
- padding 40
- Paint
  - Aufzählungszeichen 51
  - GIF-Bild erstellen 8
- PDF 21
- PHP 72
- Planen
  - Ordnerstruktur 54
- PNG 8
- Projektordner 5
- Pseudo-Klassen für
  - Hyperlinks 50
- Publizieren 4
- Quirks-Modus 6, 43
- Rahmen 37
- Randeinstellung 39
- Ränder 37
- reset 74
- Rumpf 6
- Schaltflächen, dynamische 50
- Schriftart
  - Alternativen 24
  - lange Schriftartnamen 26
  - mit CSS ändern 24
  - Sammelbezeichnungen 25
- Schriftgewicht 26
- Schriftgröße 26
- Schriftstil 26
- Selbstdarstellung 5
- Selektor 30
  - Gruppierung von Selektoren 31
  - Verschachteln von Selektoren 31
- size 73
- Sonderzeichen 7
- Standard-Modus 43
- Stapelreihenfolge 59, 62
- Stilklassen 38
- Style Sheets
  - extern auslagern 48
  - im HEAD notieren 30
- submit 74
- Tabellen 67
  - Breite und Höhe 69
- Rahmenlinien mit CSS 69
- Tags
  - als Selektor 30
  - als Steuerbefehle 7
  - Definition 13
  - End-Tags 7
  - Übersicht 13
- Tapete für Homepage 44
- TARGET 20
- TEXTAREA 73
- Textausrichtung 27
- Textdatei 5
- Texteingabefelder
  - Attribute 73
  - für Formulare 72
  - großer Bereich mit TEXTAREA 73
- text-indent 33
- text-transform 33
- TITLE 20
- Transparenz 8
- Überschriften 11
- Übungsteil A
  - Erste Schritte mit HTML 14
- Übungsteil B
  - Hyperlinks 22
- Übungsteil C
  - CSS 35
- Übungsteil D
  - Rahmen, Ränder, Effekte 46
- Übungsteil E
  - externe CSS-Datei 53
- Übungsteil F
  - Übungen zum Projekt 62
- Übungsteil G
  - Übungen zu Tabellen 70
- Übungsteil H
  - Übungen zu Formularen 76
- UL 15, 33
- Umlaute 7
- Umrandungsdicke 37
- Umrandungsfarbe 37
- Umrandungsstile 37
- Unterstreichung 26
- URL 19
- value 73
- Vereinshomepage 54
- Vererbung 29
- Videos 21
- Vorwort 4
- W3C 13
- Wave 23
- Weaverslave 11
- white spaces 9
- width 41
- word-spacing 33
- Workshop
  - Aufzählungszeichen mit Paint 51
  - GIF-Bild mit Paint 8
- WS FTP 4
- Zeichenebene 47
- Zeichensatz 6
- Zeilenabstand 27
- Zeilenumbruch 11
- z-index 59, 62



## KnowWare – Bestellservice

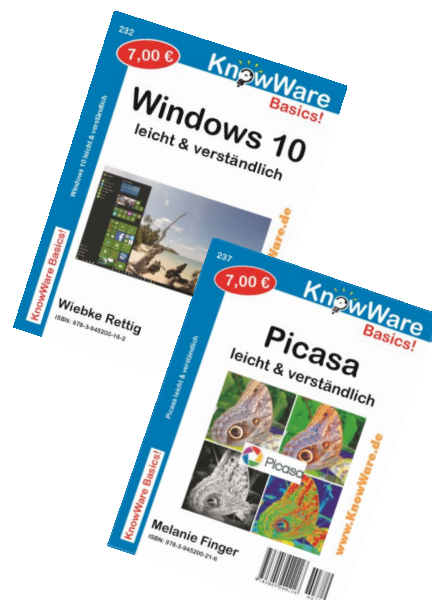
Wissensware aus dem Quadratur Verlag UG

Tel.: +49 (0) 541/33145-20

Fax: +49 (0) 541 / 33145-33

Email: [bestellung@knowware.de](mailto:bestellung@knowware.de)

Internet: [www.knowware.de](http://www.knowware.de)



Nr.	KnowWare Basics	Preis	Menge
173	Access 2003/2002 leicht u. verst.	3,50 €	
198	Access 2007 Workshop	2,50 €	
162	Access 2000 für Einsteiger	2,00 €	
219	Access 2010 leicht & verständlich	6,00 €	
242	<b>Android 5 "Lollipop"</b>	7,00 €	
202	Contao! Webseiten clever gestalten	6,00 €	
179	Excel 2002 leicht und verständlich	3,00 €	
209	Excel 2010 leicht & verständlich	6,20 €	
213	Excel 2010 für Berufsschulen	6,00 €	
236	Excel mit VBA steuern und verwalten	6,00 €	
225	Facebook, Instagram & Google+	6,00 €	
230	Firefox leicht & verständlich	6,00 €	
159	Frontpage 2000 für Einsteiger	2,00 €	
184	Frontpage 2003 leicht & verständlich	3,00 €	
235	Google leicht & verständlich	6,00 €	
215	Schönere Fotos mit <b>GIMP</b>	6,00 €	
214	Homepages für Einsteiger (Plus: HTML Grundwissen)	6,00 €	
231	Homepages mit HTML5 & CSS3	6,00 €	
217	Internet für Einsteiger	6,00 €	
234	IT-Wissen von A-Z	4,00 €	
208	Joomla 1.7	6,00 €	
228	<b>LibreOffice 5.x leicht &amp; verständlich</b>	6,00 €	
238	<b>Office 2016 (2010/2013) leicht &amp; verständlich</b>	6,00 €	
222	Onlineshops eröffnen und führen	6,00 €	
227	OneNote 2010/2013	6,00 €	
165	Outlook 98/2000/2002 für Einst.	2,00 €	
210	Outlook 2010 leicht & verständlich	6,00 €	
237	<b>Picasa leicht &amp; verständlich</b>	7,00 €	
241	<b>PowerPoint 2016 leicht &amp; verständlich</b>	7,00 €	
239	<b>PowerPoint 2016 für Fortgeschr. (2010/2013)</b>	7,00 €	
146	Start mit Access 7/97	1,00 €	
180	Tipps & Tricks zu Windows	2,00 €	
221	Wichtige Windows-Werkzeuge	6,00 €	
200	Windows 7 leicht & verständlich	4,00 €	
232	<b>Windows 10 leicht &amp; verständlich</b>	7,00 €	
148	Windows 95 für Einsteiger	1,00 €	
166	Windows ME für Einsteiger	4,00 €	
192	Windows Vista leicht u. verständlich	2,50 €	

Nr.	KnowWare Basics	Preis	Menge
164	Word 2000 für Einsteiger	2,00 €	
194	Word 2007 im Schnellkurs	3,00 €	
204	Word 2010 leicht und verständlich	6,00 €	
205	Word 2010 für Fortgeschrittene	6,20 €	
211	Word 2010 für Studenten + Schüler	6,00 €	
229	Word 2010/2013 Anwendungstipps&Praxisbeispiele	6,00 €	
206	Word Press	6,00 €	

Nr	KnowWare Extra	Preis	Menge
E21	Delphi leicht und verständlich	5,00 €	
E17	Eltern und ComputerKids	2,90 €	
E05	Windows 2000 für Einsteiger	2,00 €	

Nr	KnowWare Special	Preis	Menge
S08	C++ leicht und verständlich	5,20 €	

Nr	KnowWare Management	Preis	Menge
M12	Dreamweaver 8/CS3 leicht & v.	3,00 €	
M13	Joomla 1.5 leicht und verst.	2,50 €	
M02	MindManager X5 für Einsteiger	2,90 €	

Nr	KnowWare PLUS	Preis	Menge
P14	Dreamweaver 3/4 für Einsteiger	2,90 €	
P20	Excel 2000 für Fortgeschrittene	3,00 €	
P35	Excel 2003 für Fortgeschrittene	3,00 €	
P21	GoLive 5 für Einsteiger	2,90 €	
P33	Office 2003 für Einsteiger	3,00 €	
P34	Outlook 2003 für Einsteiger	3,00 €	
P10	Paint Shop Pro 5/6 für Einsteig	2,00 €	
P31	WebDesign mit Fireworks	2,00 €	
P38	Windows XP I & v	1,50 €	