Arbeiten mit Stylesheets : DIV-Container

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 oder 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 – 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.

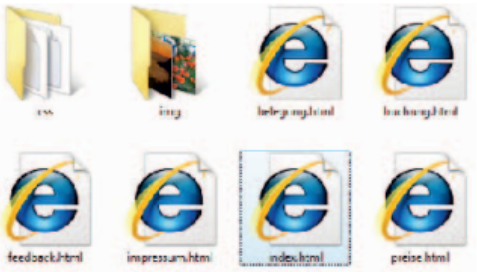
Damit du das Beispiel auch dann nutzen kannst, wenn du zufällig gerade keine kleine Firma bist, habe ich der Musterseite einen „Vereins-Anstrich" verpasst und die Textmenge (= Schreibarbeit) auf das Allernötigste reduziert. Auch das Layout kannst du – wenn du das Prinzip einmal begriffen hast – problemlos 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 Anzeigegeräten (Handheld-PCs, TV-Set-top-Boxen) gibt.

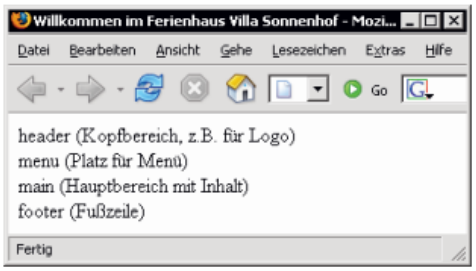
Schritt für Schritt zum Erfolg

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 jedoch 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 *identifier*) 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 Mausrad 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 im 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
float		Textumfluss (Schwebezustand)
	none	Voreinstellung, kein Schweben und kein Textumfluss
	left	Element steht links, darunter notierter Inhalt fließt rechts vorbei
	right	Element steht rechts, darunter notierter Inhalt fließt links vorbei
clear		Beendigung des „float"-Textflusses
	left	Element steht unterhalb der mit clear: left gefloateten Boxen
	right	Element steht unterhalb der mit clear: right gefloateten Boxen
	both	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.

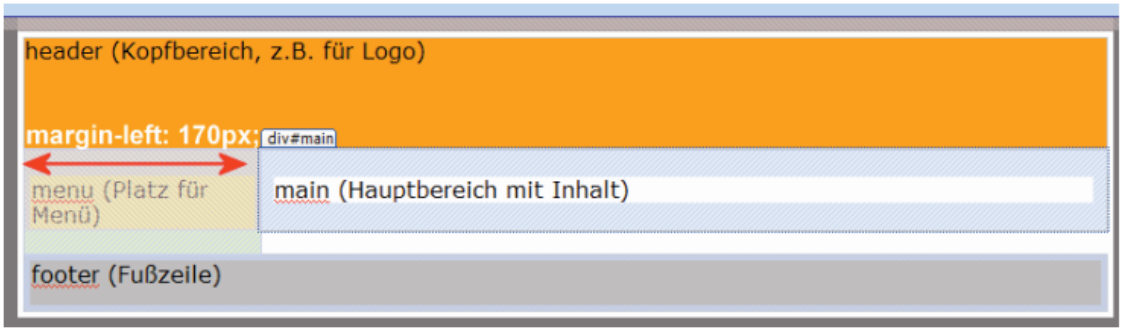
> Leider bereitet diese Variante Probleme im Internet Explorer. Wenn der Inhalt in der Hauptspalte zu breit wird – beispielsweise durch eine breite Tabelle oder ein Bild mit Überbreite – rutscht der rechte Bereich häufig nach unten weg. Neben dem Menü ist dann einfach nicht mehr genug Platz.

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.