Inhaltsverzeichnis

CSS für Einsteiger ..... Seite

CSS für Einsteiger

Vorwort von Friedel
Seit vielen Jahren habe ich vor, einen CSS-Kurs zu schreiben. Aber ich habe aus beruflichen und familiären Gründen nicht die Zeit dazu. Um so mehr freue ich mich, dass sejuma so einen Kurs geschrieben hat und ich ihn hier veröffentlichen darf. Der vorliegende Kurs stammt also nicht von mir sondern von sejuma. Vielen Dank dafür.
Friedel

Vorwort von sejuma

Hallo und herzlich willkommen zum CSS-Einsteigerkurs. Als "sejuma" bin ich seit 27.09.2005 Mitglied in "Friedels Board", http://www.friedels-home.com/index.htm?/Board/index.php.
Mein "bürgerlicher" Name ist Friedhelm Senck, Jahrgang 1959, wohnhaft in der Pfalz in Ludwigshafen-Ruchheim. Nach einigen anfänglichen "Wirrungen" beim Homepagebasteln habe ich zu CSS gefunden und diese Methode als sehr hilfreich und effektiv zur Gestaltung von Webseiten erkannt. Wie ich dazu kam diesen CSS-Einsteigerkurs zu schreiben, erfahrt ihr gegen Ende dieses Kurses.
Meine eigene Homepage, die eine Jugend-Handballmannschaft als Thema hat, erreicht ihr unter www.perspektivteam.de. Soviel zu meiner Person.

Für die Gestaltung moderner Websites ist CSS heute nicht mehr wegzudenken.

Viele Anfänger scheuen sich jedoch, diese äußerst nützliche Formatierungssprache zu verwenden, da sie bisher gewohnt waren, in "Tabellenlayout" zu denken.

Wer für seine Homepage von Anfang an CSS verwendet wird sich schnell an die Systematik gewöhnen. Dagegen kann es eher Probleme bereiten, wenn eine HTML-formatierte Seite komplett auf CSS umgestellt werden soll. Das ist gewiss ein großer Aufwand. Wer ihn jedoch investiert wird schnell merken, dass er sich bei der laufenden Homepagepflege oder Layoutänderungen schnell amortisiert.

Der folgende Kurs richtet sich vorwiegend an den bisher unbedarften Einsteiger.
Sein Ziel ist die praktische Entwicklung einer kleinen Homepageseite mit CSS, verbunden mit der Vermittlung wesentlicher Grundkenntnisse, die je nach Bedarf im Einzelfall zu vertiefen sind.

Theorie ist für das Verständnis von Zusammenhängen stets erforderlich und unabdingbar.

Allerdings wurden die theoretischen Erläuterungen nach dem Motto

"So viel wie nötig – so wenig wie möglich"

verfasst.

Im Vordergrund soll vielmehr die praktische Arbeit durch "learning by doing" und die Entwicklung "sichtbarer Ergebnisse" stehen.
Zielsetzung ist es, auf möglichst lockere Weise durch "learning by doing" die wesentlichen Grundkenntnisse für CSS zu vermitteln.

Dankadresse

Ohne Friedel wäre dieser CSS-Kurs an dieser Stelle nicht veröffentlicht worden. Ich bedanke mich deshalb recht herzlich bei ihm, dass er dies ermöglicht hat. Darüber hinaus hat Friedel das Erstkonzept einer kritischen und fachgerechten Überprüfung unterzogen und mit fundierten Verbesserungsvorschlägen zur besseren Verständlichkeit beigetragen.

Ein großes "Dankeschön" richte ich ebenso an "Adlerauge" für ihr akribisches Korrekturlesen und ihre zahlreichen Verbesserungsvorschläge, die sicherlich einem (noch) leichteren Verständnis dienen.

Teil 1

1. Grundsätzliches

1.1 Was ist CSS und warum man es verwenden sollte

CSS ist die Abkürzung für "Cascading-Style-Sheets". Übersetzt man das mit "Kaskadierenden Stil-Blättern" ist man auch noch nicht viel weiter.
Stellt euch vielleicht mal einen "Kaskadenbrunnen vor http://de.wikipedia.org/wiki/Kaskade_(Wasserfall) wo das Wasser Stufe um Stufe von oben nach unten fließt. So ähnlich funktioniert CSS: Elementeigenschaften, die auf einer höheren Ebene definiert werden (z.B. die Hintergrundfarbe im Body) setzen sich in untergeordnete Ebenen (z.B. in den Head- oder Navi-Bereich) fort, was man auch als "Vererbung" bezeichnet. Dennoch kann man die untergeordneten Elemente abweichend von den übergeordneten formatieren.
Einmal abgesehen von diesem nützlichen Effekt lässt sich CSS vereinfacht vielleicht auch so zusammenfassen:
CSS ist eine Formatierungssprache http://de.wikipedia.org/wiki/Cascading_Style_Sheets mit der das Format einer Homepage definiert wird. Während HTML (Textbeschreibungssprache) für den Inhalt einer Seite verantwortlich ist, wird über CSS die Formatierung vorgenommen. Mit CSS wird also z.B. definiert, wie groß bestimmte Objekte sein sollen, an welchen Stellen sie positioniert werden, welchen Abstand sie voneinander haben, welche Farben Schrift und Hintergrund sie haben, wie die Links einer Navigation gestaltet werden – kurz: wie eben das gesamte Layout und die Formatierung aussehen soll.

Das hat mehrere Vorteile:
Dadurch, dass sich HTML auf den Inhalt beschränkt, wird der Quelltext äußerst übersichtlich, denn er enthält eben nur diesen Inhalt. Durch entsprechende Auszeichnungen wird eine Referenz zu CSS hergestellt und die Formatierung übernommen.
Damit kann man auf eine Vielzahl vordefinierter Elemente zurückgreifen und muss im HTML-Teil nicht immer wieder auf's Neue Formatierungsangaben machen.
Das erspart natürlich auch Zeit bei einem Layoutwechsel. Ist man die rosa-rote Hintergrundfarbe seiner Website leid, so ändert man nur an einer Stelle (wir kommen darauf später zurück) den Farbcode und hat sich die Arbeit erspart, Änderungen auf jeder Einzelseite vorzunehmen.

Bis heute werden HTML-Tabellen nicht nur für strukturierte Darstellungen, sondern gerade auch für das Gesamtlayout einer Homepage verwendet. Das macht mitunter den Quelltext nicht nur äußerst umfangreich und unübersichtlich, sondern auch anfälliger für Code-Fehler. Mit CSS werden Tabellen für reine Layoutzwecke (nahezu) überflüssig.

Die Möglichkeiten von CSS sind äußerst vielfältig, aber das wird sich im Laufe des Kurses noch zeigen.
Am Ende des Einsteigerkurses solltet ihr in der Lage sein, eine einfache Homepage mit Header, Navi und Content in Kombination von HTML und CSS zu erstellen.

1.2 Vorbereitende Maßnahmen

Ohne HTML kann man mit CSS nichts anfangen. HTML-Kenntnisse sind für diesen Einsteigerkurs zwar keine zwingende Voraussetzung (ihr bekommt den HTML-Code vorgegeben), sie erleichtern aber in jedem Fall die Arbeit und dienen einem besseren Verständnis. Auf Friedels HTML-Crashkurs http://www.friedels-home.com/index.htm?/Kurse/HTML/01voraussetzungen.htm wird verwiesen.

Erstellen wir uns zunächst mit einem Editor (z.B. Windows Editor oder Phase 5) eine Seite, die dem Grundgerüst einer HTML-Seite entspricht.

Wir ergänzen diesen Code
http://www.friedels-home.com/index.htm?/Kurse/HTML/02struktur.htm

noch um einen Doctype, einen Title und eine Zeichensatzangabe. Innerhalb des Body schreiben wir "Meine erste CSS-Seite". Auf Erklärungen des HTML-Codes soll hier verzichtet werden.

Das Ganze sollte dann so aussehen (das dürft ihr ausnahmsweise kopieren):

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

Diesen Code speichern wir – erstellt euch für diesen Kurs am besten einen eigenen Ordner - unter "index.html" ab. Somit kann diese Datei als spätere Startseite einer Homepage verwendet werden.

In eurem eigenen Interesse:
Eine kleine Bitte hätte ich noch, bevor es richtig los geht: Sofern nicht besonders angegeben gebt im folgenden Kursverlauf bitte den Quelltext per Hand ein – sozusagen ganz herkömmlich manuell. Meist sind es nur kleine Änderungen und Ergänzungen, die ihr eintragen müsst. Gegenüber dem Kopieren hat dies den Vorteil, dass man Fehler machen kann (meist sind das ja nur Kleinigkeiten). Und genau aus diesen Fehlern wollen wir schließlich lernen.

1.3 Erste Schritte:

Genug der Theorie. Jetzt tasten wir uns mal ganz sachte an die Materie heran:

Wenn ihr euch die zuvor erstellte Seite in einem Browser aufruft, sieht das alles noch ziemlich dürftig aus (mal davon abgesehen, dass es auch nur wenig Inhalt gibt).
Bringen wir deshalb etwas Farbe ins Spiel:

1.4 Der CSS-Style-Tag

Da wir ja zwecks Übersichtlichkeit innerhalb des Body möglichst keinerlei Formatangaben mehr haben wollen, fügen wir diese (zunächst) in den Head-Bereich ein. Damit die Browser erkennen, dass es sich um einen CSS-Code handelt, müssen wir ihn entsprechend deklarieren. Üblicherweise beginnt der CSS-Code mit der Einleitung
<style type ="text/css"> und wird mit **</style>** beendet.
Zwischen diese beiden Tags fügen wir dann immer alle CSS-Angaben ein. Also ergänzen wir unsere Datei wie folgt und speichern danach ab:

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

Am Erscheinungsbild hat sich dadurch noch nichts geändert.

1.5 Hintergrundfarbe

Zunächst verschönern wir die Seite mit einer Hintergrundfarbe. Die kann jeder nach eigenem Gusto wählen. Ich habe mich für einen hellgrauen Pastellton mit dem Hexacode #EBECE4 entschieden. Anregungen zur Farbwahl findet ihr z.B. hier http://www.somacon.com/p142.php oder an zahlreichen anderen Stellen im Web.

Unsere gewählte Hintergrundfarbe soll für den gesamten Body-Bereich gelten.
Wir müssen also die Farbe dem "body" zuweisen.
Hierzu rufen wir im CSS-Bereich den "body" auf, benennen die Eigenschaft (Stilangabe) - immer mit einem Doppelpunkt am Ende - und fügen die entsprechenden Werte ein. Die Eigenschaften mit Wertangaben schreiben wir immer in geschweifte Klammern.

Merke:
Alle Eigenschaften mit den dazugehörigen Wertangaben – sowohl im head-Bereich oder später in einer ausgelagerten CSS-Datei – immer innerhalb geschweifter Klammern **{ }** schreiben.

Daraus ergibt sich folgender neuer Queltext:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Meine erste CSS-Seite</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<style type ="text/css">
body
{
background-color: #EBECE4;
}
</style>
</head>
<body>
Meine erste CSS-Seite
</body>
</html>
```

Abspeichern, ansehen und schon kommt Farbe ins Spiel (auch wenn sie bei mir nur grau ist).

Man kann das Ganze auch in einer Reihe schreiben:

```
body {background-color: #EBECE4;}
```

Ich finde jedoch die erste Methode übersichtlicher. Vor allem ist besser zu erkennen, ob die geschweiften Klammern überhaupt vorhanden sind und wo sie geöffnet und geschlossen werden.

Wichtig:
Gewöhnt euch gleich an, nach jeder Angabe ein Semikolon zu setzen. Das ist wichtig um eine Trennung der jeweiligen Angaben und damit eine korrekte Browseranzeige zu gewährleisten.

1.6 Schriften und Überschriften

Der Text sieht noch ziemlich mickrig aus.
Statt der üblichen Times wollen wir besser die schnörkellose Verdana als Schriftart verwenden. Natürlich könnt ihr auch eine andere Schriftart wählen. Die sollte aber gängig sein, so dass sie möglichst jeder Besucher auf seinem Rechner installiert hat. Ansonsten wird sie nicht angezeigt.
Wir ergänzen also unsere Body-Angaben mit der Schriftart, die wir als "font-family" bezeichnen.
Sollte eine gewählte Schriftart auf anderen Rechnern einmal nicht installiert sein, so kann man gleich Ersatzschriften mitliefern oder auf die "Schriftfamilie" verweisen.

Also schreiben wir entweder

font-family: Verdana;

oder

font-family: Verdana, Arial, sans-serif;

Dies bewirkt, dass eine beliebige, serifenlose Schriftart verwendet wird, falls weder Verdana noch Arial installiert sein sollten. Die dann verwendete Schriftart muss jedoch nicht unbedingt ähnlich zu Verdana oder Arial sein.

Die generelle Schriftgröße stellen wir ebenfalls gleich im Body ein. Üblicherweise verwendet man dafür einen Wert von 100%, was der Browsereinstellung des Benutzers entspricht. Hat dieser keine Änderungen an der Standardschriftgröße vorgenommen, entspricht dies 16px. Zu dieser Basis können wir dann später mit einer relativen Größeneinheit die Schriftgröße in unseren einzelnen Seitenelementen festlegen.
Aufgrund eines Opera-Bugs sollte man jedoch besser 100.01% verwenden. http://www.css-technik.de/details/2/5/CSS-Browser-Bugs.htm

Bereits an dieser Stelle sei erwähnt, dass man für manche Browser, insbesondere den Internet-Explorer kleiner 7 manchmal einige Spezialangaben machen muss, damit auch sie den an sich standardkonformen CSS-Code richtig umsetzen. Das ist nicht gerade logisch. Nimmt man darauf jedoch keine Rücksicht, so führt dies zu einer abweichenden Darstellung unserer Homepage in verschiedenen Browsern und das wollen wir möglichst vermeiden.

Wichtig:
Achtet immer darauf,

- **dass zwischen Wertangabe und Einheit keine Leerzeichen sind,**
- **dass für Dezimalwerte anstelle des Kommas ein Punkt verwendet wird.**
- **Bei "Null-Werten" kann auf die Angabe der Einheit verzichtet werden** (es ist schließlich egal, ob ihr "0" Bratwurst oder "0" Schnitzel esst. In beiden Fällen müsst ihr Kohldampf schieben!

Schreibt also nicht 100,01 % sondern 100.01%

Irgendwann werdet ihr euch wundern, wenn eine Wertangabe einmal nicht dargestellt wird. Meist liegt der Fehler dann an einem Leerzeichen zwischen Wert und Einheit oder eben an einem Komma statt Punkt.

Exkurs:
Weil wir gerade beim Body sind, schreiben wir – ebenfalls um wieder ein einheitliches Browserverhalten zu gewährleisten – noch folgende Angaben hinzu (auf die Bedeutung dieser Eigenschaften wird später noch eingegangen):

width: 100%;
height: 100%;
margin: 0;
padding: 0;

Somit ergänzen wir *body* um folgende Angaben:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Meine erste CSS-Seite</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<style type ="text/css">
body
{
background-color: #EBECE4;
font-family: Verdana, Arial, sans-serif;
font-size: 100.01%;
width: 100%;
height: 100%;
margin: 0;
padding: 0
}
</style>
</head>
<body>
Meine erste CSS-Seite
</body>
</html>
```

Sofern ihr die Angaben wie vorgegeben erfasst habt, sollte euch etwas aufgefallen sein:
Am Ende des Wertes für "padding" fehlt nach der "0" das Semikolon. Man kann darauf bei der jeweils letzten Style-Angabe eines Elements verzichten (aber nur beim letzten!).
Da wir nicht ausschließen können, dass wir noch weitere Angaben einfügen, setzen wir zur Sicherheit auch hier noch am Ende besser ein Semikolon.

Merke:
Einige Angaben, die wir dem Body zuweisen, wie hier Schriftart und Schriftgröße gelten grundsätzlich (von einigen Spezialtags wie z.B. Überschriftengröße abgesehen) für alle weiteren Elemente. Man spricht deshalb von "Vererbung". Es ist deshalb nicht erforderlich, diese Angaben - sofern sie identisch sind - in weiteren Elementen zu wiederholen.

Nun wollen wir die Überschrift noch etwas aufpeppen. Dafür gibt es die sechs Hierarchien h1 bis h6.

Verwenden wir h1 und fügen zunächst die Überschrift ins Element ein:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Meine erste CSS-Seite</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<style type ="text/css">
body
{
background-color: #EBECE4;
font-family: Verdana, Arial, MS Sans Serif
}
</style>
</head>
<body>
<h1>Meine erste CSS-Seite</h1>
</body>
</html>
```

Sieht zwar schon etwas besser aus, aber noch nicht zufriedenstellend.
Wir bessern durch CSS etwas nach, indem wir zusätzlich die Überschrift h1 formatieren.
Nehmen wir an, sie soll dunkelblau und zentriert sein.

Für die Schriftfarbe verwenden wir "color" und für die Textausrichtung "text-align".
Unsere Style-Angaben ergänzen wir wie folgt:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Meine erste CSS-Seite</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<style type ="text/css">
body
{
background-color: #EBECE4;
font-family: Verdana, Arial, sans-serif;
font-size: 100.01%;
width: 100%;
height: 100%;
margin: 0;
padding: 0
}

h1
{
color: #000080;
text-align: center;
}

</style>
</head>
<body>
```

```
<h1>Meine erste CSS-Seite</h1>
</body>
</html>
```

Ihr hättet die Überschrift gerne rechts platziert? Dann nehmt statt "center" einfach "right".

Immer noch nicht ganz zufrieden?
Dann unterstreicht doch die Überschrift mit

text-decoration: underline;

Probiert mal einige Sachen selbst aus. Die möglichen Eigenschaften und Werte findet ihr hier: http://www.css4you.de/Texteigenschaften/textproperty.html

Teil 2

2. Zentrale Style-Angaben einbinden

Eine Homepage besteht ja meist aus mehreren Seiten. Kopiert deshalb mal die index.html und benennt die Kopie um in "seite1.html".

Nun verlinken wir beide Seiten, wie das im Rahmen einer Navigation üblich ist.

Auf der Index-Seite fügen wir nach der Überschrift folgenden Link ein:

** CSS Teil 1**

Auf der "seite1.html" ändern wir nun noch Title und Überschrift und verlinken zur index.html. Der Quelltext der "seite1.html" sollte dann so aussehen:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>CSS Teil 1</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<style type ="text/css">
body
{
background-color: #EBECE4;
font-family: Verdana, Arial, sans-serif;
font-size: 100.01%;
width: 100%;
height: 100%;
margin: 0;
padding: 0;
}

h1
{
color: #000080;
text-align: center;
text-decoration: underline;
}

</style>
</head>
<body>

<h1>CSS Teil 1</h1>
<a href="index.html">Startseite</a>
</body>
</html>
```

Beide Seiten sind nun verlinkt.
Zwar sehen die Links noch nicht besonders schön aus, aber das heben wir uns für ein späteres Kapitel auf.

Auf die Gefahr hin, dass ich jetzt nerve: Ich hab's mir mittlerweile anders überlegt.
Der hellgraue Hintergrund gefällt mir nicht mehr und ich möchte ihn durch eine freundlichere Farbe ersetzen (bitte macht es bei euch genauso, denn wie ihr später merken werdet, kommt es mir eigentlich gar nicht auf die Farbe an, ich möchte damit nur etwas anderes demonstrieren).
Wie das geht, solltet ihr ja mittlerweile wissen: Einfach den Farbcode der background-color vom body ändern.

Wir öffnen also die "index.html" mit dem Editor und weisen der background-color einen anderen Farbwert zu, z.B. diesen:

background-color: **#FCF6CF;**

Abspeichern, ansehen und nochmals die Links betätigen.

Hoppla:
Auf der ersten Seite hat die Farbänderung durchgeschlagen, auf der zweiten nicht. Logisch, da haben wir den Farbwert ja auch nicht verändert. Also: Das Ganze noch mal mit "seite1.html" und schon haben wir wieder ein einheitliches Layout.

Jetzt stellt euch mal vor, eure Homepage umfasst zehn oder noch mehr Seiten (was ja nicht ungewöhnlich ist) und ihr wollt überall die Hintergrundfarbe Schriftart, Schriftfarbe oder sonst was ändern. Seite eins aufrufen, ändern, abspeichern. Seite zwei aufrufen, ändern, abspeichern. Seite drei aufrufen, ändern, abspeichern. Und so weiter und so fort. Ganz schön aufwändig und umständlich – gut erkannt! Aber bisher musstet ihr das ohne CSS ja ebenfalls so machen.

Da jedoch der Mensch von Natur aus faul ist, geht's auch einfacher.
Bisher hatten wir die Style-Angaben in den Head-Bereich jeder Seite eingebunden. Deshalb mussten wir sie auch auf jeder Seite ändern.
Um uns künftig diese Mühe zu sparen, verwenden wir nun eine einheitliche CSS-Datei, die für sämtliche Seiten gelten soll.

Öffnet zu diesem Zweck eine neue Datei mit eurem Editor.
Schneidet aus der "index.html" eure bisherigen Style-Angaben aus und fügt sie in die neue Datei ein (bitte **ausschneiden** und nicht kopieren!).
Diese hat dann folgenden Inhalt:

```
body
{
background-color: #FCF6CF;
font-family: Verdana, Arial, sans-serif;
font-size: 100.01%;
width: 100%;
height: 100%;
margin: 0;
padding: 0;
}

h1
{
color: #000080;
text-align: center;
text-decoration: underline;
}
```

Wichtig:
Bitte nicht die style-Tags in die CSS-Datei einfügen, die werden hier nicht benötigt und führen an dieser Stelle lediglich zu Problemen.

Diese Datei speichern wir nun unter "style.css" ab. Ihr könnt auch eine andere Bezeichnung wählen, müsst das dann aber bei der nachfolgenden Referenzierung entsprechend berücksichtigen.

Ebenso die nun wieder abgespeckte "index.html" abspeichern und im Browser wieder ansehen.
Sieht alles wieder so "nüchtern" aus wie zu Beginn des Kurses.
Damit die neu angelegte "style.css" ihre Wirkung entfalten kann, müssen wir nun eine Verbindung zwischen ihr und den html-Dateien herstellen.

Zu diesem Zweck löschen wir noch die stehen gebliebenen style-Tags und ersetzen sie damit:

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

Eure index.html sollte jetzt so aussehen und sollte ebenfalls wieder etwas Farbe haben:

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

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

```
</head>
<body>
<h1>Meine erste CSS-Seite</h1>
<a href="seite1.html"> CSS Teil 1</a>
</body>
</html>
```

Die zweite Seite sieht farblich noch unverändert aus, denn dort haben wir ja bisher noch nichts am Quelltext geändert.
Löscht also ebenfalls die Style-Angaben auf der "seite1.html" und ersetzt sie wie oben durch den Verweis auf die "style.css".

Abspeichern und ansehen. Jetzt sollte auch unsere zweite Seite farblich exakt der ersten entsprechen.

Wollt ihr künftig mal wieder die Hintergrundfarbe (oder was auch immer) ändern, dann nehmt ihr diese Änderung nur **einmal** in der "style.css" vor.

Vorausgesetzt, ihr habt diesen Verweis
<link rel="stylesheet" type="text/css" href="style.css">
im Head-Bereich sämtlicher Seiten stehen, schlagen die Änderungen in der "style.css" nun automatisch auf alle Seiten durch.

Probiert's mal aus, indem ihr nochmal die Hintergrund- oder Schriftfarbe ändert (abspeichern nicht vergessen).
Auf diese Weise habt ihr jetzt das Grundprinzip von CSS mit seinen Vorteilen erkannt:

Man kann damit den HTML-Code erheblich abspecken, wodurch er übersichtlicher wird.
Es findet eine strikte Trennung zwischen Inhalt und Layout statt.
Änderungen von Style-Angaben brauchen nur noch über die "style.css" erfolgen, ohne dass man jede Seite einzeln anpassen muss.

Wir haben jetzt gesehen, dass man Style-Angaben sowohl im Head-Bereich, als auch in einer ausgelagerten CSS-Datei hinterlegen kann, wobei letztere Möglichkeit die weitaus elegantere und vorteilhaftere ist.
Eine zentral hinterlegte CSS-Datei erspart uns die Wiederholung von Formatierungscode auf jeder Seite. Eine Arbeit, die ihr bisher ohne CSS auf jeder Einzelseite durchführen musstet.
Unsere einzelnen Dateigrößen werden kleiner, was schnellere Ladezeiten (die CSS-Angaben werden nur einmal geladen und bleiben im Cache), geringeren Webspace und weniger Traffic zur Folge hat.

Vielleicht habt ihr vereinzelt in manchen Quelltexten auch schon folgendes gesehen:

<body style=" background-color: #FCF6CF; font-family: Verdana, Arial, sans-serif;">

In diesem Fall werden die Style-Angaben direkt beim Aufruf des jeweiligen Elements im HTML-Quelltext definiert.
Auch das ist eine Möglichkeit und kann im Einzelfall durchaus mal angewandt werden. Aber unser Ziel ist ja ein knackig kurzer und übersichtlicher Quelltext. Deshalb werden wir soweit möglich das Instrument einer ausgelagerten CSS-Datei nutzen.

Fassen wir die wesentlichen Vorteile einer ausgelagerten CSS-Datei noch mal zusammen:

- Strikte Trennung zwischen Inhalt und Format
- Übersichtlicher HTML-Quelltext
- Zentrale Verwaltung der Style-Angaben über eine Datei
- Änderungen in der Zentraldatei schlagen auf alle referenzierten HTML-Seiten durch
- Layoutwechsel lassen sich einfacher und schneller vornehmen
- Keine Wiederholung gleichartiger Formatierungen
- Größere Flexibilität bei der Layoutgestaltung

Teil 3

3. Div's &Tricks

Bevor wir mit unserem ersten CSS-Layout loslegen, benötigt es nun doch noch ein wenig Theorie zum Verständnis:

3.1 Div's, Container und Boxen

Im Folgenden wird häufig von Div's, Containern oder Boxen die Rede sein. Im Prinzip handelt es sich dabei immer um das Gleiche.
"div" bedeutet division, oder übersetzt etwa "Bereich". Wenn künftig von einem "div" die Rede ist, bedeutet dies also, dass wir von einem bestimmten, begrenzten Bereich auf der Homepage sprechen. Unter Container oder Boxen versteht man letztlich das Gleiche, sozusagen ein "Behältnis" wo was rein kommt.

Div's sind "**Blockelemente**" und bilden rechteckige Blöcke. Man kann ihnen deshalb bestimmte Breiten und Höhen zuordnen. Sie erzeugen einen eigenen Absatz im Textfluss und können weitere Elemente beinhalten.
Sofern nicht speziell definiert, nehmen Div's die gesamte Breite des ihnen zur Verfügung stehenden Raumes ein. Ihre Höhe richtet sich nach ihrem Inhalt.

Im Gegensatz dazu erzeugen "**Inline-Elemente**" keinen eigenen Absatz im Textfluss. Sie können i.d.R. weitere Inline-Elemente beinhalten, jedoch keine Blockelemente.
Zur Vertiefung und Einteilung der einzelnen Elemente siehe
http://de.selfhtml.org/html/referenz/elemente.htm

Div's werden im HTML-Teil mit ** ** geöffnet und mit ** ** geschlossen. Achtet also darauf, dass ihr jeden geöffneten Div auch wieder schließt. Am besten gewöhnt ihr euch an, bei jedem Öffnen eines Div's diesen auch gleich wieder zu schließen und ihn erst danach mit Inhalt zu füllen. Dabei ist euch z.B. der Editor "Phase 5" behilflich.

Merke:
Auf einer Seite können mehrere unterschiedliche Div's vorkommen (z.B. für Header, Navi und Content).

Da wir unsere Container für verschiedene Zwecke (z.B. Header, Navi, Content) benutzen wollen, ist es sinnvoll, sie mit "Identitäten" zu bezeichnen. Geben wir unserem späteren Header also die Bezeichnung "header".

Im HTML-Teil wird der "header" dann so aufgerufen

** **

und mit ** ** geschlossen.

"id" bedeutet übrigens "identifier", übersetzen wir es einfach mit "Erkennungsmerkmal". Man spricht deshalb auch von "ID-Selektoren".

Merke:
Ein "ID" darf je Seite nur **einmal** verwendet werden.

Tipp:
Da man Div's auch sehr oft ineinander verschachtelt indem man in einen Div mehrere weitere Div's reinpackt, erkennt man durch die ID zwar welchen Div man wo geöffnet hat, aber nicht unbedingt an welcher Stelle er geschlossen wird.
Hier ist die Kommentierungsauszeichnung von HTML ein nützlicher Helfer:

Unseren Header-Div könnten wir damit so schließen:

```
</div> <!--Ende header-->
```

Gewöhnt euch das vielleicht gleich mal an. Es wird vor allem später bei komplexeren Seiten noch hilfreich sein.

Formatiert wird dieser Container in der "style.css" unter der Überschrift "#header"

Merke:
Div-Elemente, die auf jeder Seite nur einmal vorkommen dürfen (ID-Selektoren), werden im CSS-Teil durch Voranstellung einer Raute **#** aufgerufen.
Dagegen werden die "Elementselektoren" wie body, a, h, p, li oder ul **ohne** vorangestellte Raute in der CSS-Datei deklariert.

3.2 Das Boxmodell – unendliche Breiten!

An dieser Stelle ist auch Gelegenheit, das CSS-Boxmodell kurz vorzustellen, welches hier sehr anschaulich dargestellt ist: http://de.selfhtml.org/css/formate/box_modell.htm#w3c
Ich möchte mir an dieser Stelle deshalb nähere Ausführungen sparen; die einzelnen Begriffe werden wir noch im weiteren Verlauf kennen lernen.

Merkt euch für den Moment folgendes:
Die Gesamtbreite eines Elements ergibt sich aus seiner definierten Breite, zuzüglich der Breiten für Innenabstände und Rahmen. Entsprechendes gilt für die Gesamthöhe.

Damit der Internet-Explorer 6 das Boxmodell korrekt darstellt, müsst ihr im Doctype die URL für die "Document-Type-Definition" (DTD) angeben. Die Seiten dieses Kurses haben deshalb folgenden Doctype:

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

Je nachdem, welche HTML- oder XHTML-Code-Variante ihr verwendet, ist jeweils ein entsprechend anderer Doctype einzusetzen.

3.3. Das erste CSS-Layout

So langsam sollte es nun Zeit werden, sich an das erste CSS-Layout zu wagen. Es gibt dazu unzählige Möglichkeiten, um hier nur mal einige zu nennen:

Zweispaltig mit Navi und Content, darüber einen Header
Dreispaltig mit dem Content in der Mitte
Header und Navi oben quer, darunter der Content, evtl. in mehreren Spalten
Layouts mit festen Breiten
Flexible Layouts
usw.

Ich habe mich für den Anfang für ein Layout mit Header, linker Navi-Spalte und daneben angeordnetem Content entschieden. Die Navi soll eine feste Breite haben, der Content soll rechts neben ihr den übrigen Platz einnehmen.
Mit zunehmender Praxis könnt ihr später weitere Layouts selbst kreieren.

3.4 Header

Unsere Homepage soll einen Header erhalten, in den wir z.B. den Homepage-Titel reinschreiben können, und der ggf. auch eine Grafik sowie noch andere Elemente enthalten kann.
Greifen wir auf unsere "index.html" zurück und setzen die Überschrift in einen Header.

Wer noch das Kapitel 3.1 in Erinnerung hat, sollte das bereits selbst können. Aber zur Sicherheit noch mal für alle:

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

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

</head>
<body>
<div id="header">
<h1>Meine erste CSS-Seite</h1>
</div> <!--Ende header-->
<a href="seite1.html">CSS Teil 1</a>
</body>
</html>
```

Falls ihr diese Seite jetzt abspeichert und wieder aufruft, werdet ihr keinerlei Überraschungen erleben. Dazu müsst ihr in der "style.css" erst wieder einige Definitionen hinterlegen.
Unser Header soll einen andersfarbigen Hintergrund und einen Rahmen erhalten.
Wir öffnen also die "style.css" und fügen folgendes ein:

```
#header
{
background-color: #FFCC00;
border: 2px solid #990000;
}
```

Abspeichern und ansehen: Das sollte einer Homepage jetzt schon etwas ähnlicher kommen.

3.5 Rahmen

border: 2px solid #990000; bedeutet:
Es handelt sich um einen Rahmen, der auf jeder Seite gleich ist, mit 2px Rahmenstärke, gerader Linie und der Farbe #990000.

Man hätte das auch etwas ausführlicher schreiben können, nämlich:

border-width: 2px;
border-style: solid;
border-color: #990000;

Aber wer sich gleich die verkürzte Schreibweise angewöhnt, muss nicht soviel schreiben.

Merke:
Bei Verwendung der Kurzschreibweise werden die einzelnen Werte durch ein Leerzeichen getrennt. Das Semikolon ist lediglich nach der jeweils letzten Angabe zu setzen.

Bleiben wir noch etwas beim Rahmen und variieren den:

Oben und rechts soll er eine andere Farbe haben als unten und links.

Wir können jede Seite ansprechen mit border-top, border-right, border-bottom und border-left.

Euer CSS-Header könnte jetzt z.B. so aussehen:

```
#header
{
background-color:#FFCC00;
border-top: 3px solid #990000;
border-right: 3px solid #990000;
border-bottom: 3px double #003300;
border-left: 3px double #003300;
}
```

Probiert das mal selbst mit unterschiedlichen Angaben aus. Was alles möglich ist findet ihr hier: http://www.css4you.de/borderproperty.html

3.6 Hintergrundbilder

Der Header ist euch immer noch zu langweilig? Na gut, garnieren wir ihn mit einer Hintergrundgrafik.

3.6.1 Kachelgrafiken

Zu diesem Zweck hat uns Adlerauge – wie kann's auch anders sein – dankenswerter Weise eine kleine Kachelgrafik zur Verfügung gestellt:
http://ptest.pt.funpic.de/css1/kachel.jpg

Kopiert sie euch in den gleichen Ordner, der eure "index.html" und "style.css" beinhaltet.

Fügt den CSS-Angaben des #header noch folgendes hinzu (selbstverständlich könnt ihr auch eine eigene Grafik verwenden):

background-image: url(kachel.jpg);

Schon habt ihr eine schöne Hintergrundgrafik.

3.6.2 Banner

Ihr wollt ein größeres Banner als Hintergrund einfügen? Auch das ist möglich.
Nehmen wir hierzu folgende transparente Grafik "banner1.gif" mit der Größe 750px auf 80px.
http://ptest.pt.funpic.de/css1/banner1.gif

Oder entwerft euch selbst eine Hintergrundgrafik.

Bindet sie genau wie unter 3.6.1 beschrieben wieder ein.

Ergebnis: Auch diese Grafik wiederholt sich bei größeren Bildschirmen. Das wollen wir jedoch in diesem Fall unterbinden.
Wir ergänzen deshalb unseren #header mit

background-repeat: no-repeat;

und sind schon einen Schritt weiter.

Jetzt soll das Banner noch zentriert werden. Dafür gibt es

background-position: center;

Gehen wir jetzt einmal von der Banner-Variante aus und betrachten unsere CSS-Angaben für den Header.
Da kommt ziemlich oft "background" vor, allerdings mit unterschiedlichen Eigenschaften und Werten.

Nämlich:

```
background-color:#FF9900;
background-image: url(banner1.gif);
background-repeat: no-repeat;
background-position: center;
```

Wir können das alles einfacher darstellen, indem wir es wie folgt zusammenfassen:

background: #FF9900 url(banner1.gif) no-repeat center;

Damit sieht unser CSS-Header jetzt so aus:

```
#header
{
border-top: 3px solid #990000;
border-right: 3px solid #990000;
border-bottom: 3px double #003300;
border-left: 3px double #003300;
background: #FFCC00 url(banner1.gif) no-repeat
center;
}
```

Dass wir trotz Hintergrundgrafik immer noch die Hintergrundfarbe beibehalten, hat folgenden Sinn: Es kann möglich sein, dass die Hintergrundgrafik nicht den gesamten Platz ausfüllt, oder transparent ist. Für diesen Fall bietet sich eine passende Hintergrundfarbe an.
Auch für den Fall, dass bei langsamen Internetzugängen die Hintergrundgrafik langsam geladen wird bietet es sich an, eine Hintergrundfarbe zu hinterlegen, die als "Ersatz" dargestellt wird, bis die Grafik vollständig erscheint.

Seht euch mal hier etwas um, welche weitere Eigenschaften es zum background noch gibt:
http://www.css4you.de/background.html

3.7 Margin

Der Header sieht zwar jetzt schon etwas besser aus, allerdings würde er sicher noch mehr aufgewertet, wenn er nicht so weit an den Rändern "kleben" würde.
Hier hilft uns die Eigenschaft "margin" weiter.

Unter "margin" versteht man den äußeren Abstand zu einem anderen Element.
Eine rechteckige Box kann also einen oberen, rechten, unteren und linken Abstand zu anderen Elementen haben.

Demnach gibt es die vier Einzelwerte

margin-top
margin-right
margin-bottom
margin-left

Unser Header soll links, oben und rechts einen Abstand von je 15 Pixel haben.
Sein Abstand zu den darunter folgenden Elementen soll 30 Pixel betragen.

Dementsprechend müssten wir hinterlegen:

margin-top: 15px;
margin-right: 15px;
margin-bottom: 30px;
margin-left: 15px;

Wichtig (noch mal):
Achtet immer darauf, dass zwischen Wertangabe und Einheit keine Leerzeichen sind.

Also nicht 10 px sondern 10px sonst wird die Angabe nicht interpretiert.
Setzt nach jeder Angabe ein Semikolon.
Kommen wir zurück auf das menschliche Laster der Trägheit:
Sollen alle Seiten den gleichen Abstand zum jeweils nächsten Element haben, so reicht es aus wenn man schreibt
margin: 15px;

Aber auch für unterschiedliche Werte gibt es eine Kurzschreibweise. Statt wie oben jeden Margin-Wert einzeln anzugeben, wählt man folgende Vereinfachung:

margin: 15px 15px 30px 15px;

Die Reihenfolge ist im Uhrzeigersinn, beginnend mit 12 Uhr von oben (top).

Selbst diese Angabe lässt sich noch verkürzen, da in diesem Fall zwei gegenüberliegende Seiten (right und left) den gleichen Wert von 15px haben. Diese beiden identische Werte können zusammengefasst werden, so dass wir folgende Angabe erhalten:

```
margin: 15px 15px 30px;
```

Hier steht der erste Wert für den oberen, der zweite Wert für den rechten und linken sowie der dritte Wert für den unteren Abstand.

Hätten zwei gegenüberliegende Seiten den gleichen Abstand, z.B. oben und unten je 10px sowie rechts und links je 15px, so schreiben wir

margin: 10px 15px;

Prägt euch das etwas ein. Es erspart euch Schreibarbeit und Quellcode und wir werden es für einen anderen Wert später noch mal brauchen.

Somit verschaffen wir unserem Header etwas "Luft", indem wir ihm folgende Margin-Werte in Kurzschreibweise zuweisen:

```
#header
{
border-top: 3px solid #990000;
border-right: 3px solid #990000;
border-bottom: 3px double #003300;
border-left: 3px double #003300;
background: #FFCC00 url(banner1.gif) no-repeat center;
margin: 15px 15px 30px;
}
```

Merke:
Liegen zwei Container nebeneinander, so werden ihre benachbarten Margin-Werte (right und left) addiert.
Liegen sie untereinander gilt für die benachbarten Margin-Werte (bottom und top) folgendes: Bei gleichen benachbarten Margin-Werten wird nur ein Abstand verwendet. Bei unterschiedlichen Werten wird der kleinere vom größeren eliminiert.
Um den Abstand zweier untereinander liegenden Container zu verändern muss man stets den größeren Wert anpassen.

Teil 4:

4.1 Navi-Box

Passen wir zunächst unsere "index.html" an, indem wir den bereits vorhandenen Link in einen Navi-Div einfügen, bzw. um den Link herum die Navi basteln (ihr solltet jetzt schon wissen, wie das geht):

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

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

</head>
<body>

<div id="header">
<h1>Meine erste CSS-Seite</h1>
</div><!--Ende header-->

<div id="navi">
<a href="seite1.html">CSS Teil 1</a>
</div> <!--Ende Navi-->
</body>
</html>
```

Unsere Navi soll eine Breite von 180px erhalten.
Um den "Navi-Standort" besser zu erkennen, verzieren wir sie noch mit einem Rahmen. Ihr wisst ja mittlerweile wie's funktioniert.

Also fügen wir in unsere CSS auch noch die Angaben für die Navi ein und speichern ab:

```
#navi
{
width: 180px;
border: 1px solid #000080;
}
```

Na ja, ist zwar erkennbar, aber sollte doch besser ebenfalls den gleichen linken Abstand haben wie unser Header.

Seht also zu, dass ihr den linken Navi-Rand mit dem Header in Einklang bringt.
Und da ihr gerade dabei seid: Legt bitte auch noch die Schriftgröße für die Navi auf 0,9 em fest.

Noch nie was von "**em**" gehört?
"em" bezeichnet die Größe (Höhe) des Buchstabens "M" der jeweiligen Schriftart. "em" ist im Gegensatz zu "pt" und bedingt auch zu "px" eine relative Größeneinheit.

Im Zusammenhang mit unserer Schriftgrößendefinition im Body (siehe 1.6) von 100% bzw. 100.01% entspricht 1em etwa 16px. Hat der Benutzer dagegen als Schriftgröße 14 px standardmäßig eingestellt, hätte 1 em lediglich die Größe von 14 px.
Durch relative Größeneinheiten wird die Schriftgröße in Bezug zur Basis definiert. So entsprechen 0.8em 80 Prozent von 1 em. 1.3 em ist um 30 % größer als 1 em.
Ein weiterer Vorteil relativer Größeneinheiten ist, dass die Schriftgröße praktisch in allen Browsern (über "Ansicht / Schriftgrad") von den Besuchern verändert (skaliert) und damit an individuelle Lesegewohnheiten (z.B. Schriftvergrößerung bei Sehschwächen) angepasst werden kann.

4.2 Höhenangaben

Wie ihr vielleicht schon bemerkt habt, nehmen div's immer die erforderliche Höhe ihres Inhaltes an.

Schreibt ihr also in die Navi noch einen Text rein oder fügt weitere Links hinzu, so "dehnt" sie sich automatisch nach unten aus.
Man kann den Containern aber auch eine fixe Höhe zuweisen, indem man einen height-Wert angibt.

Soll unsere Navi also 400px hoch werden, so fügen wir in der CSS hinzu:

height: 400px;

Ich fasse die CSS-Angaben für die Navi noch einmal zusammen, damit wir wieder einen einheitlichen Stand haben:

```
#navi
{
width: 180px;
border: 1px solid #000080;
margin-left: 15px;
font-size:0.9em;
height: 400px;
}
```

Ihr wollt der Navi noch eine spezielle Hintergrundfarbe verleihen? Das sollte mittlerweile eine eurer leichtesten Übungen sein.

4.3 Content

Natürlich wollen wir unsere Homepage auch mit Inhalt füllen, wozu wir noch eine Content-Box brauchen.
So langsam bekommen wir Routine:

"index.html" anpassen:

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

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

</head>
<body>

<div id="header">
<h1>Meine erste CSS-Seite</h1>
</div><!--Ende header-->

<div id="navi">
<a href="seite1.html">CSS Teil 1</a>
</div><!--Ende Navi-->

<div id="content">
Ich bin der Content-Container
</div> <!--Ende content-->

</body>
</html>
```

Speichern, ansehen – fertig.

Einwand Euer Ehren?
Der Content soll doch sicher rechts neben die Navi und nicht unter die Navi.

Versuchen wir deshalb einmal, den Content etwas nach rechts zu schieben und bemühen dabei unsere mathematischen Grundkenntnisse:

Navi-Abstand von links: 15px
Navibreite: 180px
Navi-Rahmen: links und rechts je 1px
Schließlich berücksichtigen wir noch einen Abstand zwischen Navi und Content von 18px.
Macht zusammen nach Adam Riese: 15px + 1px + 180px + 1px + 18px = 215px
Vielleicht könnte also ein

margin-left: 215px;

für unsere Content-Anordung hilfreich sein.

Wir ergänzen also unsere CSS-Datei wie folgt (zur besseren Orientierung auch diesmal wieder mit einem Rahmen):

```
#content
{
border: 1px solid #8A2BE2;
margin-left: 215px;
}
```

Abspeichern, ansehen.

Unserem Ziel sind wir immerhin schon einen Schritt näher gekommen. Von der horizontalen Betrachtung her sitzt unser Content bereits richtig.

Wäre aber sicher schöner, wenn er nun auch noch "nach oben rutschen" würde.
Aber nach dem was wir über Blockelemente gelernt haben (siehe 3.1) ist das bei Div's ganz normal, dass sie sich untereinander und nicht nebeneinander anordnen.

"Schönes" CSS! Aber wer wird denn so schnell aufgeben?

4.4 Float

Keine Bange, unser Problem lässt sich mit "float" lösen.
Float verwendet man, wenn ein Element oder ein Bereich von einem anderen "umflossen" werden soll.

Ergänzen wir mal die #navi mit einem

float:left;

und sehen nach, was passiert (natürlich vorher abspeichern).
Hervorragend: Der Content ist nach oben "geflossen" und hat dank unserer Rechenkünste auch einen passablen Abstand zur Navi.

Aber warum dann "fließe links"? Ich finde diese Sprachart auch nicht gerade glücklich.
Die offizielle Definition für "float:left" lautet:
"Das Element steht links und wird rechts davon von nachfolgenden Elementen umflossen."

Wenn es also "steht" kann es doch nicht "fließen". Sei's drum und merkt's euch am besten so:

Merke:
Ein Element mit der Eigenschaft "float: left;" steht links und wird rechts umflossen.
Ein Element mit der Eigenschaft "float: right;" steht rechts und wird links umflossen.

Wer das Thema "float" noch weiter vertiefen möchte (wir sind ja schließlich hier "nur" Einsteiger), dem sei folgender Link empfohlen:
http://www.andreas-kalt.de/webdesign/tutorials/float-theorie

Tipp / Merkspruch:
"Wer floatet muss auch clearen!"

Das bedeutet, dass der "Umfluss" auch wieder beendet werden muss, wenn ein weiterer Container unterhalb des gefloateten platziert werden soll. Dies erfolgt mit den Angaben: "clear: left;" "clear: right;" oder "clear: both;"

Für unser jetziges Layout benötigen das momentan jedoch noch nicht.

Hinweis:
Es gibt per CSS noch einige weitere Möglichkeiten, Navi und Content zu positionieren. Z.B. mit float: right für den Content oder durch absolute Positionsangaben. Je nach Variante müssen dann andere Werte unterschiedlich gehandhabt werden.
Auch hier gilt: "Viele Wege führen nach Rom."

4.4.1 Exkurs: Float und Internet-Explorer

Wer seine Seite bisher im Internet-Explorer 6 entwickelt hat und über einen scharfen Blick verfügt wird folgendes feststellen:
Hatte die Navi bisher einen linken Abstand von 15px, so ist der Abstand jetzt doppelt so groß. Ferner ist der Abstand zwischen Navi und Content auch keine 18px wie vorgesehen. Dabei hatten wir am margin-left-Wert der Navi überhaupt nichts verändert. Er beträgt nach wie vor 15px, genau wie beim Header. Das "Missgeschick" im IE 6 wurde somit durch die float-Eigenschaft verursacht.
Auf der Suche nach einer Erklärung für dieses Phänomen bin ich auf den "IE-Doubled-Float-Margin-Bug", gestoßen, nachfolgend (leider nur in englisch) dargestellt: http://www.positioniseverything.net/explorer/doubled-margin.html
Der Internet-Explorer (<7) macht einem die Entwicklung von CSS-Seiten nicht gerade leicht

Man könnte jetzt speziell für den IE 6 einige "Spezialangaben" machen, die nur dieser Browser interpretieren kann (bezeichnet werden diese Spezialangaben als sogenannte "Hacks" oder "conditional comments").

Konkret müssten wir also für den Internet-Explorer nur den hälftigen Abstand der Navi angeben.
Ändern wir dann aber mal den Margin-Wert des Headers, müssen wir immer auch an die Navi denken – und der Mensch ist ja bekanntermaßen vergesslich.

Fügt also deshalb wie im Link beschrieben noch ein

```
display: inline;
```

beim Navi-Style hinzu und ihr habt dem IE ein Schnäppchen geschlagen.

Tipp:
Verwendet zur Kontrolle der Entwicklung eurer Websites möglichst den Firefox oder IE 7, jedoch möglichst **nicht** den Internet-Explorer 6 oder kleiner. Wenn's im Firefox passt, dann kontrolliert die Seite mit dem IE 6 und seht nach ob und wo es evtl. Abweichungen gibt.
Informiert euch und sucht nach entsprechenden Lösungen für den IE 6, die es meistens auch gibt. Oft ist auch nur ein fehlender oder falscher Doctype verantwortlich.
Mit anderen Browsern habe ich keine Erfahrungen, aber die sollten, vom Netscape einmal abgesehen, eher unproblematisch sein.

4.5 Content-Breite:

Wir wissen aus Kapitel 3.1, dass Container die Breite des ihnen jeweils zur Verfügung stehenden Raumes einnehmen.
Damit können wir uns sparen, unserem #content eine Breite zuzuweisen.
Je nach Bildschirmbreite wird er dann einmal schmäler und einmal breiter werden.
Wollt ihr ihm aber wie bereits bei der Navi eine feste Breite zuweisen, müsst ihr lediglich noch einen width-Wert hinterlegen.

4.6 Content-Justierung

Jetzt klebt der Content noch am rechten Rand – kein Wunder, denn was er an Platz erhaschen kann, das nimmt er sich auch.

Als Abstand wählen wir wieder den gleichen Wert wie bei unserem Header, damit wir "auf Linie" kommen. Das schafft ihr doch selbst! Versucht es gleich in Kurzschreibweise unter Berücksichtigung des bereits vorhandenen linken Abstands.

4.7 Inhalt des Contents

Füllen wir unseren Content mit etwas Inhalt, zum Beispiel mit einem beliebigen Text.

Wir beginnen mit der Überschrift, indem wir diesmal eine Stufe kleiner, also "h2" verwenden.
Den Text zeichnen wir mit den üblichen p-Tags aus und für einen Zeilenwechsel verwenden wir

Ihr könnt jetzt mal euren eigenen Text schreiben, eine Story über eure Schwiegermutter oder vom Pferd verfassen, oder damit's schneller geht mal einfach meine Story übernehmen (Kopieren ist hier mal ausnahmsweise erlaubt – ihr habt das doch bisher hoffentlich strikt unterlassen und manuell gearbeitet?).

Die "index.html" wird zu diesem Zweck wie folgt erweitert:

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

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

</head>
<body>

<div id="header">
<h1>Meine erste CSS-Seite</h1>
</div><!--Ende header-->

<div id="navi">
<a href="seite1.html">CSS Teil 1</a>
</div><!--Ende Navi-->
```

```
<div id="content">

<h2>Wie ich zum CSS-Kurs kam</h2>

<p>
Es begab sich zu Friedels Board. Als Friedel bereits einen
hervorragenden HTML-Crash-Kurs veröffentlicht hatte, machte
sich ein Adlerauge auf, um den Inhalt dieses Kurses in die
Praxis umzusetzen. Das Adlerauge hielt sein Adlerauge aber auf
und achtete insbesondere auf die Rechtschreibfehler, so dass
das Adlerauge vom Friedel zur "Ehrenlektorin" ernannt wurde.
</p>

<p>
Ob dieses Titels fühlte sich das Adlerauge nicht nur sehr
geehrt, sondern leitete daraus ab, weitere Ansprüche stellen
zu dürfen. Also wurde schnell der Ruf nach einem CSS-Kurs
laut, denn CSS ist schließlich "in" und so wollte unser
Adlerauge natürlich immer "up to date" sein.<br>
Es bohrte also immer mal wieder nach, bis unserem Friedel die
glorreiche Idee einfiel, dieses Thema auf seine "to-do-Liste"
zu setzen. Immerhin verschaffte er sich damit zunächst mal
Ruhe.
</p>

<p>
Zwischenzeitlich war Adlerauge - von kleineren Jokes mal
abgesehen - immer schön brav und hat sich sogar aufgeopfert,
einen Pi-Kurs (Photo-Impact-Kurs) ins Board zu stellen.<br>
Die Resonanz darauf war so euphorisch, dass Adlerauge sehr
schnell neben Friedel zum zweiten Forenheld (besser: Heldin)
mutierte.
</p>

<p>
Aus so einer Position heraus konnte Adlerauge natürlich noch
größere Töne spucken und so versuchte sie es bei einem anderen
"Opfer".<br>
Nun, sie war auch schon des öftern unserem "sejuma"
behilflich, hatte ihm dankenswerter Weise mal ein paar Smily-
Gifs gebastelt, und so stand irgendwann auch sejuma in ihrer
Schuld. So nutze sie eine Forenabwesenheit unseres Friedel zu
"konspirativen" Postings und e-mails mit sejuma, um diesen
schließlich zu überreden einen kleinen CSS-Grundkurs
einzustellen.
</p>
<p>
Wie Männer nun mal sind, sind sie es auch gewohnt, rechtzeitig
und vorausschauend ihren Widerstand aufzugeben, um schlimmere
Folgen einer Auseinandersetzung zu vermeiden.<br>
```

```
So ließ sich sejuma überreden, haute kräftig in die Tasten und
stellte schließlich den lang ersehnten CSS-Grundkurs in dieses
Forum.
</p>
<p>
Bleibt zu hoffen, dass er Anklang findet und damit für eine
gewisse Zeit wieder "CSS-Ruhe" im Board einkehrt.
</p>
Es grüßt euch:<br>
sejuma
</div><!--Ende content-->

</body>
</html>
```

4.8 Padding

Ob euch dieser Text gefällt oder auch nicht möchte ich jetzt mal dahingestellt sein lassen. Ihr durftet ja auch einen eigenen Text über's Pferd oder die Schwiegermutter schreiben.

Wie ihr sicherlich schon bemerkt habt, steht er überall zu dicht am Rand.
Um das zu verbessern verwenden wir die Eigenschaft "padding".

Beim "padding" handelt es sich um den Leerraum oder Innenabstand zwischen dem Inhalt und seinem Element.

Fügt also dem #content noch ein

padding:15px;

hinzu und es sieht optisch schon viel besser aus.
Ihr könnt für alle Seiten den gleichen Abstand oder auch unterschiedliche Abstände wählen.
Die Schreibweise funktioniert genauso wie unter "Margin" (3.7) beschrieben.

4.9 Style-Kosmetik

Ihr habt noch weitere Ansprüche an eure erste CSS-Seite?

Dann machen wir noch etwas Feinkosmetik.

Wir wollen den Text im Content in Blocksatz schreiben, also links- und rechtsbündig.

Das geht mit

text-align: justify;

Die Schrift ist euch zu groß oder zu klein? Dann stellen wir sie eben ein.
Wir können die Schriftgröße speziell dem #content zuweisen indem wir einfügen:

font-size: 1.1em; oder
font-size: 0.9em;

Probiert mal beides aus.

Ihr habt euch für 0.9em entschieden aber die Zeilen sind jetzt zu dicht beieinander.
Dem kann mit

line-height: 1.5em;

oder einem anderen Wert eurer Wahl abgeholfen werden.

Probiert einfach mal verschiedene Werte in Abhängigkeit von der Schriftgröße aus.

Einzelne Worte sollen rot und fett hervorgehoben werden? Auch das schafft CSS.
In diesem Fall verwenden wir jedoch eine "Klasse" und keinen Div. Ihr erinnert euch: Ein "Div-Identifyer" darf nur einmal auf jeder Seite vorkommen. "Klassen" kann man dagegen mehrfach verwenden.
In CSS definieren wir Klassen, indem wir ihnen einen Punkt voranstellen.

Fügt in eure CSS also noch folgendes ein:

```
.rotfett
{
color: #f00;
font-weight: bold;
}
```

Sucht euch jetzt in der "index.html" zwei oder mehrere Textstellen aus, die rot und fett markiert werden sollen.

Ich mach' das gleich mal in der ersten Zeile.
Die Markierung innerhalb eines Textes wird mit **** eingeleitet und mit **** beendet.

Das sieht dann so aus:

Es begab sich zu `<span class="rotfett">`Friedels Board`</span>`. Als Friedel bereits einen hervorragenden `<span class="rotfett">`HTML-Crash-Kurs`</span>`

Versucht mal, auf diese Weise einzelne Wörter wie mit einem Textmarker gelb hervorzuheben, indem ihr eine zusätzliche CSS-Klasse bildet und im HTML-Teil einzelne Wörter damit kennzeichnet (Stichwort: "Hintergrundfarbe", der Farbcode für gelb ist #FFFF00;)

Natürlich könnt ihr auch ganzen Absätzen oder Div's "Klassendefinitionen" zuordnen. Statt `<span class="rotfett">` verwendet ihr `<p class="rotfett">` oder `<div class="rotfett">` und schon habt ihr einen ganzen Absatz oder Div mit roter Fettschrift. Den Tag dann aber bitte mit `</p>` bzw. `</div>` schließen.

4.10 CSS-Angaben kommentieren

Zum Schluss dieses Kapitels noch ein Tipp: Man kann – wie bei HTML – auch die CSS-Angaben kommentieren.
Wollen wir uns später noch erinnern was eine einzelne Angabe bedeutet, können wir einen Kommentar anfügen. Damit dieser mit dem übrigen Code nicht in Konflikt gerät, müssen wir ihn entsprechend deklarieren.

Ein CSS-Kommentar beginnt mit einem Schrägstrich und einem Sternchen /*. Beendet wird er mit Sternchen und Schrägstrich */. Ich habe mir angewöhnt, vor und nach dem Text noch zwei Minus-Zeichen (ähnlich wie bei den HTML-Kommentaren) einzufügen, damit der Kommentar noch besser hervorgehoben wird.

In der CSS-Datei könnte dies dann z.B. so aussehen:

```
/*--mit der folgenden Klasse "rotfett" können einzelne Wörter
oder Textstellen rot und fett dargestellt werden--*/
.rotfett
{
color: #f00; /*--Schriftfarbe: rot--*/
font-weight: bold; /*--Fettschrift--*/
}
```

4.11 Bilder einfügen

Bilder und Grafiken lockern die Homepage auf und "sagen mehr als tausend Worte". Deshalb wollen den vielen Text im Content durch ein Bild etwas ansprechender gestalten. Verwenden wir dazu unser passendes "Gruppenbild" vom Forentreff.
http://www.friedels-home.com/Kurse/CSSsejuma/Teil01/027grafiken/forentrio.jpg
Kopiert euch dieses Bild und speichert es unter "forentrio.jpg" im gleichen Ordner ab in dem sich eure "index.html" befindet. Natürlich könnt ihr auch passend zu eurem Text ein anderes Bild verwenden.

Wie man Grafiken und Bilder in den HTML-Quelltext einfügt erfahrt ihr hier.
http://www.friedels-home.com/index.htm?/Kurse/HTML/19einbinden.htm

Unser Bild soll an dieser Stelle zwischen `<p>` und

```
Aus so einer Position heraus
```

eingefügt werden mit

```
<img src="forentrio.jpg" width="409" height="246"
border="0" alt="Forentrio">
```

An dieser Stelle haben wir dann folgenden Quelltext stehen:

```
<p>
<img src="forentrio.jpg" width="409" height="246"
border="0" alt="Forentrio">
Aus so einer Position heraus
```

Das sieht nun zunächst recht unschön aus, denn neben dem Bild ist bis auf die letzte Zeile alles frei. Die Ursache ist, dass "img" ein Inline-Element ist und somit wie ein einzelnes Wort selbst im Textfluss steht.
Sinnvoll wäre es, wenn das Bild vom Text umflossen würde.
Versuchen wir es mit einem **float: left**; falls der Text rechts vom Bild stehen (bzw. fließen) soll.
Die float-Eigenschaft verwandelt Elemente automatisch zu Blockelementen.

Es gibt nun mehrere Möglichkeiten, wie man das per CSS regeln kann. Wir könnten z.B. speziell für das Bild eine eigene Klasse mit den erforderlichen Angaben hinterlegen. Aus der Vorüberlegung, dass das Bild später auch noch einen Untertitel erhalten soll und mehrere Bilder unterschiedliche Größen haben können, halte ich es in diesem Fall (ausnahmsweise) für sinnvoll, die Style-Angaben einmal nicht in der CSS-Datei zu hinterlegen, sondern direkt im Quelltext anzugeben. Ich hatte auf diese Möglichkeit bei den Styleangaben (ganz unten) hingewiesen. Ihr könnt dabei auch gleich sehen wie es möglich ist, Formatierungs-Angaben einmal außerhalb der CSS-Datei vorzunehmen. Allerdings soll das nicht zur Regel werden, sondern nur dann Anwendung finden, wenn mal eine einmalige Spezialangabe erforderlich wird.

Somit fügen wir den img-Tag in einen Div mit folgender Angabe:

```
<div style="float: left;">
<img src="forentrio.jpg" width="409" height="246"
border="0" alt="Forentrio">
</div>
```

Noch unschön ist, dass der Text zu sehr am Bild klebt. Wir können dies lösen, indem wir noch einen margin-right-Wert von 15px zuweisen.
Außerdem wollen wir gleich berücksichtigen, dass das Bild einen Untertext in Fettschrift haben soll.
Als Schriftgröße verwenden wir 0.8em.
Die "line-height" soll 1.2em betragen.
Damit dieser Text nicht über die Bildgröße herausragt müssen wir den Div auf die Bildbreite, in diesem Fall 409px begrenzen.
Schließlich gilt es zu beachten, dass unser Content-Text im Blocksatz formatiert ist. Für den Bilduntertext mag dies nicht gut aussehen, weshalb wir hier eine linksbündige Textausrichtung vorsehen wollen.
All diese Angaben weisen wir unserem Div wie folgt zu:

```
<div style="float: left; margin-right: 15px; font-
weight: bold; font-size: 0.8em; line-height: 1.2em;
width: 409px; text-align: left;">
<img src="forentrio.jpg" width="409" height="246"
border="0" alt="Forentrio">
</div>
```

Fehlt jetzt nur noch ein passender Bild-Untertext. Hierzu schlage ich folgendes vor:

```
<div style="float: left; margin-right: 15px; font-
weight: bold; font-size: 0.8em; line-height: 1.2em;
width: 409px; text-align: left;">
<img src="forentrio.jpg" width="409" height="246"
border="0" alt="Forentrio">
<br> Haben beim "Pfälzer Schoppen" den CSS-Kurs
ausgeheckt: Adlerauge, sejuma und Friedel beim ersten
Forentreff
</div>
```

Dies war nur eine Möglichkeit, wie man mittels CSS-Angaben Bilder "in Position" bringen kann. Das gleiche Ergebnis hätte man auch mit einer Tabelle oder einer Liste erzielen können.
Für unseren CSS-Einsteieger-Kurs wollen wir das Thema "Bilder und Grafiken" jedoch damit bewenden lassen.

Kommen wir nun zu Teil 5 mit der "Menue-Formatierung".

Teil 5

5. Formatierung eines Menues

Unsere erste CSS-Seite ist nun fast schon perfekt.
Bleibt eigentlich nur noch, den anfangs eingefügten Link noch etwas aufzupeppen.
Weil ihr aber mittlerweile schon richtige CSS-Experten seid wollen wir es dabei nicht belassen, sondern wagen uns mal gleich an ein richtiges Menue mit mehreren Links heran.

5.1 Listenformat

Wir bedienen uns dabei des HTML-Instrumentes der "unordered list", kurz "ul" genannt. Das ist keine unordentliche, sondern lediglich eine "unsortierte" Liste. Soll eigentlich nur heißen ohne Durchnummerierung, also eine reine Aufzählungsliste.
Näheres hier: http://de.selfhtml.org/html/text/listen.htm

In unsere Navi-Box (HTML-Teil) fügen wir mal zunächst die Grundstruktur einer solchen Liste ein und versehen jeden Listeneintrag mit einem Link.
Den ersten Listeneintrag verwenden wir für die Überschrift und zeichnen diese zusätzlich mit h3 aus. Danach kommt unser erster Link zur Startseite. Für den zweiten nehmen wir den vorhandenen zu unserer Seite eins. Bei den anderen verfahren wir entsprechend. Ihr könntet zum Beispiel für jeden Kursteil einen Link setzen und euch dann auf den entsprechenden Seiten aufschreiben, was euch wichtig war.
Unser Navi-Code der index.html sieht dann so aus:

```
<div id="navi">

<ul>
<li><h3>Menue</h3></li>
<li><a href="index.html">Startseite</a></li>
<li> <a href="seite1.html">CSS Teil 1</a></li>
<li> <a href="seite2.html">CSS Teil 2</a></li>
<li> <a href="seite3.html">CSS Teil 3</a></li>
<li> <a href="seite4.html">CSS Teil 4</a></li>
<li> <a href="seite5.html">CSS Teil 5</a></li>
<li> <a href="seite6.html">CSS Teil 6</a></li>
</ul>

</div><!--Ende navi-->
```

Fügt bei Bedarf noch weitere Links hinzu (aber für diesen Kurs reicht's mal aus).

Tipp:
Macht euch an dieser Stelle bereits unbedingt Gedanken, wie viele Links ihr für welche Seiten insgesamt braucht. Wie lauten die Dateibezeichnungen für eure Einzelseiten?
Fügt in der Navi möglichst bereits jetzt die Links zu euren Einzelseiten ein.
Ihr könnt dann diesen Code-Teil später einfach kopieren, bei den anderen Seiten einfügen und habt auf diese Weise überall das gleiche Menue.
Wird das Menue nachträglich durch weitere oder wegfallende Links geändert, so müsst ihr diese Änderungen auf allen Seiten vornehmen. Schließlich ist nur unser CSS-Code zentral hinterlegt, nicht aber die Menue-Links.

Wir werden im Fortgeschrittenen Teil (sofern es einen jemals geben sollte) noch eine Lösung kennen lernen, wie man auch diesen Aufwand mit einem kleinen PHP-Script vereinfachen kann.

Speichert nun die "index.html" wieder ab und seht euch die Seite an.
Hat schon eine große Ähnlichkeit mit einem Menue, aber genügt sicher noch nicht unseren Ansprüchen. Deshalb müssen wir erneut unsere "style.css" bemühen und ergänzen.

Entfernen wir einmal zunächst die Aufzählungspunkte und sorgen dafür, dass die Liste keinerlei äußere und innere Abstände zur Navi-Box hat, indem wir unserer unordered list "ul" innerhalb des Navi-ID's folgende Eigenschaften zuweisen:

```
#navi ul
{
margin: 0;
padding: 0;
list-style-type: none;
}
```

Statt "**#navi ul"** würde auch ein einfaches "**ul"** ohne vorangestellte Raute reichen, da **"ul"** ein Elementselektor ist. Durch Voranstellen des ID's **"#navi"** wird jedoch leicht erkennbar, in welchem DIV sich unsere Liste befindet. Verwenden wir dagegen nur **"ul",** gelten die Formatierungen automatisch für alle weiteren Listen und wir könnten verschiedene Listen nicht unterschiedlich definieren.

Zu den Möglichkeiten der Listenformatierung findet ihr hier weitere Hinweise:
http://www.css4you.de/listproperty.html

5.2 Linkformatierung

Kommen wir nun zu unseren Links.
Da nicht auszuschließen ist, dass wir später auch andere Links außerhalb des Menues verwenden, stellen wir in der CSS zunächst einen Bezug zwischen dem Navi-Div und dem Link her (wie wir es auch bereits bei "ul" gemacht haben. Wir geben damit an, dass sich die nachfolgenden Linkformatierungen nur auf Links innerhalb der Navi beziehen.

```
#navi a
{
display: block;
padding: 5px;
width: 170px;
text-decoration: none;
background-color: #FFCC00;
border-bottom: 2px solid #eee;
font-weight: bold;
}
```

Speichert eure "style.css" ab und seht euch das Ergebnis in der "index.html" wieder an.
Das sieht doch jetzt schon viel besser aus.

Was bedeuten nun die einzelnen Angaben?

Der a-Tag ist ein Inline-Element (erinnert euch bitte an 3.1). Er wäre damit vergleichbar einem einzelnen Wort in einem längeren Text. Durch "display: block" haben wir ihn jedoch zum Blockelement umfunktioniert. Vorteil: er nimmt jetzt die gesamte Breite der Navi ein und damit werden alle Links – unabhängig von ihrer Textlänge – in einem gleich großen Block dargestellt. Falls ihr daran Zweifel habt, dann entfernt noch einmal das "display: block;" und seht nach, was passiert. Fügt es dann aber anschließend bitte wieder ein.

Ein weiterer Vorteil von "display: block;" besteht darin, dass später der Hover-Effekt über den ganzen Link-Kasten geht und nicht nur im engen Bereich der Linkbeschriftung.

Padding ist euch bereits bekannt. Es ist der Innenabstand vom Text zu den Rändern.

Kommen wir zu width: 170px:

Wer bisher gut aufgepasst hat wird sich fragen: Wozu denn das? Schließlich haben wir obendrüber bereits a zum Blockelement deklariert. Und von Blockelementen wissen wir, dass sie die gesamte, ihnen zur Verfügung stehende Breite (in diesem Fall sind das die 180px Breite des #navi div's) automatisch einnehmen.
Also könnten wir die Width-Angabe doch auch weglassen. Stimmt – zumindest für alle "intelligenten" Browser. Löscht die width-Angabe aus der CSS und ihr werdet im Firefox und IE 7 keinerlei Unterschied erkennen.
Wer sich die Seite dagegen im IE 6 betrachtet, wird über das Aussehen der Links mit den großen Zwischenräumen doch etwas erstaunt sein. Also erweisen wir erneut dem Internet-Explorer 6 die "Ehre" und setzen allein ihm zu Liebe einen width-Wert. Wir könnten auch darauf verzichten, indem wir die li-Tags neben- statt untereinander schreiben, aber das würde zu unübersichtlich. Alternativ könnten für li auch noch ein "display: inline" setzen.
Ihr habt erkannt: CSS ist einfach (gebe es den IE 6 nicht).

Warum nehmen wir nun eine Breite von 170px, wenn unsere Navi 180px breit ist?
Dafür müssen wir wieder unser Boxmodell aus Kapitel 3.2 bemühen: Die Gesamtbreite ergibt sich aus dem eigentlichen Breitenwert zuzüglich Padding- und Borderwerten. Hier haben wir durch 5px Padding (jeweils links und rechts) bereits 10px an Breite in Anspruch genommen, so dass für die Restbreite zur Navi eben nur noch 170px verbleiben.

Ihr wisst sicher, dass Links von Natur aus unterstrichen sind, damit man sie als solche erkennt. Nachdem wir unser Menue jedoch betitelt haben, sollten die Links auch ohne Unterstrich als solche erkennbar sein.
Wir entfernen ihn deshalb mit

text-decoration: none;

Die beiden anderen Angaben sollten euch bereits aus früheren Kapiteln bekannt sein.

Kommen wir zu weiteren Feinheiten, die es bei den Links zu beachten gilt:

Ein Link kann folgende Zustände einnehmen:

a: link
Das ist der Normalzustand oder die Grunddarstellung.

a: visited
So sieht der Link aus, nachdem man ihn geklickt hat. Man soll also erkennen, dass man die entsprechende verlinkte Seite bereits besucht hat.

a: hover
In diesem Zustand befindet sich der Link, wenn man "mit der Maus drüberfährt"

a: active
Das ist der Linkzustand gerade in dem Moment, wo man mit der Maus draufklickt.

Ich persönliche erachte "active" nicht als so wichtig und werde es deshalb vernachlässigen.

Sämtlichen vier Linkzuständen können wir per CSS nun unterschiedliche Formatierungen zuweisen.

Wichtig:
Achtet unbedingt darauf dass ihr in der CSS Datei immer die Reihenfolge link, visited, hover, active einhaltet. Sonst funktionieren die Formatierungen nicht im Sinne des Erfinders.
Ihr könnt euch die Reihenfolge mit der Eselsbrücke Love Hate für die entsprechenden Anfangsbuchstaben der Linkzustände merken.

Die unter #navi a hinterlegten Angaben gelten für sämtliche Linkzustände. Wir müssen sie nun dafür nicht mehr wiederholen, sondern beschränken uns lediglich noch auf abweichende Angaben.

Schreiten wir also zur Tat und bringen für unsere Linkdarstellungen noch etwas Farbe ins Spiel.

Für die Normaldarstellung des Links verwenden wir z.B.

```
#navi a:link
{
color: #0000CD;
}
```

Bereits besuchte Links sollen so aussehen:

```
#navi a:visited
{
color: #F00;
background-color: #C1CDCD;
}
```

Für den Hover-Link schlage ich vor, mal die Farben des Normalzustandes "umzudrehen":

```
#navi a:hover
{
color: #FFCC00;
background-color: #0000CD;
}
```

Die einzelnen Werte muss ich jetzt sicher nicht mehr erläutern, ihr kennt euch da schon bestens aus. Euch gefallen die Farben nicht? Dann ändert sie doch einfach!

5.3 Menue-Überschrift

Bleibt zum Schluss noch, die Menue-Überschrift etwas aufzupeppen.
Wir haben diese mit "h3" ausgezeichnet und dementsprechend können wir in der CSS wieder eine Zuordnung vornehmen.

Dort haben wir bereits "h1" formatiert. Schreibt zu besseren Übersichtlichkeit die Angaben für "h3" direkt darunter.

Wir wollen für die Überschrift "h3" folgende Werte verwenden:

Schriftfarbe: #FFB90F
Textausrichtung: zentriert
Hintergrundfarbe: #000080
Schriftabstand oben und unten: je 10px
Außenabstand des Überschriftenblocks nach oben und unten: 0px

Nachdem unsere Navi ja jetzt nicht mehr zu übersehen ist, sollten wir den ursprünglichen Navi-Rahmen entfernen und ebenfalls wieder den height-Wert löschen.

Bevor ich euch die Lösung verrate, versucht mal selbst, das hinzubekommen.

Farben sind Geschmackssache. Verwendet deshalb eure eigenen, die euch am besten gefallen. Das Ganze soll hier auch lediglich der Demonstration dienen und ist deshalb mitunter etwas zu bunt geraten. Bei einer "normalen" Homepage solltet ihr unbedingt auf eine gewisse Farbharmonie achten.

5.4 Unsere CSS-Datei im Überblick

Meine "style.css" und damit auch die Lösung zu 5.3 sieht nun so aus:

```
body
{
background-color: #FCF6CF;
font-family: Verdana, Arial, sans-serif;
font-size: 100.01%;
width: 100%;
height: 100%;
margin: 0;
padding: 0;
}

h1
{
color: #000080;
text-align: center;
text-decoration: underline;
}

h2
{
color: #006400;
margin: 0;
}

h3
{
color: #ffb90f;
text-align: center;
background: #000080;
padding: 10px 0;
margin: 0;
}


#header
{
border-top: 3px solid #990000;
border-right: 3px solid #990000;
border-bottom: 3px double #003300;
border-left: 3px double #003300;
background: #FFCC00 url(banner1.gif) no-repeat center;
margin: 15px 15px 30px;
}
```

```
#navi
{
width: 180px;
margin-left: 15px;
font-size:0.9em;
float:left;
display: inline;
}

#content
{
border: 1px solid #8A2BE2;
margin: 0 15px 0 215px;
padding:15px;
text-align: justify;
font-size: 0.9em;
line-height: 1.5em;
}
/*--mit der folgenden Klasse "rotfett" können einzelne Wörter
oder Textstellen rot und fett dargestellt werden--*/
.rotfett
{
color: #f00; /*--Schriftfarbe: rot--*/
font-weight: bold; /*--Fettschrift--*/
}

#navi ul
{
margin: 0;
padding: 0;
list-style-type: none;
}

#navi a
{
display: block;
padding: 5px;
width: 170px; /*--wichtig für den IE 6--*/
text-decoration: none;
background-color: #FFCC00;
border-bottom: 2px solid #eee;
font-weight: bold;
}

#navi a:link
{
color: #0000CD;
}
```

```
#navi a:visited
{
color: #F00;
background-color: #C1CDCD;
}

#navi a:hover
{
color: #FFCC00;
background-color: #0000CD;
}
```

5.5 Wie kommt ihr zu euren anderen Seiten?

Bisher haben wir ja lediglich eine Seite (unsere "index.html") perfektioniert. Die anderen sollen ebenfalls das gleiche Layout erhalten, was durch unsere "style.css" gewährleistet ist.
Ihr habt euch über die endgültige Navi und den Header Gedanken gemacht und strebt für diese Elemente keine Änderungen mehr an (Formatierungen könnt ihr ja weiterhin unbegrenzt über die CSS vornehmen).

Dann schlage ich folgendes vor:
Macht euch von der "index.html" eine Kopie. Streicht aus dieser Kopie alles, was rein spezifisch nur auf der Startseite vorkommt. Also z.B. nur den Text samt Überschrift im Content. Löscht dabei aber keinesfalls die einzelnen Div's, sondern lediglich deren Inhalt.
Die Navi soll überall die gleiche sein. Deshalb hier den Inhalt (die einzelnen Links) keinesfalls löschen.
Speichert die Seiten dann als Einzelseiten ab, und benennt sie wie bei euren Links bereits vorgesehen. Ihr habt euch deshalb bereits darüber vorher Gedanken gemacht, siehe 5.1

Füllt dann die anderen Seiten mit entsprechendem Inhalt eurer Wahl.

Teil 6

6. Ein neues Layout gefällig?

Wir nähern uns bereits dem Ende unseres CSS-Einsteigerkurses. Eure Kenntnisse und Fähigkeiten sollten mittlerweile – so hoffe ich doch – so weit gediehen sein, dass ihr ein weiteres Layout nach entsprechenden Vorgaben in Eigenregie umsetzen könnt.
Habt ihr noch Lust auf eine kleine "Abschlussarbeit", sozusagen euer "Gesellenstück"?

Gut, dann packen wir's gemeinsam an und wollen diesmal ein "fixes" Layout mit genau definierten Breitenangaben erschaffen. Wir machen das in moderner Arbeitsteilung: Ich geb' euch die Vorgaben und ihr arbeitet!

Die einzelnen Schritte werde ich jetzt zur Vereinfachung nicht mehr titulieren aber dennoch nummerieren, damit wir uns bei eventuell auftretenden Problemen besser orientieren können.
Arbeitet bitte genau nach Vorgabe. Interpretiert nichts zusätzliches hinein, lasst aber auch nichts weg. Speichert nach jedem Arbeitsschritt immer die jeweils geänderte Datei ab und seht euch das Ergebnis an.
Hat eine Angabe mal nicht durchgeschlagen, dann verzagt nicht gleich. Meist liegt es sicherlich nur an einem Schreibfehler oder an einem einzelnen Zeichen (ich hatte im Kursteil verschiedene Hinweise gegeben, auf die ihr achten solltet). Falls ihr mal einen "Hänger" habt, dann meldet euch (bevor ihr ganz durchdreht und die Laune verliert) über's Board unter Angabe des Problems.
Zum Schluss sollte jedenfalls jeder das gleiche Ergebnis haben.
Auf geht's Leute!

6.1
Erstellt euch einen neuen Ordner für unsere Abschlussarbeit.
Verwendet den unter 1.2 dargestellten Quelltext, ändert den Title in "Abschlussarbeit", entfernt den Text im body und speichert das Ganze wieder als "abschluss.html" ab.
Erstellt eine (noch leere) CSS-Datei und speichert diese als "abschluss.css" ab.

6.2
Stellt eine Verbindung zwischen beiden Dateien her, indem ihr die "abschluss.css" in die "abschluss.html" als Stylesheet einbindet.

6.3
Weist dem "body" folgende Angaben zu:
Hintergrundfarbe: #F0F8FF
die Schriftart Verdana
die Schriftgröße von 100,01 %
eine Breite von 100 %
eine Höhe von 100 %
einen Außenabstand von 0
einen Innenabstand von 0

6.4
Fügt in eure "abschluss.html" einen Container mit der Bezeichnung "wrapper" ein.
Schreibt darein "Ich bin der wrapper"

Der wrapper soll folgende Eigenschaften haben:
Breite: 750 Pixel
einfacher Rahmen mit 1 Pixel Rahmenstärke und der Farbe #8A2BE2

6.5
Zentriert den wrapper horizontal.
Da wir das noch nicht behandelt haben hierzu folgende Hilfestellung:
Eine horizontale Zentrierung erreicht ihr über
margin: 0 auto

6.6
Schiebt den wrapper noch etwas nach unten indem ihr ihm einen oberen Abstand von 20 Pixeln zuweist.

6.7
Packt in den wrapper eine weitere Box für den Header und gebt ihr die Bezeichnung "kopf"

6.8
Verseht den "kopf" mit der Hintergrundfarbe #A1F2A1
Schreibt in den "kopf" eine Überschrift h1 mit dem Text "Abschlussarbeit".
Zentriert die Überschrift,
gebt ihr die Farbe #FF80FF
und eine Schriftgröße von 2,1 em sowie
den Außenabstand: 0 Pixel

6.9
Fügt unterhalb des "kopf" aber innerhalb des "wrapper" eine weitere Box für die Navi mit der Bezeichnung "menue" ein.

Hinterlegt für das menue folgende Angaben:
Breite: 140 Pixel
Außenabstände: links: 10 Pixel, sonst überall 0
Hintergrundfarbe: nach eurer Wahl
es soll links stehen und rechts umflossen werden

6.10
Fügt in das menue eine Überschrift h4 mit dem Text "Mein Menue" ein und
ordnet dieser Überschrift folgende Werte zu:
Hintergrundfarbe: #FF80FF
Schriftfarbe: #FFF
Außenabstand oben: 20 Pixel
Außenabstand unten: 0 Pixel
Innenabstände: 5 Pixel

6.11
Fügt in das Menue (unterhalb der Überschrift) eine "unordered list" mit insgesamt fünf Links ein. Verwendet dabei als Bezeichnung Link 1, Link 2 .....

6.12
Sorgt dafür, dass in der Liste die Aufzählungspunkte verschwinden.
Setzt die Außen- und Innenabstände der Liste auf 0.

6.13
Sämtliche a-Tags innerhalb des menues sollen
nicht unterstrichen sein,
einen Innenabstand von 5 Pixeln haben
eine Schriftgröße von 0,9 em haben
als Blockelement definiert werden

6.14
Denkt auch daran, dass die Benutzer des Internet-Explores 6 ein ordentliches Menue sehen möchten.

6.15
Im Normalzustand und im besuchten Zustand
soll der Link
die Hintergrundfarbe #CEBCE7
die Schriftfarbe #553186
einen unteren Rahmen mit 1 Pixel Stärke und der Farbe #FFF
haben.

<<<
Hilfestellung: Ihr könnt das zusammenfassen mit dem CSS-Aufruf:
#menue a:link, a:visited
>>>

6.16
Im Hover-Zustand soll der Link
die Hintergrundfarbe #553186
die Schriftfarbe #FFF
fette Schrift
einen linken Rahmen mit 10 Pixeln Stärke und der Farbe #FF00FF
einen unteren Rahmen mit 1 Pixel Stärke und der Farbe #FF00FF
einen rechtsbündigen Text
haben.
Stellt bitte noch sicher, dass beim Hovern die Linkbreite nicht über die Navibox hinausragt.

6.17
Fügt im Anschluss an das Menue und noch innerhalb des wrappers einen weiteren Container mit der Bezeichnung "inhalt" ein.
Gebt ihm eine Hintergrundfarbe eurer Wahl und einen Innenabstand von 10 Pixeln.
Er soll rechts stehen und links umflossen werden und eine Breite von 550 Pixeln haben.

Schreibt in die Inhaltsbox einen beliebigen Text und formatiert ihn nach eurer Wahl

6.18
Fügt nach dem Inhalt noch einen Footer mit der Bezeichnung "fuss" ein.

6.19
Schreibt in den "fuss" das copy-right mit eurem Namen.
Setzt diesen Text rechtsbündig
mit einem rechten Innenabstand von 20 Pixeln,
verwendet die Hintergrundfarbe #FF00FF
und gebt dem "fuss" eine Höhe von 20 Pixeln

6.20
Achtung!
Ihr seid mittlerweile Fortgeschrittene und dürft euch deshalb an den letzten Merkspruch aus Kapitel 4.4.1 erinnern. ("Wer floatet, muss auch clearen!")
Hilfestellung:
Damit der Footer auch unten ankommt (wo er hingehört) müsst ihr ihn aus dem float von menue und inhalt nehmen. Weist ihm deshalb noch ein "**clear: both;**" zu.

6.21
Zugabe
Ihr wisst mittlerweile, dass Blockelemente immer die Höhe ihres Inhalts annehmen, sofern ihre Höhe nicht definiert ist.
Deshalb haben auch Menue und Inhalt unterschiedliche Höhen, was bei unterschiedlichen Hintergründen nicht immer schön aussieht. Ihr könntet ihnen nun jeweils die gleiche Höhe zuweisen um dieses Problem zu beheben.
Aber ein solcher Wert ist nicht von großem Nutzen, wenn ihr den Inhalt ausweitet oder verkürzt, die Schriftgröße ändert, später eventuell noch Grafiken einfügt, oder die Besucher andere Bildschirmauflösungen haben, als bei eurer Seitenentwicklung verwendet.
Wir lösen das Problem mit "faux-columns" wie hier http://alistapart.byteshift.de/fauxcolumns beschrieben. Weil ihr bisher alle so fleißig wart, hab ich euch diese Arbeit mal ausnahmsweise abgenommen, ihr könnt das aber gerne einmal selbst probieren:

Zu diesem Zeck habe ich euch die Hintergrundgrafik fauxclumn.gif gebastelt.
Weil sie "Übergröße" hat, könnt ihr sie euch hier downloaden:
http://ptest.pt.funpic.de/css1/fauxcolumn.gif

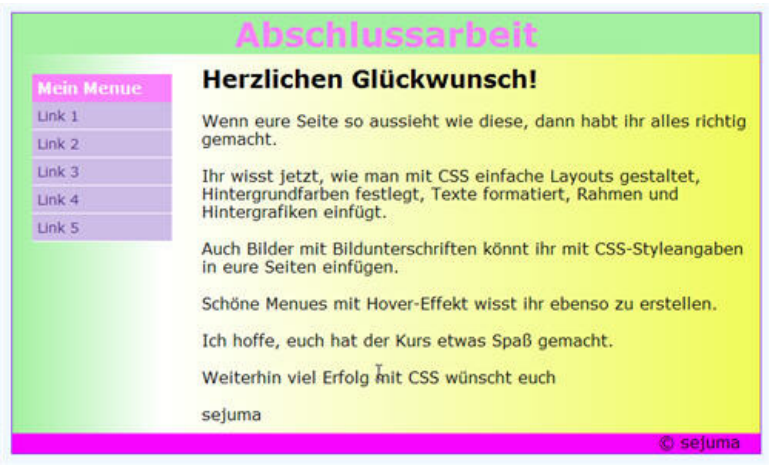
Sie ist wie unser wrapper insgesamt 750 px breit. Für die Höhe habe ich 20px gewählt; selbst ein Wert von 1px würde seinen Zweck erfüllen, da die Grafik ja gekachelt wird.

Ich habe als Hintergrund eine Grafik mit 150px von grün nach weiß verlaufend (entspricht der Menue-Breite 140px + 10px margin) und eine weitere mit 600px von weiß nach gelb verlaufend (entspricht der Rest-Breite) eingefügt. Mit einem geschickten Farbverlauf kann man hier dank PI (PhotoImpact) oder einem anderen Bildbearbeitungsprogramm sehr gute Effekte erzielen, so dass die Trennung zwischen Navi und Content nicht auffällt.

Löscht jetzt mal aus **"#menue"** und **"#inhalt"** jeweils die background-color.
Löscht aus dem wrapper noch den blöden Text "Ich bin der Wrapper"
Fügt in den wrapper die Grafik fauxcolumn.gif als Hintergrundgrafik ein.

Die Wirkungsweise dieser "Zaubergrafik" ist folgende: Weil der **"#wrapper"** die jeweils größte Höhe eines seiner Inhaltselemente annimmt (das kann sowohl **"#menue"** als auch **"#inhalt"** sein) dehnt er sich über beide Container entsprechend aus. Mit ihm auch die gekachelte Hintergrundgrafik.

Fertig und alles richtig? Glückwunsch!
Dann sollten wir jetzt mal unsere Seiten vergleichen.
Wenn jeder nach Anleitung gearbeitet hat, sollten die Ergebnisse auch gleich aus sehen.
Mein Ergebnis sieht übrigens so aus (HTML- und CSS-Code hierzu siehe Anhang):



7. Schluss:

Damit sind wir bereits am Ende unseres CSS-Einsteiger-Kurses angelangt.

Ihr habt die Wirkungsweise und die Vorteile von CSS erkannt und verstanden?

War's zu schwer? Unverständlich? Zu kompliziert?

Habt ihr in der Abschlussarbeit (hoffentlich) auch Fehler gemacht, indem mal ein Semikolon vergessen wurde, sich zwischen Wertangabe und Einheit noch ein Leerzeichen befand oder mal eine geschweifte Klammer in der CSS fehlte?
Sehr gut! Dann habt ihr ja die gleichen Erfahrungen wie ich (und vermutlich jeder andere) gesammelt und wisst, worauf künftig zu achten ist.

Oder hat es möglicherweise sogar etwas Spaß gemacht?

Haut jetzt alle kräftig auf mich ein. Ich hab' ein dickes Fell und freue mich auf eure Kritik.

Den gesamten Kurs gibt es übrigens auch als PDF zum Download hier:
http://www.friedels-home.com/index.htm?/Kurse/CSSsejuma/Teil01/css1.pdf

Habt ihr Anregungen was man noch verbessern oder evtl. in einem Fortgeschrittenen-Kurs vertiefen könnte? Immer her damit! Wir sind bestrebt, euch so weit wie möglich zu unterstützen.

Für mich war es eine ganz neue Herausforderung, der ich mich aber gerne gestellt habe und die mir Dank eurer aktiven Mitarbeit sehr großen Spaß gemacht hat.

Weiterhin viel Erfolg mit CSS
wünscht euch

sejuma

Anhang:

HTML-Code zur Abschlussarbeit:

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

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

</head>
<body>
<div id="wrapper">

<div id="kopf">
<h1>Abschlussarbeit</h1>
</div> <!--Ende kopf-->

<div id="menue">

<ul>
<li><h4>Mein Menue</h4></li>
<li><a href="#">Link 1</a></li>
<li><a href="#">Link 2</a></li>
<li><a href="#">Link 3</a></li>
<li><a href="#">Link 4</a></li>
<li><a href="#">Link 5</a></li>
</ul>
</div>

<div id="inhalt">
<h2>Herzlichen Glückwunsch!</h2>
<p>
Wenn eure Seite so aussieht wie diese, dann habt ihr alles richtig
gemacht.
</p>
<p>
Ihr wisst jetzt, wie man mit CSS einfache Layouts gestaltet,
Hintergrundfarben festlegt, Texte formatiert, Rahmen und
Hintergrafiken einfügt.
</p>
<p>
Auch Bilder mit Bildunterschriften könnt ihr mit CSS-Styleangaben in
eure Seiten einfügen.
</p>
<p>
Schöne Menues mit Hover-Effekt wisst ihr ebenso zu erstellen.
</p>
<p>
Ich hoffe, euch hat der Kurs etwas Spaß gemacht.
</p>
```

```
<p>
Weiterhin viel Erfolg mit CSS wünscht euch
</p>
<p>
sejuma
</p>
</div><!--Ende inhalt-->

<div id="fuss">© sejuma</div><!--Ende fuss-->

</div><!--Ende wrapper-->
</body>
</html>
```

CSS-Code zur Abschlussarbeit:

```
body
{
background-color: #F0F8FF;
font-family: Verdana;
font-size: 100.01%;
width: 100%;
height: 100%;
margin: 0;
padding:0;
}

#wrapper
{
width: 750px;
border: 1px solid #8A2BE2;
margin: 20px auto;
background-image: url(fauxcolumn.gif);
}

#kopf
{
background-color: #A1F2A1;
}

h1
{
text-align: center;
color: #FF80FF;
margin:0;
font-size: 2.1em;
}

h4
{
background-color:#FF80FF;
color: #fff;
margin:20px 0 0 0;
padding:5px;
}

#menue
{
width: 140px;
margin:0 0 0 10px;
float: left;
}
```

```
ul
{
list-style-type: none;
margin: 0;
padding: 0;
}

#menue a
{
text-decoration: none;
padding:5px;
font-size: 0.9em;
display: block;
width: 130px;
}


#menue a:link, a:visited
{
background-color: #CEBCE7;
color: #553186;
border-bottom: 1px solid #fff;
}

#menue a:hover
{
background-color: #553186;
color: #FFF;
font-weight: bold;
border-left: 10px solid #FF00FF;
border-bottom: 1px solid #FF00FF;
text-align: right;
width: 120px;
}

#inhalt
{
width: 550px;
padding: 10px;
float: right;
}


#fuss
{
text-align: right;
padding-right: 20px;
background-color: #FF00FF;
height:20px;
clear: both;
}
```