O'REILLY

2. Auflage

Webdesign

mit CSS

Designer-Techniken für kreative
und moderne Webseiten

Jens Meiert & Ingo Helmdach

Webdesign mit CSS

Webdesign mit CSS

2. Auflage

Designer-Techniken für kreative
und moderne Webseiten

Jens Meiert & Ingo Helmdach

O'REILLY®

BEIJING • CAMBRIDGE • FARNHAM • KÖLN • PARIS • SEBASTOPOL • TAIPEI • TOKYO

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
Tel.: 0221/9731600
Fax: 0221/9731608
E-Mail: kommentar@oreilly.de

1. Auflage 2006
2. Auflage 2007

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.ddb.de* abrufbar.

Text und Webdesigns: Jens Meiert und Ingo Helmdach, mit Beiträgen von Gerrit van Aaken
Lektorat: Michael Gerth, Inken Kiupel, Köln
Korrektorat: Oliver Mosler, Köln
Fachgutachten: Jørgen W. Lang, Jens Grochtdreis, Moritz Sauer
Satz: Conrad Neumann, München
Umschlaggestaltung: Michael Henri Oreal, Köln
Produktion: Karin Driesen, Köln
Belichtung, Druck und buchbinderische Verarbeitung:
Media-Print, Paderborn

ISBN-13 978-3-89721-712-6

Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

Für meine Eltern, Hermann und Andrea Meiert.

Jens Meiert

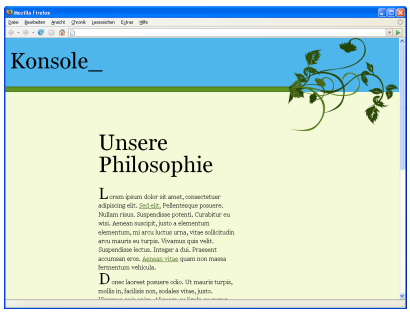
Inhalt

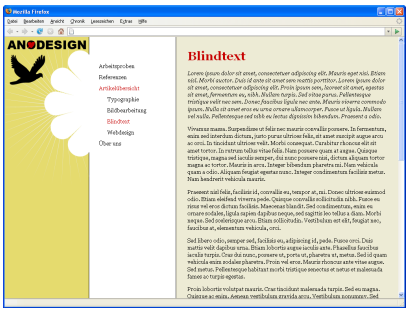
Teil III: Formulare und Tabellen

Teil IV: Effekte, Hingucker und Zaubereien

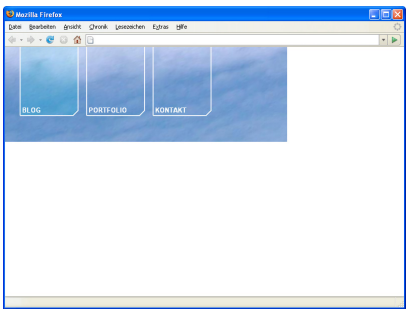
Designübersicht

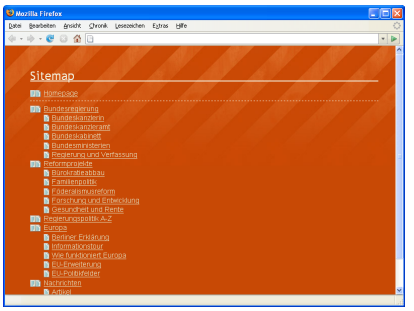
Teil I: Seitenvorlagen

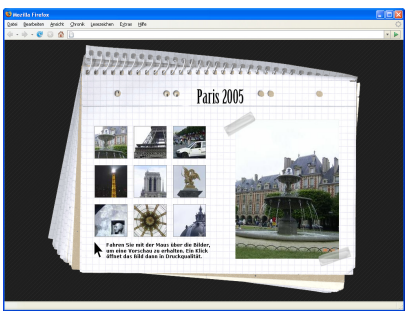


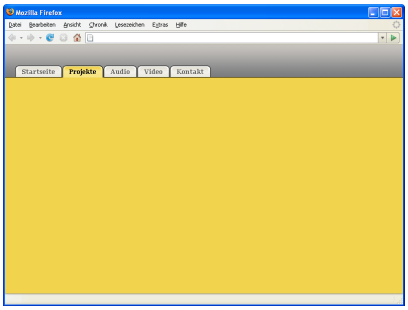


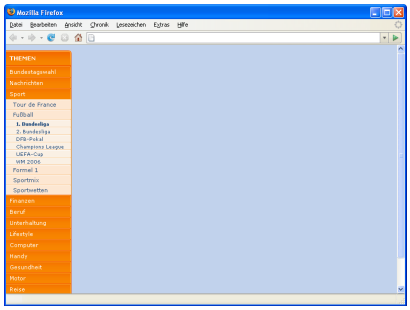
Teil II: Listen und Menüs

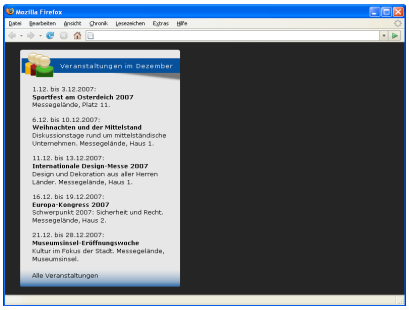




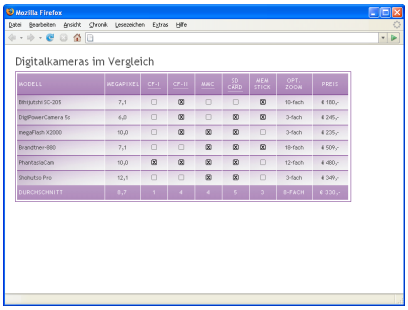


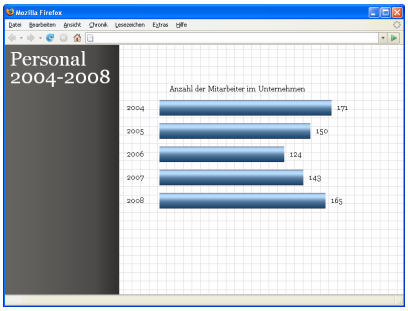


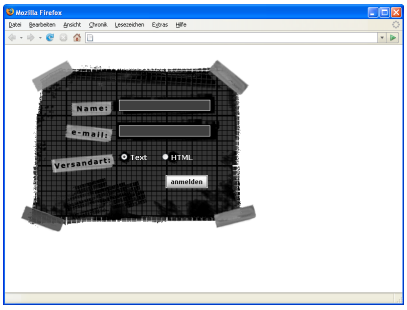


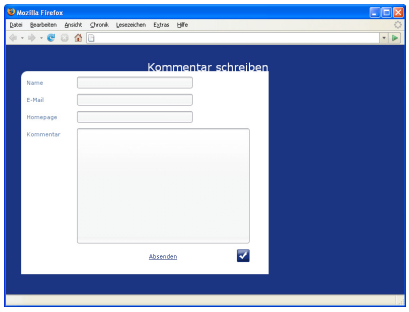


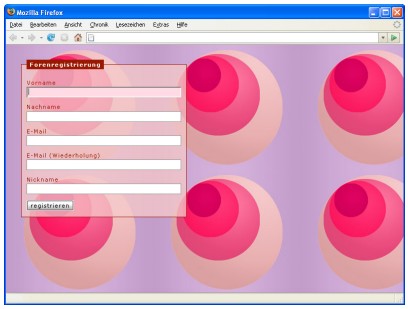
Teil III: Formulare und Tabellen

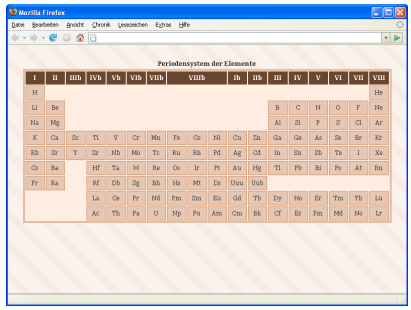


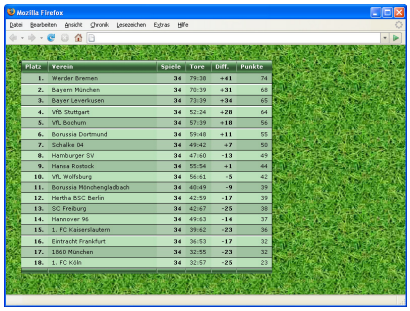




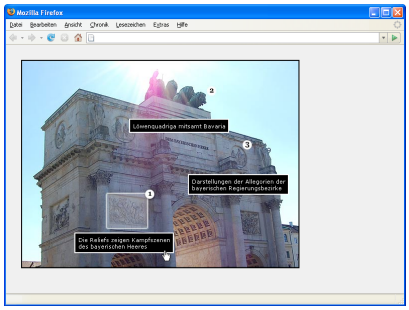


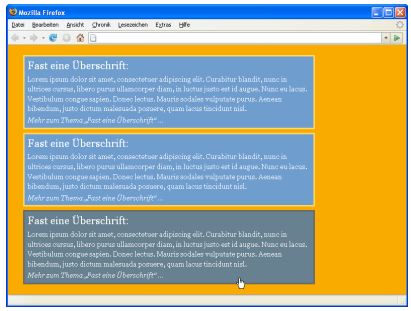


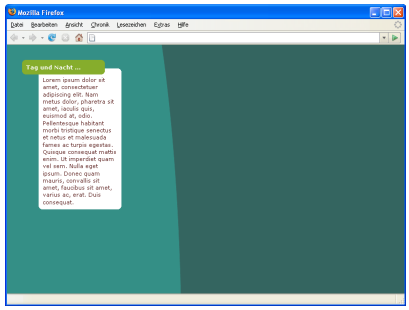


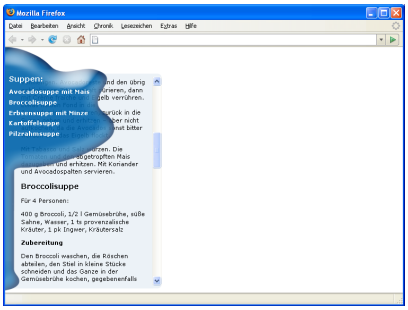


Teil IV: Effekte, Hingucker und Zaubereien

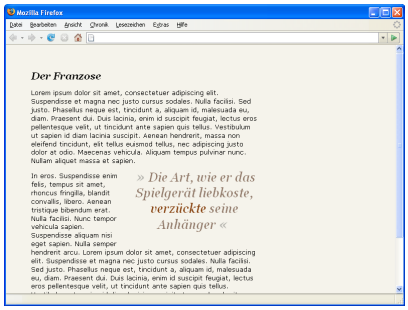




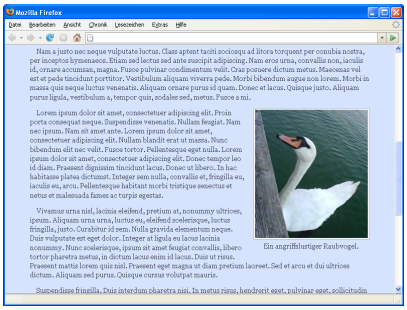




28. Mini-Kochbuch 215



29. Pullquote 223



30. Bildunterschriften 227

Seitenvorlagen

I

In diesem Teil geht es um vollständige Webseiten, deren Realisierung nicht nur von einer einzelnen Technik abhängt. Er soll einen ersten Eindruck von der Vielfalt an Wegen vermitteln, die CSS Webdesignern und Webentwicklern offeriert. Teil I bietet zudem Einblick in bestimmte Vorgehensweisen, vor allem in Bezug auf die Struktur einer Webseite und ihres HTML-Markups.

In diesem Teil

Fließlayout »Porto Seguro«

1

Urlaubszeit! Wir sind von unseren Ferien in Porto Seguro derart begeistert, dass wir dazu gleich eine kleine Website aufsetzen wollen. Unser Anspruch: ein voll flexibles, »fließendes« Layout.

Schwierigkeitsgrad: mittel

Abbildung 1-1: Fließlayout »Porto Seguro«

Abbildung 1-2: foto-1.jpg, 78 x 78 Pixel

Abbildung 1-3: foto-2.jpg, 78 x 78 Pixel

Abbildung 1-4: foto-3.jpg, 78 x 78 Pixel

Abbildung 1-5: foto-4.jpg, 78 x 78 Pixel

Abbildung 1-6: foto-5.jpg, 78 x 78 Pixel

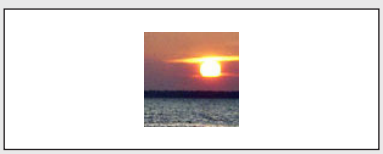


Abbildung 1-7: foto-6.jpg, 78 x 78 Pixel

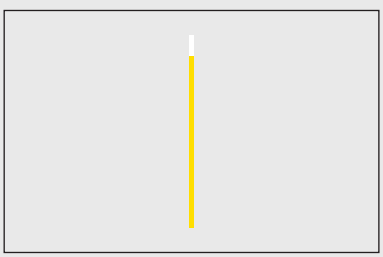


Abbildung 1-8: hintergrund-1.gif, 1 x 45 Pixel

Grundgerüst

Das HTML für unsere Website ergibt sich durch die Inhalte. Dazu gehören die »Seiten-Identität« sowie einen Inhaltsbereich mit »Geschichte«, »Brasilien erleben« und »Fotos«.

```
<h1>Porto <span>Seguro</span></h1>
<div id="inhalt">
  <div id="anreisser-1">
    <h2>Geschichte</h2>
    <p />
    <p />
    <a href="#">mehr zur Geschichte</a>
  </div>
  <div id="anreisser-2">
    <h2>Brasilien erleben</h2>
    <p />
    <p />
    <a href="#">mehr über Porto Seguro</a>
  </div>
  <div id="galerie">
    <h2>Fotos</h2>
    <ul>
      <li><a href="#"><img src="bilder/foto-1.jpg" alt="" /></a></li>
      <li><a href="#"><img src="bilder/foto-2.jpg" alt="" /></a></li>
      <li><a href="#"><img src="bilder/foto-3.jpg" alt="" /></a></li>
      <li><a href="#"><img src="bilder/foto-4.jpg" alt="" /></a></li>
      <li><a href="#"><img src="bilder/foto-5.jpg" alt="" /></a></li>
      <li><a href="#"><img src="bilder/foto-6.jpg" alt="" /></a></li>
    </ul>
  </div>
</div>
```

Was wir mit dem span-Element in der Überschrift vorhaben, erfahren Sie im nächsten Abschnitt.

Formatierung, 1. Akt

```
html {
background: #FFF500 url(bilder/hintergrund-1.gif) bottom repeat-x;
color: #000;
font: 75%/1.4 georgia, serif;
}

body {
background: url(bilder/hintergrund-2.gif) 0 5px repeat-x;
border-top: 40px solid #FF9800;
}
```

Über das `html`-Element definieren wir wie sonst auch Schrift und Schriftfarbe: 12 Pixel Georgia und Schwarz. Die Hintergrundzuweisungen lassen sich rasch erklären: Das `html`-Element erhält eine generelle Hintergrundfarbe mitsamt einem die Seite am Ende abschließenden Bild, das horizontal wiederholt wird. Unser Design sieht eine ähnliche Darstellung oben vor, nämlich ein ebenfalls horizontal wiederholtes Bild mit zwei »Balken«. Hier

möchten wir aber ein anderes Vorgehen demonstrieren und dieses über Rahmen realisieren. So wird der erste Balken über einen 40 Pixel starken Rahmen auf dem body realisiert, während 5 Pixel darunter ein horizontal wiederholter Verlauf folgt.

Die Fünf-Pixel-Aussparung benötigen wir für die h1-Überschrift:

```
h1 {
background: url(bilder/blickfang.gif) center no-repeat;
border-top: 5px solid #FFF;
}
```

Diese schließt nämlich die »Lücke« mit einem fünf Pixel dicken, weißen Rahmen nach oben und erfreut uns mit einem weiteren Hintergrundbild, das zentriert und nicht wiederholt wird. So haben wir nicht nur oben und unten jeweils zwei »Balken«, sondern auch gleich einen »Blickfang« eingebunden, der immer mittig über unserem via body eingebundenen Verlauf sitzt und bei jeder Fenstergröße passt.

Damit der Blickfang auch vollständig sichtbar ist und zudem der Text richtig sitzt, müssen wir uns der h1-Überschrift noch etwas intensiver widmen:

```
h1, h2 {
font-weight: normal;
}

h1 {
font-size: 4em;
height: 150px;
margin-bottom: 1.3em;
padding: 174px 30% 0 0;
text-align: right;
}
```

Zunächst soll die h1-Überschrift mitsamt den ebenfalls vorgesehenen h2-Überschriften nicht gefettet sein. Die Schriftgröße von 4em sorgt schließlich für genug Prominenz. Ein Außenabstand von 1.3em nach unten muss ebenfalls her, um etwas Luft zu schaffen.

Dann aber: Die Höhe würden wir über `height: 324px;` festlegen können, wir benötigen aber auch einen Hebel für die richtige Textpositionierung. Hierfür richten wir den Text erst mal rechts aus (`text-align: right;`) und »drücken« ihn über den Innenabstand 174 Pixel nach unten. 30% Innenabstand nach rechts sorgen dafür, dass der Inhalt der Überschrift mittig sitzt, aber auch je nach Fenstergröße »mitwandert«. Das funktioniert ganz gut, kann aber bei eher kleinen oder sehr großen Fenstergrößen befremdlicher wirken. Darüber sind wir uns im Klaren.

Um einen optischen Umbruch zwischen »Porto« und »Seguro« zu erzielen, haben wir mehrere Optionen: Die einfachste ist ein Zeilenumbruch via `<br />`, der bei uns aufgrund seiner schlechten Beeinflussbarkeit verpönt ist. Die nächste Möglichkeit ist ein weiterer Innenabstand nach links, der

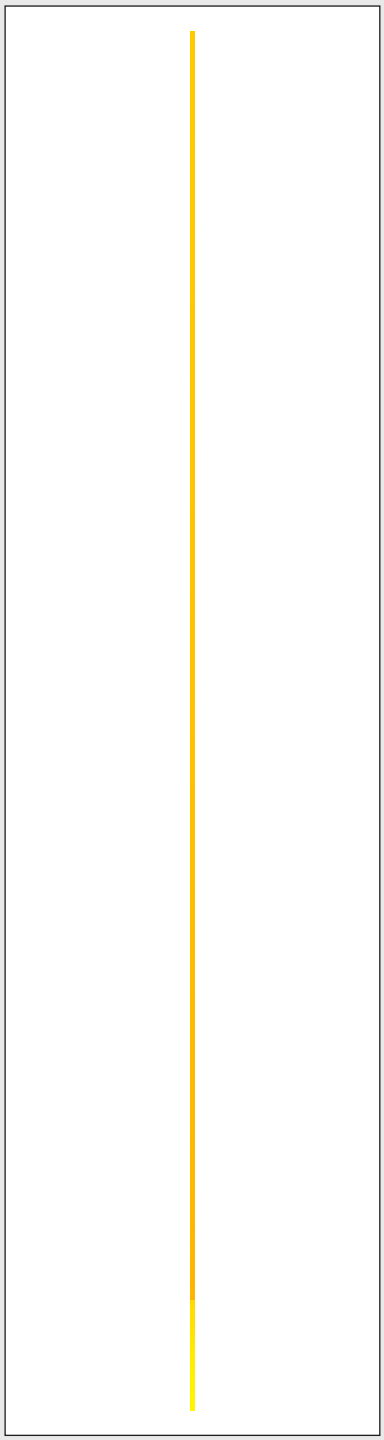


Abbildung 1-9: hintergrund-2.gif, 1 x 323 Pixel

Abbildung 1-10: blickfang.gif, 1.024 x 324 Pixel

so austariert ist, dass er dem Text zu wenig Platz gibt, um (bei möglichst vielen Fenstergrößen) in einer Zeile zu stehen. Wir könnten aber auch ein span-Element verwenden.

Letzteres ist unser Weg, wie auch das HTML andeutete, und diese Entscheidung wird noch durch einen anderen Umstand bedingt: Augenscheinlich stoßen wir auf nur schwer überbrückbare Probleme im Internet Explorer 7, wenn wir zudem die Zeilenhöhe (via `line-height`) auf unter `1.0` reduzieren wollen – was unser Plan war. So können wir via `display: block;` und `margin-top: -.75em;` eine robuste Lösung für beide Herausforderungen erzielen: zwei Zeilen, geringe Zeilenhöhe.

```
h1 span {
display: block;
margin-top: -.75em;
}
```

HINWEIS

Die genannten Werte klingen ja alle ganz nett, aber wir müssen dafür ab und an schon mal eine Weile rumprobieren. Und das tun wir in diesem sehr flexiblen Layout auf jeden Fall.

Formatierung, 2. Akt

Nachdem nun die grobe Seitenformatierung mitsamt »Key Visual« (Blickfang) steht, können wir uns der Strukturierung des Inhaltsbereichs widmen. Dafür greifen wir gerne auf die `float`-Eigenschaft zurück und »floaten« alle `div`-Elemente innerhalb des zunächst rein strukturell benötigten `inhalt`-Containers. Dabei legen wir gleichzeitig einen rechten Abstand von 5% fest.

```
div#inhalt div {
display: inline;
float: left;
margin-right: 5%;
}
```

HINWEIS

`display: inline;` *verdanken wir dem Internet Explorer 6 und seinem »Doubled Float-Margin Bug«. Dieser wird hier nur erwähnt, aber in den Referenzen nochmals aufgegriffen.*

Den einzelnen Abschnitten im Inhaltsbereich verpassen wir nun jeweils eine individuelle Breite, wobei das erste Element, `anreisser-1`, zudem einen

linken Abstand von 5% erhält, so dass die Abstände zu den Seiten immer 5% betragen:

```
div#anreisser-1 {
margin-left: 5%;
width: 18%;
}

div#anreisser-2 {
width: 32%;
}

div#galerie {
width: 29%;
}
```

Anschließend stellen wir zwei Probleme fest, die noch gelöst werden müssen: Zum einen »hängt« das `html`-Hintergrundbild im Opera-Browser knapp unter der `h1`-Überschrift, was wir dem Fließen der Elemente zu verdanken haben und mittels `overflow: hidden;` lösen können. Zum anderen haben wir in allen Browsern außer dem Internet Explorer 6 zuwenig Abstand nach unten, was wir über `padding-bottom` und in einer Kindselektor-Regel verpackt adressieren und somit an die Darstellung im IE 6 angleichen können:

```
* > div#inhalt {
overflow: hidden;
padding-bottom: 60px;
}
```

Formatierung, 3. Akt

Wir tasten uns weiter vor, nun mit dem Bedarf, ein paar allgemeine Zuweisungen zu treffen. Zwischenüberschriften, also `h2`-Überschriften, sollen eine definierte Größe und Zeilenhöhe haben, mitsamt etwas Abstand nach unten:

```
h2 {
font-size: 1.8em;
line-height: 1.2;
margin-bottom: .55em;
}
```

Ebenso wird der Ruf nach einer Überschriften- sowie Linkfarbe laut:

```
h2, a {
color: #02ACCA;
}
```

Absätze mögen Abstand:

```
p {
margin: 1em 0;
}
```

Abbildung 1-11: punkt.gif, 8 x 8 Pixel

Und auch die Links bedürfen noch unserer Aufmerksamkeit:

```
a {
background: url(bilder/punkt.gif) 0 .3em no-repeat;
padding-left: 12px;
text-decoration: none;
}

a:focus, a:hover, a:active {
text-decoration: underline;
}
```

Dem können wir mit einem Hintergrundbild, das wir über einen kleinen Innenabstand vernünftig platzieren, und Textunterstreichung nur bei Fokus, Hover und aktivem Zustand gerecht werden.

Formatierung, 4. Akt

Durch die einfache, da rein auf Inhalte bezogene Struktur der Container für »Geschichte« und »Brasilien erleben« profitieren diese von den zuvor vorgenommenen Definitionen. Der »Fotos«- oder »Galerie«-Bereich ist jedoch noch nicht abgeschlossen.

Da wir keinen Eindruck von regulären Listen erwecken wollen, neutralisieren wir diesen:

```
div#galerie ul {
list-style: none;
}
```

Unserem Ziel, eine kleine Nebeneinanderstellung diverser Foto-»Thumbnails« zu veröffentlichen, kommen wir näher, wenn wir die einzelnen `li`-Elemente fließen lassen. Dies werten wir gleichzeitig auf, indem wir sie mit jeweils einem kleinen Hintergrundbild versehen, das links neben und über dem jeweiligen Foto sichtbar ist – wenn wir über `padding` einen entsprechenden Innenabstand vorsehen. Mit ein wenig Außenabstand lockern wir auch gleich noch die ganze Komposition etwas auf.

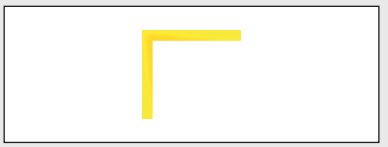


Abbildung 1-12: hintergrund-foto.gif, 82 x 73 Pixel

```
div#galerie li {
background: url(bilder/hintergrund-foto.gif) no-repeat;
float: left;
margin: 0 .4em .4em 0;
padding: 9px 0 0 9px;
}
```

Der im vorigen Akt definierte Hintergrund und Innenabstand für Links ist bei Verlinkung der Bilder unangenehmerweise auch in unserer Mini-Galerie sichtbar. Das haben wir jedoch schnell erledigt:

```
div#galerie a {
background: none;
padding: 0;
}
```

Zum Schluß versehen wir die Bilder mit einem 1 Pixel breiten, schwarzen Rahmen und bedenken noch alle Browser, die unsere `li`-Außenabstände wegen der Berücksichtigung der Text-Unterlänge nicht auf die gleiche Art interpretieren:

```
a img {
border: 1px solid #000;
display: block;
}
```

Fast schöner als der Urlaub, unsere »Porto Seguro«-Microsite.

Referenzen

- Porto Seguro

  http://www.portosegurotur.com.br

- Position Is Everything: »The IE Doubled Float-Margin Bug«

  http://www.positioniseverything.net/explorer/doubled-margin.html

- Eric Meyer: Bilder, Tabellen und mysteriöse Lücken

  http://devedge-temp.mozilla.org/viewsource/2002/img-table/

Fotogalerie

2

Vor uns haben wir ein wunderbares Layout, das eine Fotogalerie darstellt und das wir nicht nur als Screenshot ansehen wollen. Anhand der anstehenden Aufgaben, die uns über horizontale und vertikale Navigationslisten hin zu »fließenden« Inhalten führen, werden wir bei der Umsetzung eine Menge Spaß haben.

Schwierigkeitsgrad: schwer

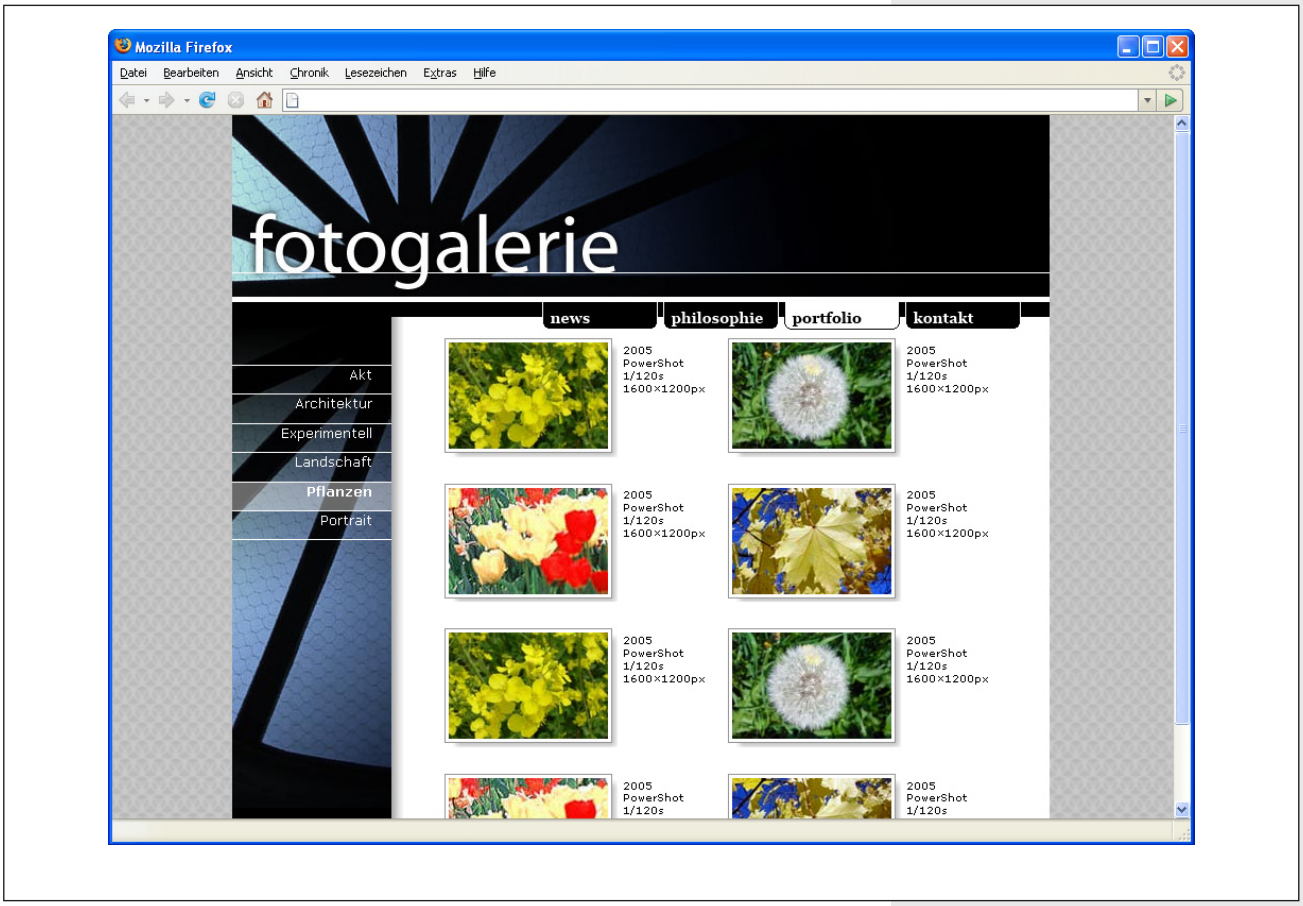


Abbildung 2-1: Fotogalerie

Grundlagen

Bevor wir überhaupt irgendetwas an HTML schreiben, setzen wir die beiden folgenden CSS-Regeln auf. Zum einen die in der Einleitung erwähnte »CSS-Zurücksetzung«:

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

Abbildung 2-2: hintergrund.gif, 16 x 16 Pixel

Zum anderen das Hintergrundbild der ganzen Webseite sowie die später am häufigsten verwendete Schrift (11 Pixel Verdana in Schwarz):

```
html {
background: url(bilder/hintergrund.gif);
color: #000;
font: 68.75%/1.2 verdana, arial, helvetica, sans-serif;
}
```

HINWEIS

Als effizienzbewusste Webentwickler setzen wir auch in diesem Buch oft auf die Kurzschreibweise von CSS-Eigenschaften: An Stelle von `font-size`, `line-height` *und* `font-family` *benutzen wir die Kurzschrifteigenschaft* `font`, *so wie in diesem Kapitel mit* `font: 68.75%/1.2 verdana, arial, helvetica, sans-serif;`.

Da wir hier eine ganze Webseite basteln, müssen wir gleich zu Beginn schon eine gewisse Struktur schaffen. Mit ein wenig exemplarischem Markup geht es also weiter, wobei wir uns hier nur mit Elementen innerhalb des `body` beschäftigen:

```
<div id="haupt">
  <h1><a href="#">Fotogalerie</a></h1>
  <a href="#inhalt" id="aux">Navigation überspringen</a>
  <div id="nav-1"></div>
  <div id="nav-2"></div>
  <div id="inhalt"></div>
  <div id="fusszeile"></div>
</div>
```

Dies sollte schon für sich sprechen, auch wenn wir festhalten wollen: Wir leiden nicht an »Div-itis« oder »Div-Ebola« (ein Begriff, den wir wohl Benjamin A. Howell zu verdanken haben). Einen Link »Navigation überspringen«, dem wir uns später noch näher widmen werden, haben wir bereits eingefügt. Und musterhaft ist bereits der Text »Fotogalerie« enthalten, den wir als Überschrift verwenden.

Kümmern wir uns erst mal um die Zentrierung der Seite:

```
div#haupt {
margin: auto;
width: 770px;
}
```

Erstes Leben hauchen wir unserer Seite durch einen vernünftigen Seitenkopf ein. Wie im obigen Codeschnipsel gezeigt, handelt es sich um eine `h1`-Überschrift, und wir setzen die Phark-Methode auf dem `a`-Element ein, um ein Bild anzuzeigen. In Kapitel 19, »Newsletter-Formular«, gehen wir näher auf »Image Replacement« à la Phark ein.

Abbildung 2-3: ueberschrift.jpg, 770 x 176 Pixel

```
h1 {
background: url(bilder/ueberschrift.jpg);
}

h1 a {
display: block;
height: 176px;
text-decoration: none;
text-indent: -999em;
width: 770px;
}
```

Prinzipiell hätten wir uns die erste der beiden Regeln sparen und die `background`-Deklaration ebenfalls unter `h1 a` einhängen können. Da wir jedoch beim Überfahren mit der Maus unschöne »Flackereffekte« im Internet Explorer beobachten würden, akzeptieren wir die zusätzliche Regel.

Bevor wir uns etwas Interessantes für den bislang einzigen wirklichen Textlink auf unserer Seite einfallen lassen, definieren wir, dass alle Links weiß sein sollen. Das übernimmt ein a-Selektor mit der Deklaration `color: #FFF;`.

Unsichtbares Hilfsmittel

»Navigation überspringen« ist unser nächstes Ziel. Wir verwenden diesen Link, um Menschen mit motorischen Einschränkungen (und auch einfache Tastaturnutzer) sowie Blinde zu unterstützen, und wenn wir das bedenken, ist klar, dass wir den Link nicht unbedingt anzeigen müssen. Wir gehen also so vor, dass der Link grundsätzlich über eine von WebAIM vorgestellte (und erwiesenermaßen zugängliche) Methode ausgeblendet, aber beim »Tabben« (beim Verwenden der Tabulator-Taste) angezeigt wird.

Das Ausblenden erledigen die folgenden Deklarationen:

```
a#aux {
height: 1px;
overflow: hidden;
position: absolute;
top: -999em;
width: 1px;
}
```

Diese Zuweisungen gewährleisten ein hohes Maß an Zugänglichkeit, da ein Ausblenden über `display` oder `visibility` in vielen Fällen Probleme beim Vorlesen durch Bildschirmlesesoftware verursacht. Der Link wird gewissermaßen »eingedampft« (auf 1 x 1 Pixel Größe, und über die `overflow`-Eigenschaft wird dies anschließend noch mal sichergestellt) und dann außerhalb des sichtbaren Bereichs positioniert.

Mit den Pseudoklassen `:focus` und `:active` ermöglichen wir das Einblenden, wenn das Element den Fokus erhält bzw. gerade verwendet wird. Und dazu geben wir noch eine Prise Styling:

```
a#aux:focus, a#aux:active {
background: #000;
border-bottom: 5px solid;
display: block;
height: auto;
padding: 1em 0;
text-align: center;
top: 0;
width: 770px;
}
```

Mit diesen Zeilen erreichen wir, dass bei Bedarf am Seitenanfang unser nützlicher Link »Navigation überspringen« erscheint, und zwar als prominenter Block, weiß auf schwarz, mit einem abgrenzenden Rahmen, wie wir ihn auch bei dem Hintergrundbild der Überschrift verwenden. (Ja, dort haben wir auf den Einsatz der `border`-Eigenschaft verzichtet.)

Navigation, Teil 1

Widmen wir uns nun der Topnavigation. In unserem oben vorgestellten Markup-Gerüst haben wir bereits einen Container für sie vorgesehen, `nav-1`. Unschwer zu erraten, was wir in ihm ablegen:

```
<div id="nav-1">
  <ul>
    <li id="news"><a href="#">News</a></li>
    <li id="philosophie"><a href="#">Philosophie</a></li>
    <li id="portfolio">Portfolio</li>
    <li id="kontakt"><a href="#">Kontakt</a></li>
  </ul>
</div>
```

Was haben wir vor? Wir brauchen vier nebeneinander stehende Reiter, die sich auf unserer Webseite eher rechts befinden. Der jeweils aktive Reiter soll hervorgehoben werden, analog auch jeder Reiter, wenn man mit der Maus über ihn fährt.

Wir beginnen mit ein paar generellen Modifikationen. Der Außencontainer erhält einen schwarzen Hintergrund und eine feste Höhe, zudem bietet er uns die Grundlage für die Positionierung unserer Liste:

```
div#nav-1 {
background: #000;
height: 14px;
position: relative;
}
```

Dann kommt die Liste dran. Sie wird an die richtige Stelle manövriert, indem wir sie positionieren. Das Ganze geschieht vor allem vor dem Hintergrund, dass der eigentliche Seiteninhalt später ja halbwegs »unter« der Navigation erscheinen soll, die Navigation den Inhaltsbereich also ein Stück weit überdeckt.

```
div#nav-1 ul {
left: 292px;
list-style: none;
position: absolute;
}
```

Kurzen Prozess machen wir auch mit den einzelnen Listenelementen, die wir »floaten«:

```
div#nav-1 li {
float: left;
height: 26px;
margin-right: 5px;
width: 109px;
}
```

Dann beginnt die Arbeit mit den einzusetzenden Bildern. Ähnlich wie beim `h1`-Element setzen wir den Hebel an den Kind-Ankern an und holen die Phark-Methode aus dem Werkzeugkoffer.

```
div#nav-1 li a {
display: block;
height: 100%;
text-decoration: none;
text-indent: -999em;
}
```

Die beiden letzteren Deklarationen werden wir später an eine andere Stelle im Stylesheet verschieben, um die Selektoren für `h1`-Anker und Topnavigationsanker zu gruppieren.

Es folgt eine ganze Reihe von Regeln für die jeweiligen Reiter sowie mögliche Zustandswechsel (Fokus, Hover und aktiv). Das Ganze sieht kompliziert aus, ist aber trivial:

```
div#nav-1 li#news a {
background: url(bilder/news.gif);
}

div#nav-1 li#news a:focus, div#nav-1 li#news a:hover, div#nav-1 li#news
a:active {
background: url(bilder/news-an.gif);
}
```

Abbildung 2-4: news.gif, 109 x 26 Pixel

Abbildung 2-5: news-an.gif, 109 x 26 Pixel

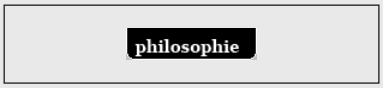


Abbildung 2-6: philosophie.gif, 109 x 26 Pixel

Abbildung 2-7: philosophie-an.gif, 109 x 26 Pixel

Abbildung 2-8: portfolio.gif, 109 x 26 Pixel

Abbildung 2-9: portfolio-an.gif, 109 x 26 Pixel

Abbildung 2-10: kontakt.gif, 109 x 26 Pixel

Abbildung 2-11: kontakt-an.gif, 109 x 26 Pixel

```
div#nav-1 li#philosophie a {
background: url(bilder/philosophie.gif);
}

div#nav-1 li#philosophie a:focus, div#nav-1 li#philosophie a:hover,
div#nav-1 li#philosophie a:active {
background: url(bilder/philosophie-an.gif);
}

div#nav-1 li#portfolio a {
background: url(bilder/portfolio.gif);
}

div#nav-1 li#portfolio a:focus, div#nav-1 li#portfolio a:hover,
div#nav-1 li#portfolio a:active {
background: url(bilder/portfolio-an.gif);
}

div#nav-1 li#kontakt a {
background: url(bilder/kontakt.gif);
}

div#nav-1 li#kontakt a:focus, div#nav-1 li#kontakt a:hover, div#nav-1
li#kontakt a:active {
background: url(bilder/kontakt-an.gif);
}
```

Was ist aber nun mit den aktiven Punkten, wo wie bei »Portfolio« gar kein Link existiert? Die gerade angelegten Regeln, die die an-Bilder definieren, ergänzen wir alle jeweils um einen Selektor, der das entsprechende li-Element wählt, also `div#nav-1 li#news`, `div#nav-1 li#philosophie`, `div#nav-1 li#portfolio` und `div#nav-1 li#kontakt`.

Damit der jeweilige Text nicht angezeigt wird, kommt erneut Phark zum Einsatz. Wir haben mittlerweile (und wie zuvor angedeutet) die Fälle, in denen wir so vorgehen (und in denen wir auch Ankern die Unterstreichung versagen), zusammengefasst, und die betreffende Regel sieht nun so aus:

```
h1 a, div#nav-1 li, div#nav-1 li a {
text-decoration: none;
text-indent: -999em;
}
```

Aktive Reiter werden hier nur dadurch – dafür aber gewissermaßen »automatisch« – gekennzeichnet, dass sie nicht verlinkt sind. Das entbindet uns von zusätzlichen Klassen oder abweichenden IDs, die wir bräuchten, wenn wir verlinkte Reiter als aktiv kennzeichnen müssten. Um Missverständnisse zu vermeiden: Wir verlinken diese Reiter nicht, weil man auf Seiten, auf denen man sich befindet, diese Seite selbst nicht verlinkt – wir tun das nicht etwa, um dadurch Code einzusparen. Dieses Usability-Prinzip beherzigen wir überall in diesem Buch.

Navigation, Teil 2

Wieder müssen wir uns erst einmal um etwas Markup kümmern. Noch eine Liste:

```
<div id="nav-2">
  <ul>
    <li><a href="#">Akt</a></li>
    <li><a href="#">Architektur</a></li>
    <li><a href="#">Experimentell</a></li>
    <li><a href="#">Landschaft</a></li>
    <li><span>Pflanzen</span></li>
    <li><a href="#">Portrait</a></li>
  </ul>
</div>
```

Dann geben wir der Navigation wieder einen Hintergrund und Dimensionen. Die Dimensionen benötigen wir, wenn der Hintergrund vollständig angezeigt werden soll. Die Verwendung der Eigenschaften `min-height` und `min-width` aus CSS 2 wäre fein, diese Eigenschaften werden aber nicht von allen aktuellen Browsern unterstützt. Wir geben also alles »fest« in Pixeln an. Die linke Navigation sowie den Inhalt lassen wir »fließen«. Und damit unsere linke Spalte auch schön weiterläuft, bekommt unser `haupt`-Container ein Hintergrundbild, das sich nur vertikal wiederholt.

```
div#nav-2 {
background: url(bilder/nav-hintergrund.jpg);
float: left;
height: 462px;
width: 150px;
}

div#haupt {
background: #FFF url(bilder/nav-schatten.gif) repeat-y;
}
```

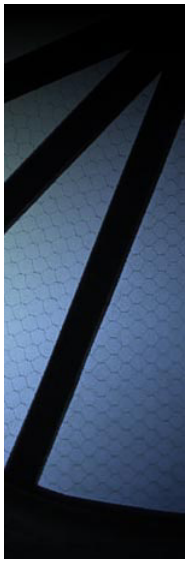


Abbildung 2-12: nav-hintergrund.jpg, 150 x 462 Pixel

Abbildung 2-13: nav-schatten.gif, 161 x 1 Pixel

Mit einem Rahmen nach unten sorgen wir dafür, dass alle Listenpunkte nach oben und unten abschließen. Mehr als einen Rahmen nach oben müssen wir den `li`-Elementen nun nicht mehr zuweisen. Den Listenstil setzen wir komplett zurück (das sollten wir mit der Topnavigation zusammenlegen und damit vereinfachen können). »Positioniert« wird die Liste, indem wir einen entsprechenden Außenabstand setzen.

```
div#nav-2 ul {
border-bottom: 1px solid #FFF;
list-style: none;
margin-top: 45px;
}
```

Was gilt nun gleichzeitig für »aktive« Listenelemente wie auch »inaktive« Punkte? Weiße Schrift und eine Ausrichtung nach rechts:

```
div#nav-2 li, div#nav-2 li a {
color: #FFF;
text-align: right;
}
```

Es folgen die Deklarationen, die wir speziell für die Formatierung unserer Listenpunkte benötigen:

```
div#nav-2 li {
border-top: 1px solid;
font-size: 1.2em;
height: 2em;
width: 150px;
}
```

Die in ihnen enthaltenen Anker sowie span-Elemente (unser aktiver Punkt, und nun kommt Licht ins Dunkel) erhalten entsprechend modifizierende Zuweisungen:

```
div#nav-2 li a, div#nav-2 li span {
display: block;
font-weight: normal;
height: 100%;
padding-right: 18px;
text-decoration: none;
}
```

Diese dienen eigentlich nur dem Zweck, den ganzen Listenpunkt auszufüllen (`display: block;` und `height: 100%;`), die Schriftstärke zu reduzieren (`font-weight: normal;`), eine Linkunterstreichung zu vermeiden (`text-decoration: none;`) – sofern vorhanden – und nach rechts den von uns gewünschten Innenabstand zu erzielen.

Aktive Punkte – durch span gekennzeichnet – sollen fett hervorgehoben werden, ebenso wie »gehoverte« Listenpunkte.

```
div#nav-2 li a:hover, div#nav-2 li span {
font-weight: bold;
}
```

Aufgrund unserer guten Vorbereitung sind wir schon fast fertig. Als Letztes werfen wir unser durchsichtiges PNG in die Waagschale:

```
div#nav-2 li a:hover, div#nav-2 li span {
background: url(bilder/nav-highlight.png);
}
```

Abbildung 2-14: nav-highlight.png, 1 x 1 Pixel (hier vergrößert dargestellt)

Auf diese Weise erzielen wir ein ansprechendes Highlighting sowohl des aktuellen Punkts wie auch der mit der Maus überfahrenen Punkte.

Das Ganze funktioniert allerdings nicht im Internet Explorer 6. Das nervt uns wirklich, aber wir sind pragmatisch: In unserer Truhe liegen noch ein paar Kindselektoren herum, die der Internet Explorer 6 auch nicht versteht. Also schrauben wir Folgendes zusammen:

```
div#nav-2 li > a:hover, div#nav-2 li > span {
background: url(bilder/nav-highlight.png);
}
```

Damit sind wir bezüglich der linken Navigation aus dem Schneider. Im Internet Explorer 6 werden aktive und »gehoverte« Punkte fett darge-

stellt, und in seinem Nachfolger sowie in Browsern, die zum Beispiel auf der KHTML-Engine (Safari, Konqueror) oder auf der Gecko-Engine (Firefox, Mozilla, Netscape) basieren, wird zusätzlich noch unser PNG als Hintergrund angezeigt.

Zu guter Letzt noch eine Quizfrage: Was benötigen wir, damit wir auch beim »Durchtabben« und bei der aktiven Linkwahl eine Hervorhebung erzielen? Die Pseudoklassen `:focus` und `:active`. Dies kann bei Gecko-Browsern unter Linux teilweise einen Darstellungsfehler beim »Hovern« der Navigation hervorrufen, aber wir schreiben sie erst mal dazu.

Inhalt

Weiter zu dem, was unsere Musterseite überhaupt ausmacht: die Inhalte. Als Erstes müssen wir den Inhaltscontainer an seinen Zielort verfrachten. Da wir bereits die linke Navigation »floaten«, gehen wir hier analog vor:

```
div#inhalt {
float: left;
padding: 10px 0 15px 55px;
width: 565px;
}
```

Justierende Deklarationen, die den in Kürze vorgestellten Kindelementen dienen, haben wir auch an Bord.

Momentan sind die beiden gefloateten Container höher als der Hauptcontainer, ragen also über diesen hinaus. Somit reichen sowohl die Hintergrundfarbe als auch das Hintergrundbild nicht mehr über die volle Höhe. Wir müssen einen genauen Blick in die Spezifikation werfen bzw. auf Anne van Kesteren hören, worauf wir in Kapitel 20, »Kommentar-Formular«, noch etwas näher eingehen werden:

```
div#haupt {
overflow: hidden;
}
```

Dann liegt das Markup an, von dem wir nicht viel verlangen:

```
<div id="inhalt">
  <div class="bild"><a href="#"><img src="bilder/blume-1.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&amp;times;1200px</div>
  <div class="bild"><a href="#"><img src="bilder/blume-2.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&amp;times;1200px</div>
  <div class="bild"><a href="#"><img src="bilder/blume-3.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&amp;times;1200px</div>
  <div class="bild"><a href="#"><img src="bilder/blume-4.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&amp;times;1200px</div>
</div>
```

Abbildung 2-15: blume-1.jpg, 150 x 100 Pixel

Abbildung 2-16: blume-2.jpg, 150 x 100 Pixel

Abbildung 2-17: blume-3.jpg, 150 x 100 Pixel

Abbildung 2-18: blume-4.jpg, 150 x 100 Pixel

Interessanter ist, was wir aus dem bisschen Markup machen. Wir wollen je zwei Bilder pro Zeile darstellen, und rechts sollen jeweils noch ein paar Daten zum Bild erscheinen. Auch hier ist es naheliegend, die Container fließen zu lassen. Durch den Zugriff auf die `bild`-Klasse können wir Ausnahmen bezüglich der Breite der Container definieren, schließlich benötigen wir bei den Bildern etwas mehr Platz.

```
div#inhalt div {
float: left;
margin: 15px 20px 15px 0;
width: 8.5em;
}

div#inhalt div.bild {
margin-right: 5px;
width: 150px;
}
```

Nebenbei bemerkt ist es interessant, dass wir den Text ganz ohne Markup wie `br`-Elemente oder Ähnliches formatieren können. (Das wird zwar bei einer größeren Textskalierung nicht mehr funktionieren, aber das ist hier nicht von Bedeutung.)

Wenn wir uns dem beschreibenden Text widmen, fühlen wir uns auch nicht mehr gefordert, ganz im Gegenteil – unsere Regel `div#inhalt div` erweitern wir um `font-size: .9em;`.

Beim Schatten und der Umrahmung der Galeriebilder wird es wieder interessanter. Wir weisen den Bildcontainern ein Hintergrundbild zu:

Abbildung 2-19: schatten.gif, 158 x 108 Pixel

```
div#inhalt div.bild {
background: url(bilder/schatten.gif) bottom right no-repeat;
}
```

Anschließend brauchen wir eine neue Breite für diesen Container, denn sein Inhalt, also jedes Bild, wird nun mit einem Rahmen sowie einem Innenabstand versehen, den wir durch eine Hintergrundfarbe weiß gestalten.

```
div#inhalt div.bild {
width: 158px;
}

div#inhalt div.bild img {
background: #FFF;
border: 1px solid #979797;
padding: 3px;
}
```

Der neuen Höhe, die wir durch Innenabstand und Rahmen erhalten, sowie einem sonst auftretenden Problem mit Opera tragen wir zwischen den Zeilen Rechnung, indem wir unseren fließenden `div`-Elementen `height: 10.8em;` zuweisen.

Unsere Bilderzuweisungen ergänzen wir nun um die folgenden drei Deklarationen:

```
div#inhalt div.bild img {
left: -5px;
position: relative;
top: -5px;
}
```

Diese Deklarationen kombinieren wir natürlich mit den anderen, bereits definierten Bild-Deklarationen. Das Bild wird nun um jeweils fünf Pixel nach links und nach oben verschoben, wodurch unser Schatten freigelegt wird.

Finale

Zu guter Letzt kommt noch die Fußzeile an die Reihe. Wir haben anfangs bereits einen Container dafür vorgesehen, eigentlich muss dort nur noch etwas Text hinein:

```
<div id="fusszeile">Copyright 2005 <a href="#">Max Mustermann</a></div>
```

Außerdem haben wir zu Anfang einen Link »Navigation überspringen« kreiert und mit einer entsprechenden Formatierung versehen. Das dort verwendete Layout wollen wir auch in der Fußzeile gebrauchen. Da keine der für `a#aux:focus` und `a#aux:active` definierten Deklarationen mit unserem Vorhaben kollidiert, können wir der Regel einfach den Selektor `div#fusszeile` hinzufügen.

Zwei Dinge sind noch zu tun: den Fluss der beiden vorangegangenen Elemente (linke Navigation und Inhaltsbereich) festlegen und dafür sorgen, dass die ganze Schrift in Weiß dargestellt wird.

```
div#fusszeile {
clear: left;
color: #FFF;
}
```

Referenzen

- WebAIM: Unsichtbare Inhalte für Screenreader-Nutzer

  http://www.webaim.org/techniques/css/invisiblecontent/

- CSS-Spezifikation: `min-width` und `max-width`

  http://www.w3.org/TR/CSS21/visudet.html#min-max-widths

- CSS-Spezifikation: `min-height` und `max-height`

  http://www.w3.org/TR/CSS21/visudet.html#min-max-heights

- CSS-Spezifikation: Kindselektoren

  http://www.w3.org/TR/CSS21/selector.html#child-selectors

Index