PEARSON Education

ADDISON-WESLEY

Dan Cederholm

Bulletproof WebDesign

absolut flexibel und für alles gewappnet mit CSS und XHTML

ADDISON-WESLEY

An imprint of Pearson Education

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

3 Dehnbare Reihen

Nehmen Sie keine festen Höhen und planen Sie eine vertikale Ausdehnung horizontaler Seitenkomponenten.

Horizontale Seitenelemente wie Site-Header, Login- und Suchleisten sowie Breadcrumbs sind ein vertrauter Anblick auf typischen Websites. Diese Bereiche befinden sich oft im oberen Seitenabschnitt und können eine Mischung aus Grafiken (z.B. Hintergründe) und Text enthalten. Im Allgemeinen sind diese Bereiche so gestaltet, dass sich eine vertikale Erweiterung verbietet – hier wird angenommen, dass ein größerer Schriftgrad oder mehr Inhalt entweder nie passiert oder das Design nicht beeinträchtigt. Während es üblich ist, dass Bereiche mit Artikeln oder langen Textabschnitten sich an jede Länge oder Größe von Text oder Inhalten anpassen, ist es wichtig (und nicht unmöglich!), dass auch andere horizontal ausgelegte Bereiche so behandelt werden.

Dieses Kapitel untersucht einen üblichen Ansatz, um einen Login- bzw. Werbebereich zu designen, der sich im oberen Bereich einer typischen Webseite befindet. Wir nehmen das Design auseinander und setzen es dann wieder neu zusammen, damit es sich an beliebige Textmengen oder Schriftgrößen anpassen kann.

Der herkömmliche Ansatz

Damit wir eine Vorstellung davon bekommen, was es heißt, diesen horizontalen Seitenkomponenten eine vertikale Ausdehnung zu erlauben, wollen wir uns ein Beispiel aus dem Web ziehen – eine Site, die wir *Der Super-Shop* nennen wollen (seine wahre Identität bleibt verborgen, um die Schuldigen zu schützen). *Der Super-Shop* ist eine ganz normale e-Commerce-Site, über die eine Vielzahl von teils nützlichen Produkten für Ihr Zuhause verkauft werden. Wir haben die Site für dieses Kapitel erfunden, aber das Design basiert auf realen Techniken. Um eine übliche Herangehensweise zu demonstrieren, wie eine horizontale Login- bzw. Werbeleiste designt wird, schauen wir uns diesen Bereich auf der Website von *Der Super-Shop* genau an (Abb.3.1). Ganz oben auf jeder Seite sitzt ein farbiger Balken, der das Login und Infos über Filialen enthält, danach kommt eine zweiter Bereich mit einer Werbebotschaft, die regelmäßig aktualisiert wird. Jede Reihe enthält nicht mehr als eine Zeile Text.

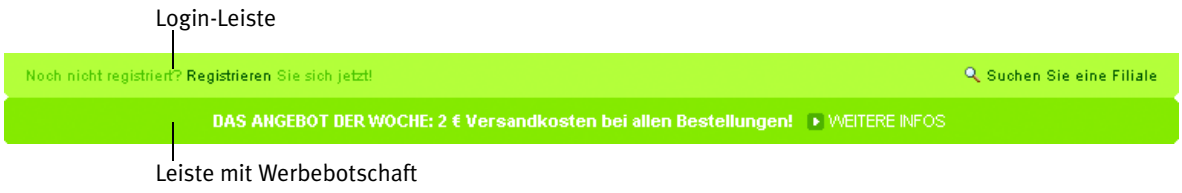

Abbildung 3.1 Dies ist der obere Bereich unserer fiktiven Website *Der Super-Shop*.

Diese beiden Reihen (so wie das gesamte Layout) wurden über eine Serie verschachtelter Tabellen konstruiert. Grafiken (wie die abgerundeten Ecken jeder Reihe) und Text sind in den Tabellenzellen platziert.

Abbildung 3.2 illustriert, wie die Tabellenzellen der oberen Reihe strukturiert sein könnten; jede Zelle ist rot umrandet. Sie sehen, dass sich jeder Abschnitt der Reihe in einer eigenen separaten Tabellenzelle befindet, einschließlich der Grafiken mit den abgerundeten Ecken an jedem Ende. Dies ist eine *Annäherung* an die Tabellenstruktur, ohne zu tief in den Code von *Der Super-Shop* einzusteigen. Der wichtige Punkt ist, dass Tabellen, Blind-GIFs und kleine Grafiken so kombiniert wurden, dass sie die beiden hier abgebildeten Reihen formen.

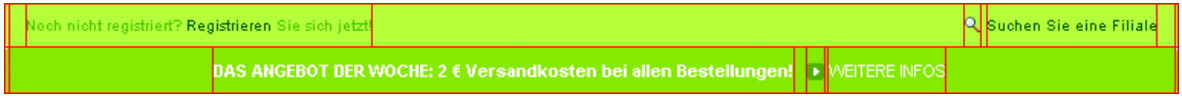

Abbildung 3.2 Jeder Abschnitt der Reihe sitzt in einer eigenen separaten Tabellenzelle.

Die Verwendung von Tabellen und Blind-GIFs zur Positionierung von Grafiken und Text ist eine Technik, die im Laufe vieler Jahre verfeinert wurde – auf diese Weise werden die meisten Websites erstellt, und eine Menge Designer rühmen sich ihrer Fähigkeit, jedes beliebige Designkonzept im Web zu replizieren – bis zum letzten Pixel genau. Falls Sie es designen und *ausdrucken* könnten, könnten Sie es in eine Webseite verwandeln.

Aber wir lernen hier bessere Wege der Gestaltung im Web. Und wir entdecken Methoden, um die Lesbarkeit und Accessibility durch schlankes, semantisches Markup und CSS für das Design zu erhöhen. Ein wenig später werden wir diese Methoden bei den zwei horizontalen Reihen von *Der Super-Shop* anwenden, aber zuerst wollen wir darüber sprechen, warum dieses Design nicht kugelsicher ist.

Warum dieses Design nicht kugelsicher ist

Wir können verschiedene Verbesserungen bei der Konstruktion der horizontalen Reihen vornehmen, die die Flexibilität dieses Abschnitts erhöhen. Zuerst wollen wir umreißen, was verkehrt ist, damit wir besser verstehen, was bei diesem Design hier kugelsicher zu machen ist.

Unnötige Grafiken

Wie wir schon gesehen haben, wurden die Reihen über eine Serie verschachtelter Tabellen erstellt, bei denen jeder Abschnitt der Reihe von einer eigenen Tabellenzelle umschlossen wird. *Verzichtbare Grafiken* wie die abgerundeten Ecken sind im Markup neben dem Text platziert. Diese Grafiken werden nur hinderlich für jemandem sein, der über einen Textbrowser oder ein Bildschirmlesegerät auf diese Site zuzugreifen versucht. Wenn den Bildern die passenden `alt`-Attributwerte zugewiesen werden, könnte das sicher helfen, obwohl *Der Super-Shop* sich dafür entschieden hat, diese nützlichen Infos wegzulassen.

»Verzichtbare« Grafiken bezieht sich auf solche, die dem Inhalt keine weitere Bedeutung hinzufügen oder die keine Anweisungen oder nähere Angaben für den Anwender enthalten. Abgerundete Ecken sind ein *Designelement*, und wir werden diese später in das Stylesheet auslagern.

Fixierte Höhe

Wenn wir versuchen, den Schriftgrad etwas heraufzusetzen, wird Ihnen auffallen, dass das Design der Reihen zusammenbricht (Abbildung 3.3). Ein Erhöhen des Schriftgrads ist immer ein guter Test, um herauszufinden, wie flexibel Designkomponenten sind. Sie bekommen nicht nur eine Vorstellung dafür, wie gut die Lesbarkeit bei größerem Text ist, sondern finden auch heraus, ob das Design ungeachtet der Textgröße unterschiedliche Mengen von Inhalten bewältigen kann. Anders gesagt, wenn ein Redakteur später einmal *zwei* Zeilen mit Werbetext haben will statt der einen aus der Spezifikation, dann würde sich in einer kugelsicheren Welt die Reihe ohne weitere Anpassungen ausdehnen. Das spart Zeit, Geld und … na ja, das sollte schon Grund genug sein.

Abbildung 3.3 Wenn der Text vergrößert wird, dehnen sich die statischen Eckenbilder nicht mit dem Rest des Designs aus.

Schauen wir uns noch mal Abbildung 3.3 an, dann sehen wir, dass die Bilder für die abgerundeten Ecken, die jede Reihe abschließen, für eine festgelegte Höhe designt wurden. Wird dort etwas Großes hineingepackt, läuft es über und zerschießt das Design. Ich möchte in diesem Kapitel (und vielen anderen) vermitteln, dass es auf eine andere Denkweise ankommt, was die Höhe angeht. Denn es ist möglich, größeren oder umfangreichen Text zu bewältigen oder was sonst noch in bestimmte Designkomponenten gelegt werden soll. Dazu kommen wir gleich.

Aufgeblähter Code

Wie bei den meisten traditionellen Methoden des Webdesigns war eine Menge Code nötig, um diese Reihen zu konstruieren. Wie Sie sich aus dem Kapitel 2 »Skalierbare Navigation« erinnern, vergrößern verschachtelte Tabellen das Markup beträchtlich. Dieser aufgeblähte Code füllt die Server, verstopft die Bandbreite und verursacht Chaos bei Nicht-Browser-Software

und Geräten. Stellen Sie sich mal vor, wie die Navigation mit einem Textbrowser in einem Meer von Code sein muss. Zum Glück gibt es einen saubereren, flexiblen Weg, um die gleichen Designvorgaben zu erzielen. Dazu kommen wir jetzt.

Der Bulletproof-Ansatz

Um das Design der Reihen kugelsicher hinzukriegen, müssen wir uns zuerst an eine Struktur für das Markup machen. Dann fügen wir Farbe, Positionierungen und Hintergrundbilder ein. Schließlich haben wir ein flexibles Endprodukt, das in der Lage ist, beliebige Textgrößen oder Inhaltsmengen zu bewältigen. Wir fangen mit dem Markup an und reduzieren den vorhin erwähnten aufgeblähten Code.

Die Struktur des Markup

Um diese beiden Reihen zu strukturieren, müssen wir uns deren Inhalt anschauen. Welche Elemente wären für dieses Szenario am sinnvollsten? Welche am aussagekräftigsten? Wenn ich das Markup ganz neu schreibe, beantworte ich diese Fragen gerne zuerst und gebe dann dem Projekt die passendste Struktur. Es mag darauf sehr wohl mehr als nur eine Antwort geben, aber Sie sollten sich diese Fragen stellen, bevor Sie den ersten Buchstaben tippen.

In diesem Fall werden wir zwei Container-Elemente brauchen: eins für jede Reihe. Für die obere Reihe sehe ich zwei Textstückchen an jedem Ende als Einträge einer Liste, während es in der zweiten Reihe nur einen Textabschnitt gibt.

Jetzt können wir die komplette Markup-Struktur anfertigen, die wir für dieses Design brauchen:

```
<ul>
  <li>Noch nicht registriert? <a href="/register/">
      Registrieren</a> Sie sich jetzt!</li>
  <li><a href="/find/">Suchen Sie eine Filiale</a></li>
</ul>
<div>
  <p><strong>Das Angebot der Woche:</strong> 2 &euro;
      Versandkosten bei allen Bestellungen!
      <a href="/special/">WEITERE INFOS</a></p>
</div>
```

Wir haben hier also zwei Listeneinträge für die obere Reihe und für die untere ein Container- , das einen Absatz umschließt. Schön und schlicht, sehr schlankes Markup – und das Beste ist, wir haben bereits den aufgeblähten Code rausgeworfen, den der Ansatz mit den verschachtelten Tabellen verbrochen hat.

Zusätzlich haben wir unser Ziel erreicht, dass die Accessibility für den Inhalt verbessert wurde. Egal mit welchem Gerät oder welcher Software diese beiden Reihen mit Informationen gelesen werden, sie werden stets als Aufzählungsliste interpretiert, der ein Absatz folgt. Und das ist genau das, was sie auch sind.

Die Einzelteile identifizieren

Unser nächster Schritt ist, all die Elemente eindeutig zu kennzeichnen, denen wir Styles geben wollen. Wenn wir ein paar ids zuweisen, können wir Positionierung, Farbe und Bilder anwenden, um dieses einfache Markup in das endgültige Design zu verwandeln.

```
<ul id="register">
  <li id="reg">Noch nicht registriert? <a href=
        "/register/">Registrieren</a> Sie sich jetzt!</li>
  <li id="find"><a href="/find/">
        Suchen Sie eine Filiale</a></li>
</ul>

<div id="message">
  <p><strong>Das Angebot der Woche:</strong> 2 _
      Versandkosten bei allen Bestellungen!
      <a href="/special/">WEITERE INFOS</a></p>
</div>
```

Wir haben gerade der Liste und allen ihren Einträgen und auch dem Container- der zweiten Reihe eindeutige ids hinzugefügt . Diese *Style-Hooks* werden gleich noch wichtig.

Ohne Styles

Die Abbildung 3.4 zeigt, wie unsere Markup-Struktur in einem Browser aussieht, wobei nur etwas grundlegendes Font-Styling angewendet wurde (*Der Super-Shop* verwendet generell die Schriftart Arial).

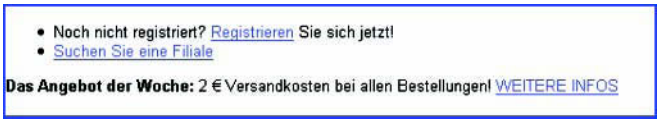

Abbildung 3.4 Dies ist eine ungestylte Ansicht unseres kugelsicheren Markups; es ist für jedes Gerät leicht lesbar und verständlich, das darauf zugreifen will.

hinweis

Bevor wir uns daran machen, die Styles anzuwenden, bauen wir diese Reihen unter der Annahme einer Container-Layoutbreite von 768 Pixel – die Breite der Seite von *Der Super-Shop*. Anders ausgedrückt: Diese Reihen sitzen in einem Container-Element (`<div>`, `<table>` etc.), dem bereits eine Breite zugewiesen wurde.

Dem `<body>`-Element haben wir eine grundlegende Schriftgröße über den Wert des Schlüsselwortes `small` zugewiesen.

```
body {
  font-family: Arial, sans-serif;
  font-size: small;
  }
```

Handys, PDAs oder Browser, die kein CSS unterstützen, würden die Reihen so darstellen. Das Ergebnis ist für alle Geräte, die darauf zugreifen, immer noch gut lesbar und leicht verständlich. Nun beginnen wir, Styles anzuwenden.

Hintergrund einfügen

Um mit dem Einfügen der Styles zu beginnen, wollen wir zuerst die Hintergrundfarbe jeder Reihe angeben. Dieser Schritt hilft uns, die Dimensionen zu definieren.

```
#register {
  background: #BDDB62;
  }
#message {
  background: #92B91C;
  }
```

Wie das Resultat aussieht, nachdem jede Reihe die korrekte Hintergrundfarbe bekommen hat, sehen Sie in Abbildung 3.5.

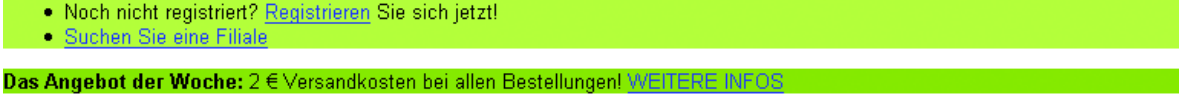

Abbildung 3.5 Wenn wir dem Hintergrund eine Farbe zuweisen, hilft uns das beim visuellen Definieren der Reihen, wenn wir die restlichen Teile hinzufügen.

Der Inhalt wird positioniert

Als Nächstes wollen wir den Inhalt positionieren, indem wir die beiden Listeneinträge der oberen Reihe ans jeweilige Ende setzen und die Info über die 2 € Versandkosten in die Mitte der unteren Reihe. Die Abbildung 3.6 zeigt das Ergebnis der folgenden CSS-Zeilen:

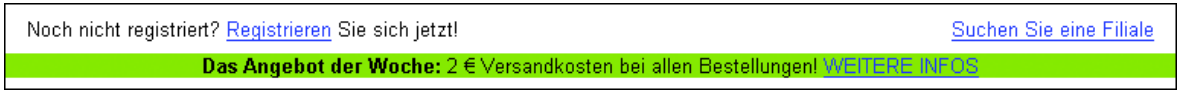

Abbildung 3.6 Wir haben jeden Listeneintrag an die Enden der oberen Reihe gesetzt.

```
#register {
  margin: 0;
  padding: 0;
  list-style: none;
  background: #BDDB62;
  }
#reg {
  float: left;
  margin: 0;
  padding: 8px 14px;
  }
#find {
  float: right;
  margin: 0;
  padding: 8px 14px;
  }

#message {
  clear: both;
  text-align: center;
  background: #92B91C;
  }
```

Von oben nach unten haben wir zuerst Standardränder und Padding von der `#register`-Liste beseitigt. Außerdem haben wir über die Regel `list-style:none;` verhindert, dass Gliederungspunkte auftauchen.

Dann haben wir die Methode der *gegenüberliegenden Floats* benutzt, um die beiden Listenobjekte an die gegenüberliegenden Enden der Reihe zu setzen. Der erste Listeneintrag wird nach links gefloatet (`#reg`) und der zweite nach rechts (`#find`). Das erlaubt uns, jeden Eintrag an der gleichen horizontalen Position, aber doch an den gegenüberliegenden Enden der Reihe auszurichten (Abbildung 3.7).

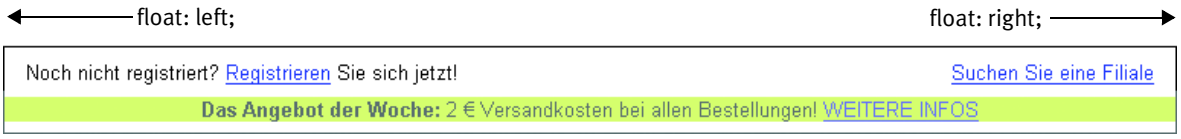

Abbildung 3.7 Die Methode der »gegenüberliegenden Floats«ist ein praktischer Weg, um Inhalt an beiden Seiten eines Containers auszurichten.

Schauen wir uns noch einmal die neu hinzugefügten Styles an – wir haben zusätzlich zum Floaten aller Listeneinträge jedem Objekt auf allen Seiten noch Padding gegeben. Es bleibt auch noch genug Platz, um später das Lupen-Icon links neben dem Link »Suchen Sie eine Filiale« aufzunehmen.

Bei der unteren Reihe haben wir eine `clear: both;`-Regel eingefügt, die den Umfluss der bei der Reihe darüber eingefügten Floats abbricht. Und um den Text zu zentrieren, nehmen wir die `text-align: center;`-Regel.

Fehlender Hintergrund

Warum ist die Hintergrundfarbe der oberen Reihe verschwunden? Sie erinnern sich sicher, dass wir in Kapitel 2 auf ein ähnliches Problem gestoßen sind. Wenn wir innere Elemente floaten (in diesem Fall die beiden `<li>`s), dann nehmen wir sie aus dem normalen Fluss des Dokuments. Für das sie umschließende `<ul>` ist es also, als ob die Listeneinträge nicht wirklich existierten. Als Folge davon weiß das `<ul>` nicht, wie hoch und breit es die Hintergrundfarbe hinter sich ausdehnen soll.

Um dieses Problem zu lösen, floaten wir das `<ul>` gemeinsam mit den Listeneinträgen (so wie wir das auch mit den Tabs aus Kapitel 2 gemacht haben). Weiterhin müssen wir eine `width` zuweisen, um sicher zu gehen, dass die Reihe über die gesamte gewünschte Breite fließt. Es scheint, dass die meisten Browser die Spezifikation von CSS2.0 wörtlich genommen haben, dass »eine gefloatete Box eine explizite Breite haben muss« (`www.w3.org/TR/REC-CSS2/visuren.html#floats`).

Wenn wir hier keine Breite angeben, wird die Reihe nur so breit sein, wie es der Inhalt erzwingt (in diesem Fall die beiden Zeilen Text).

```
#register {
  float: left;
  width: 100%;
  margin: 0;
  padding: 0;
  list-style: none;
  background: #BDDB62;
  }
#reg {
  float: left;
  margin: 0;
  padding: 8px 14px;
  }
#find {
  float: right;
  margin: 0;
  padding: 8px 14px;
  }

#message {
  clear: both;
  text-align: center;
  background: #92B91C;
  }
```

Die Abbildung 3.8 zeigt die Reihe bisher, und die Hintergrundfarbe der oberen Reihe ist auch wieder da.

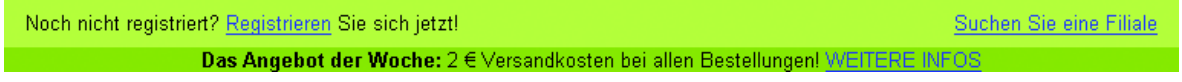

Abbildung 3.8 Wenn Sie Objekte in einem Container floaten, der den Hintergrund füllt, stellen Sie den Hintergrund wieder her, indem Sie die Container ebenfalls floaten.

Details werden hinzugefügt

Jetzt bleibt nur noch, die Details zu ergänzen, die das Design dieser Reihen vervollständigen. Wir beginnen mit der oberen Reihe und fügen die abgerundeten Ecken ein, die an den unteren Kanten beider Enden erscheinen.

Abbildung 3.9 Die abgerundete Ecke besteht in Wirklichkeit aus ein paar weißen, stufig angeordneten Pixeln, die an jede Ecke der Reihe angesetzt sind.

Sie sehen, dass die *abgerundeten* Ecken nichts anderes sind als ein paar weiße Pixel in Treppenform. Werden sie bei normaler Distanz betrachtet (statt in Großaufnahme wie in Abbildung 3.9), dann schafft das die Illusion, die Reihe sei an den Enden abgerundet.

Es ist ein großartiger Trick, Pixel für diese Illusion einer Abrundung abzuhacken – und einer, der über eine Kombination des kleinstmöglichen Bildes mit einer im CSS festgelegten Hintergrundfarbe leicht anwendbar ist.

Wir beginnen, indem wir das Bild in Photoshop erstellen (oder Ihrem bevorzugten Bildbearbeitungsprogramm). Weil wir es mit einer festen Breite zu tun haben (768 Pixel), reicht uns ein Bild, das sowohl die linken als auch die rechten Ecken enthält. Auf dieses Bild können wir in unserem CSS als Hintergrund referenzieren.

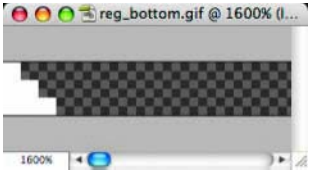

Abbildung 3.10
Dies ist das eine Ende des 768 Pixel breiten GIFs (auf 1600%).

Die Abbildung 3.10 zeigt eine Großaufnahme des gerade erstellten Bildes. Hier sehen Sie die linke Seite eines Bildes, das 768 Pixel breit ist (so breit wie unsere Reihen). Für jede Seite haben wir dieses Stufenmuster mit dem auf 1px eingestellten Zeichenstiftwerkzeug und der Farbe Weiß erstellt (der gleichen Farbe wie beim Seitenhintergrund). Der Rest des Bildes ist transparent (was in Photoshop durch das Schachbrettmuster angezeigt wird). Die weißen Bereiche dieses Bildes werden auf die Hintergrundfarbe gelegt, die wir bereits im CSS festgelegt haben. Damit wird die Illusion möglich, dass die Enden der Reihen um einige Pixel abgerundet wurden.

Schauen wir uns die Deklaration für `#register` an (das `<ul>`-Element) und fügen die folgende Regel ein:

```
#register {
  float: left;
  width: 100%;
  margin: 0;
  padding: 0;
  list-style: none;
  background: #BDDB62 url(img/reg_bottom.gif) no-repeat
bottom left;
  }
```

Damit haben wir eine Hintergrundfarbe festgelegt und das Bild darauf gesetzt, es *nicht* auf Wiederholung gestellt und es am unteren und linken Rand ausgerichtet. In den transparenten Bereichen des Bildes wird die Farbe des Hintergrundes zu sehen sein, während die weißen Ecken sie verdecken werden. Wird das Bild am unteren Rand ausgerichtet, dann bleiben die Ecken garantiert immer an der richtigen Stelle, egal wie groß die Reihe sein muss (abhängig von verschiedenen Schriftgraden oder Textmengen) (Abbildung 3.11).

Abbildung 3.11 Eine 3D-Ansicht der Schichtenreihenfolge

Abbildung 3.12 zeigt die Ergebnisse, wobei die abgerundeten Ecken nun am unteren Rand der oberen Reihe erscheinen.

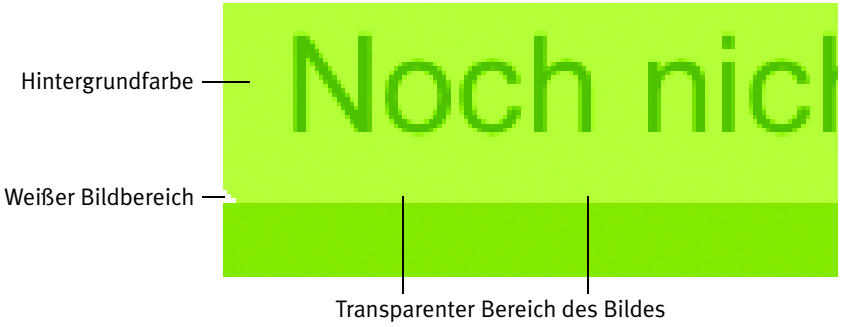

Abbildung 3.12 Die Hintergrundfarbe und die weißen und transparenten Bildbereiche schaffen gemeinsam die Illusion von abgerundeten Ecken.

Vier abgerundete Ecken

Für die zweite Reihe brauchen wir einen Weg, um abgerundete Ecken sowohl oben als auch unten anzubringen, und die Reihe soll trotzdem noch vertikal ausdehnbar bleiben. Dafür werden wir zwei Hintergrundbilder verwenden. Eines wird das selbe Bild sein, das wir für den unteren Bereich der oberen Reihe benutzt hatten, das andere (für den oberen Rand) ist wiederum das selbe Bild, aber auf den Kopf gestellt. Für diese zwei Hintergrundbilder brauchen wir auch zwei Elemente, denen wir sie zuweisen können. (Oh, wie sehr wünsche ich mir, dass ich mal einem einzelnen Element mehr als ein Hintergrundbild zuweisen könnte. Man darf ja mal träumen …).

Was für ein Glück – wir haben bereits zwei einsatzbereite Elemente. Beachten Sie, dass wir im Markup ein `<div>`-Element mit einem `<p>` darin für die zweite Reihe mit dem Inhalt haben:

```
<div id="message">
  <p><strong>Das Angebot der Woche:</strong> 2€
  Versandkosten bei allen Bestellungen! <a href="
  /special/">WEITERE INFOS</a></p>
</div>
```

Diese Hintergrundbilder können wir nun jeweils einem Element zuweisen. Die umgedrehte Version unserer Grafik mit den weißen Ecken kommt in `#message`, und das gleiche Bild, das in der oberen Reihe benutzt wird, wird dem unteren Rand des `<p>` zugewiesen.

```
#message {
  clear: both;
  text-align: center;
  background: #92B91C url(img/mess_top.gif)
    no-repeat top left;
  }
#message p {
  margin: 0;
  padding: 8px 14px;
  background: url(img/reg_bottom.gif) no-repeat bottom left;
  }
```

Indem wir die oberen Ecken `#message` zuweisen (dem äußeren `<div>`) und die unteren Ecken dem unteren Rand des `<p>` (Abb.3.13), stellen wir sicher, dass alle vier Ecken korrekt positioniert bleiben, egal wie groß oder klein der Text im Absatz ist. Wenn wir einen größeren Schriftgrad oder mehr Text wählen, werden die oberen Ecken stets nach oben ausgerichtet bleiben und die unteren Ecken immer an der Unterkante des Absatzes (wie wir gleich sehen werden).

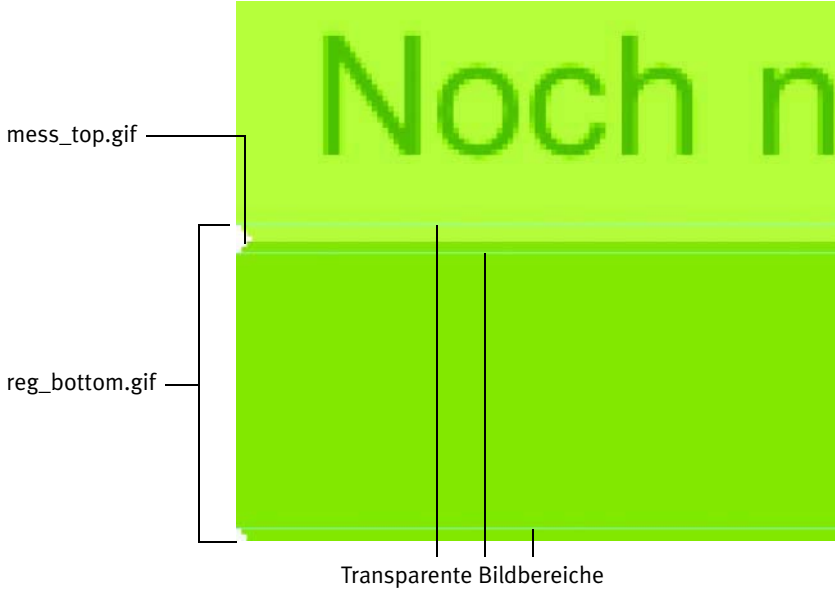

Abbildung 3.13 Für die untere Reihe nehmen wir zwei Bilder und lassen den grünen Hintergrund durch die transparenten Bereiche des GIF scheinen.

In der Abbildung 3.14 sehen wir die Ergebnisse dieser Deklarationen. So wie bei der oberen Reihe verwenden wir die transparenten Bilder, die die Hintergrundfarben durchscheinen lassen, und die weißen Ecken maskieren nur diese vier Bereiche.

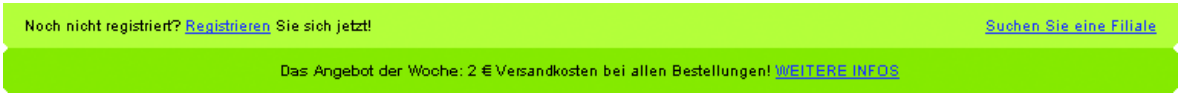

Abbildung 3.14 Durch die eingefügten Hintergründe nehmen die Reihen langsam Gestalt an.

Details von Text und Links

Bis das Design komplett ist, fehlen uns nur noch einige Styles für die Farben von Links und Text. Wir müssen auch die Grafiken wieder einbauen, die die Links »Suchen Sie eine Filiale« und »WEITERE INFOS« flankieren. Dazu kommen wir jetzt.

Als Erstes wollen wir für jede Reihe die Farben von Links und Text definieren, indem wir allen bisher deklarierten Styles die nötigen Regeln hinzufügen:

```
#register {
  float: left;
  width: 100%;
  margin: 0;
  padding: 0;
  list-style: none;
  color: #690;
  background: #BDDB62 url(img/reg_bottom.gif)
    no-repeat bottom left;
  }
#register a {
  text-decoration: none;
  color: #360;
  }
#reg {
  float: left;
  margin: 0;
  padding: 8px 14px;
  }
#find {
  float: right;
```

```
  margin: 0;
  padding: 8px 14px;
  }

#message {
  clear: both;
  font-weight: bold;
  font-size: 110%;
  color: #fff;
  text-align: center;
  background: #92B91C url(img/mess_top.gif)
    no-repeat top left;
  }
#message p {
  margin: 0;
  padding: 8px 14px;
  background: url(img/reg_bottom.gif) no-repeat bottom left;
  }
#message strong {
  text-transform: uppercase;
  }

#message a {
  margin: 0 0 0 6px;
  padding: 2px 15px;
  text-decoration: none;
  font-weight: normal;
  color: #fff;
  }
```

Wir haben die Farben der Links für alle Elemente in der Reihe `#register` eingestellt und auch die Standards für `font-size` und `color` für Text und Link in der Reihe `#message` (Abbildung 3.15).

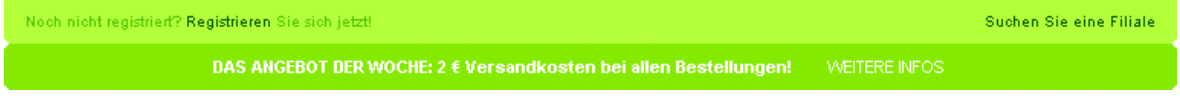

Abbildung 3.15 Die Reihen erhalten Styles für die Farben von Text und Links. Beachten Sie den Platz, der für die kleinen Symbole vorgesehen ist, die links neben »Suchen Sie eine Filiale« und »WEITERE INFOS« sitzen.

Vorher hatten wir den Text »Das Angebot der Woche:« über das Element `<strong>` betont. Wir haben uns zunutze gemacht, dass wir über die Eigenschaft `text-transform` diesen Bereich der Botschaft auf Großbuchstaben umgewandelt haben.

Der letzte Schritt

Als letzten Schritt, um die Reihen kugelsicher zu machen, fügen wir die Grafiken ein, die die Links »Suchen Sie eine Filiale« und »WEITERE INFOS« flankieren. Wir könnten diese Bilder dem Markup zufügen, aber um es später bei einer Aktualisierung einfacher zu haben und unnötige Bilder aus der Dokumentstruktur herauszuhalten, fügen wir sie einfach im CSS als Hintergrundbilder ein.

Zuerst stellen wir das Lupensymbol in der Liste in der oberen Reihe ein und richten es mit `0 50%` aus, damit wird es links und 50% von oben positioniert (vertikal zentriert):

```
#find {
  float: right;
  margin: 0;
  padding: 8px 14px;
  background: url(img/mag-glass.gif) no-repeat 0 50%;
  }
```

In der Abbildung 3.16 sehen Sie, wie dieses Symbol im Listeneintrag »Suchen Sie eine Filiale« ausschaut.

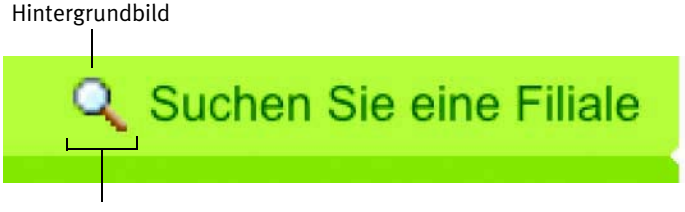

Abbildung 3.16 Das Hintergrundbild steht links neben dem Listeneintrag, wo zuvor Padding eingebaut wurde.

Und als Letztes soll nun auch die Pfeilgrafik an die richtige Stelle. Diese wird auch mit `0 50%` positioniert (also ganz nach links und mittig nach unten (vertikal zentriert):

```
#message a {
  margin: 0 0 0 6px;
  padding: 2px 15px;
  text-decoration: none;
  font-weight: normal;
  color: #fff;
  background: url(img/arrow.gif) no-repeat 0 50%;
  }
```

Die Abbildung 3.17 zeigt, wie es aussieht, wenn die Pfeilgrafik über das <a>-Element, das in der Reihe mit der Botschaft sitzt, links neben den Link gesetzt wird.

Abbildung 3.17 Wie in der oberen Reihe haben wir den Pfeil links neben die Worte »WEITERE INFOS« gesetzt und ihn diesmal an das <a>-Element gehängt.

Die Abbildung 3.18 zeigt das fertige Ergebnis. Visuell haben wir zwei Reihen, die praktisch identisch sind mit denen des *Super-Shops*, aber über die zugrunde liegende Markup-Struktur wird alles gemeinsam mit den Vorkehrungen, über die wir Hintergründe und Text strategisch platziert haben, kugelsicher. Lassen Sie uns herausfinden, warum.

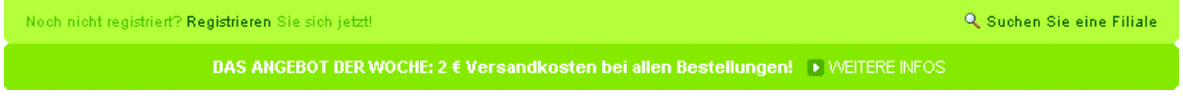

Abbildung 3.18 Dies ist die endgültige, kugelsichere Version der Leisten.

Warum dieses Design nun kugelsicher ist

Nachdem wir das Markup vereinfacht und kreativ kleine Hintergrundbilder verwendet haben, sind die Reihen erfolgreich mit kugelsicheren Methoden nachgebaut. Das schauen wir uns genauer an.

Trennung von Struktur und Design

Wir haben Tabellen und verzichtbare Grafiken hinausgeworfen und das HTML durch schlankes, strukturiertes XHTML ersetzt. Das sinnvolle Markup hat größere Chancen, von mehr Geräten und Software korrekt verstanden zu werden – sogar bei fehlendem CSS.

Anstatt die Bilder, aus denen das Design der Reihen besteht, direkt ins Markup einzubauen, haben wir sie ins Stylesheet verlegt. Falls später Änderungen nötig sein sollten, wird das deutlich einfacher sein, ganz zu schweigen von der drastischen Verringerung von Code.

Wenn beispielsweise diese beiden Grüntöne durch Rot, Blau oder eine andere Farbe ausgetauscht werden sollen, brauchen nur ein paar CSS-Regeln geändert zu werden. Das Design ist *sofort* fertig!

Keine festen Höhen mehr

Anstatt davon auszugehen, dass diese Reihen stets *x* Pixel groß sein werden, haben wir Hintergrundbilder kreativ positioniert und sowohl die Integrität des Design erhalten als auch ermöglicht, dass es bei Bedarf ausdehnt oder schrumpft. Dieser Ansatz erlaubt es uns auch, Methoden zur Festlegung von Textgröße frei anzuwenden (so wie die aus Kapitel 1), ohne dass wir uns für den enthaltenen Text auf eine Pixelvorgabe verlassen müssen.

In der Abbildung 3.19 sehen Sie unsere neu konstruierten Reihen, der Inhalt nun mit deutlich größerem Schriftgrad. Beachten Sie, wie die abgerundeten Ecken und der Hintergrund des Designs intakt bleiben.

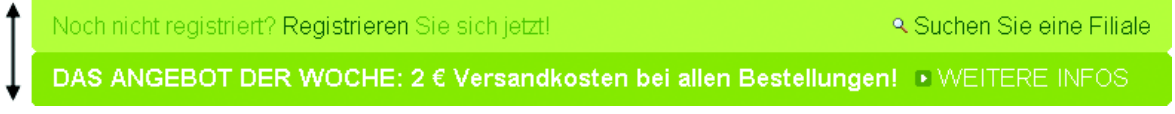

Abbildung 3.19 Bei größerem Schriftgrad dehnen sich die Reihen aus, ohne dass sich das Design ändert.

Nehmen wir an, ein Redakteur will *zwei* Werbebotschaften in der zweiten Reihe unterbringen. Wir können problemlos ohne Störung des Designs eine zweite Zeile einfügen, die sich ja ausdehnt, um die neue Botschaft aufzunehmen (Abbildung 3.20). Das demonstriert den größten Vorteil der kugelsicheren Methoden: Das Design passt sich sogar *unvorhergesehenen* Anforderungen an.

Abbildung 3.20 Wird eine zweite Zeile in die #message-Reihe eingefügt, macht das dem Designer keine Arbeit, weil wir in das Design unbegrenzt Raum eingebaut haben. Schaumgummihammer?!

Wenn ein Kunde oder Manager sagt: »Wir brauchen hier nur Platz für eine Textzeile«, und Sie machen sich gleich dran und bauen es dieser Vorgabe entsprechend, können Sie Gift drauf nehmen, dass es nach einer Woche heißt: »Prima, wir brauchen aber unbedingt Platz für *zwei* Textzeilen!« Aber wenn Sie kugelsicher designen, haben Sie diese Möglichkeit schon längst eingebaut. Aber Sie sollten ihm sagen, dass Sie eine Woche länger daran gewerkelt haben. Oder noch besser: Sie zeigen, dass Sie das schon im Vorfeld bedacht haben, und veranschaulichen somit die Vorteile von CSS, wie Kopfschmerzen bei der Wartung reduziert werden können.

tipp

Sie wissen sicher noch, dass wir die Bilder für Lupe und Pfeil 50% von oben eingefügt haben. Beachten Sie, dass ungeachtet der Textgröße das Bild immer vertikal zum Text *zentriert* sein wird.

Ein anderes Beispiel für Ausdehnung

Um die Vorteile von dehnbaren Reihen an einem anderen Szenario zu demonstrieren, werde ich den Prozess vorstellen, bei dem ich einen flexiblen Header für mein bei Blogger erstelltes TicTac-Template designt habe. Blogger ist ein beliebtes Tool zum Erstellen von Weblogs, das zu Google gehört, und die Applikation stellt neuen Anwendern mehrere vorgefertigte Templates (Vorlagen) zur Verfügung, wenn sie sich ein Weblog einrichten. Es war extrem wichtig, diese Vorlagen kugelsicher zu machen, damit der Header einen beliebig langen Titel für die Site aufnehmen kann.

Die Abbildung 3.21 zeigt den Kopfzeilenbereich der Vorlage, und der ausnehmend pfiffige Titel »Beispiel-Blog« wird als Text angezeigt. Auf die Grafiken des Headers wird mit CSS als Hintergrundbilder referenziert. Es war wichtig, den Titel der Site als Text anzuzeigen, damit die Nutzer von Blogger das Template anwenden können, ohne selbst Grafiken erstellen zu müssen – ein echtes Plug-and-Play.

Abbildung 3.21 Beispiel der TicTac-Vorlage für Blogger, bei der die Wichtigkeit der Flexibilität der Schlüssel war, unterschiedlichen Text innerhalb des Headers aufzunehmen.

Für einzeilige Titel bei einer bestimmten Schriftgröße ist das Design des Headers ganz prima. Aber was ist, wenn der Titel ein wenig länger wird (Abbildung 3.22)? Wenn ich dem Header eine feste Höhe gegeben hätte, wären längere Titel (oder solche mit höherem Schriftgrad) in den Hintergrund gelaufen und unlesbar und hässlich geworden.

Abbildung 3.22 Wenn feste Höhen auf unterschiedliche Textmengen treffen, kann es zu unschönen Ergebnissen führen.

Also ist für das Design einer guten, wieder verwendbaren Vorlage wesentlich, dass sie sich an beliebige Inhaltsmengen anpassen kann. Damit der Header der Blogger-Vorlage sich nun stets passend ausdehnt, habe ich natürlich auf CSS und die schlaue Platzierung zweier Hintergrundbilder zurückgegriffen.

Das Markup

Styles und Grafiken habe ich erst bedacht, nachdem ich die Markup-Struktur für den Header erstellt hatte. Ich brauchte die Möglichkeit, auf zwei Hintergrundbilder zu referenzieren, und auch zwei Elemente im Markup, die ich mit den Bildern verknüpfen konnte:

```
<div id="blog-header">
  <h1>Beispiel-Blog</h1>
</div>
```

Wie Sie sehen können, habe ich mich im Titel des Weblogs für eine `<h1>`-Überschrift entschieden. Wenn Sie sich dieses Element lieber für andere Zwecke aufsparen wollen, können Sie hier auch gerne etwas anderes nutzen. Wichtig ist, dass Ihnen zwei Elemente zur Verfügung stehen. Ein `<div>`, das ein `<h1>` umschließt, funktioniert also sehr gut. Das `<div>` könnte als belanglos betrachtet werden, aber ich finde, es stört nicht.

Die Erstellung der beiden Bilder

Um den Ausdehnungseffekt zu erzielen, habe ich zwei Bilder erstellt. Eins davon habe ich größer gemacht, als es nach meiner Schätzung hätte sein müssen. Die Nutzer von Blogger werden sehen, dass abhängig von der Länge ihres Titels mehr oder weniger von diesem Bild zu sehen ist. Das zweite Bild schließt den unteren Rand des Headers ab. Dieses Bild wird unterhalb des Textes erscheinen, den der Nutzer in den Header stellt.

Die Abbildung 3.23 zeigt das große Bild, bei dem sich das Muster fast über die ganze Höhe wiederholt.

Abbildung 3.23 Das obere Bild habe ich viel höher als eigentlich nötig gemacht.

In der Abbildung 3.24 sehen wir das zweite Bild – der immer *unter* dem Titeltext positionierte Abschluss. Beachten Sie, dass der gesamte obere Bereich dieses Bildes transparent ist. Dadurch konnte ich diese Bilder aufeinander legen, wobei das top_div.gif durch das top_h1.gif zu sehen ist.

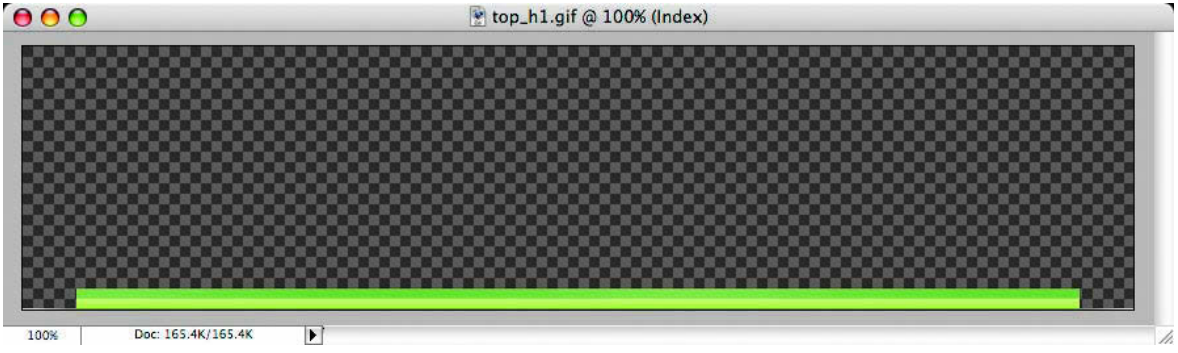

Abbildung 3.24 Die Grafik top_h1.gif enthält den unteren Abschluss des Headers mit einem transparenten Bereich, damit das andere Bild zu sehen ist.

Der Einsatz von CSS

Um nun alle Einzelteile zusammen zu setzen, habe ich zwei recht einfache CSS-Deklarationen benutzt. Zuerst habe ich die Regeln für das `<div>`-Element eingefügt:

```
#blog-header {
  margin: 0;
  padding: 0;
  font-family: "Lucida Grande", "Trebuchet MS";
  background: #e0e0e0 url(img/top_div.gif)
    no-repeat top left;
  }
```

Wie Sie sehen können, habe ich eine Standard-`font-family` eingefügt und das top_div.gif oben links auf einen hellgrauen Hintergrund gesetzt.

Als Nächstes habe ich die Deklaration für das Überschrift-Element ergänzt:

```
#blog-header {
  margin: 0;
  padding: 0;
  font-family: "Lucida Grande", "Trebuchet MS";
  background: #e0e0e0 url(img/top_div.gif)
    no-repeat top left;
  }
#blog-header h1 {
  margin: 0;
  padding: 45px 60px 50px 160px;
  font-size: 200%;
  color: #fff;
  background: url(img/top_h1.gif) no-repeat bottom left;
  }
```

Mit diesem CSS habe ich `font-size` und `padding` um den Text herum angepasst und top_h1.gif *unten* und links von der Überschrift positioniert. Weil `#blog-header` sich nur dem jeweiligen Inhalt entsprechend ausdehnt, wird hinter dem Überschrifttext gerade genug von seinem Hintergrund sichtbar.

Die Abbildung 3.25 zeigt, wie alle Teile zusammen wirken.

Abbildung 3.25 So passen alle Teile zusammen, und der obere Bereich des Headers dehnt sich nur soweit aus, wie die Inhalte des Titels es erfordern.

Wie ein Expander

Um den dehnbaren Header einem Test zu unterziehen, fügen wir mal einen langen Titel für die Site ein und schauen, was passiert. Wie in Abbildung 3.26 zu sehen ist, wird bei einem langen Titel mehr vom top_div.gif sichtbar, und top_h1.gif wird herunter geschoben und bleibt immer unter dem Titeltext.

Abbildung 3.26 Planung für unbekannte Inhaltsmengen im Voraus führt zum Erfolg.

Das Gleiche gilt auch umgekehrt. Wenn der Text nur eine Zeile füllt und der Besitzer der Site will, dass der Titel einen winzigen Text bekommt, wird der Header passend schrumpfen. Die Vorlage hat nun einen grafisch ergiebigen Header und ist doch bereit für das Unvorhergesehene.

Zusammenfassung

Sie kennen vielleicht den Ausdruck *Bäckerdutzend*? Ein kluger Bäcker wird beim Ofen beschicken immer einen Muffin oder ein Plätzchen extra zugeben. Falls ein Plätzchen verbrennt oder zerbricht (oder vom Bäcker verspeist wird), gibt es immer noch das Extrateil, damit das Dutzend *komplett* wird. Auf diese Weise plant der Bäcker Unvorhersehbares ein. Warum backen Sie nicht ebenso ein Plätzchen oder zwei extra, wenn Sie sich die Mühe machen, den Ofen anzuwerfen? Manchmal wird das zusätzliche Plätzchen gebraucht, manchmal nicht.

Legen wir die Plätzchen mal beiseite (aber natürlich noch in Reichweite): Wenn wir eine vertikale Ausdehnung von horizontalen Komponenten in unsere Designs einbauen, ist das, als ob wir uns ein eigenes Bäckerdutzend geben – wir planen für das Unerwartete und halten Platz frei für Textanpassungen und unterschiedliche Inhaltsmengen. Am Ende sparen wir Zeit und geben dem Nutzer (wie auch den Redakteuren der Site) eine bessere Kontrolle und erhöhte Accessibility.

Hier sind einige wichtige Punkte, die Sie sich merken sollten, wenn Sie horizontale Designkomponenten kreieren:

- Halten Sie das Markup von unnötigen Grafiken frei und verwenden Sie Hintergrundbilder innerhalb des CSS – das verringert aufgeblähten Code.
- Setzen Sie bei der Positionierung von Inhalt auf gegenüberliegenden Seiten eines Containers die Methode der »gegenüberliegenden Floats« ein.
- Wenn die Menge des Inhalts, der in einer Designkomponente platziert werden soll, unbekannt ist, benutzen Sie zwei Hintergrundbilder, damit diese Komponente sich ausdehnen und schrumpfen kann.
- Planen Sie mehr Platz ein, als Sie eigentlich zu brauchen meinen. Backen Sie dieses Schokoladenplätzchen extra.