Fotogalerie

1

Schwierigkeitsgrad: schwer

Vor uns haben wir ein wunderbares Layout, das eine Fotogalerie darstellt und das wir nicht mehr lediglich 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.

Abbildung 1-1: Fotogalerie

Grundlagen

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

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

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

Abbildung 1-2: hintergrund.gif, 16 x 16 Pixel

```
html {
background: url(bilder/hintergrund.gif);
color: #000;
font: 70%/1.2 verdana, arial, helvetica, sans-serif;
}
```

Da wir hier eine ganze Webseite basteln, müssen wir gleich am Anfang 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="#">Fotografie</a></h1>
  <a href="#inhalt" id="aux">Navigation &uuml;berspringen</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 »Fotografie« enthalten, den wir als Überschrift verwenden.

Kümmern wir uns erst mal um die Zentrierung der Seite:

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

Leider reicht das wegen dem Internet Explorer 5.x nicht aus. Wir benötigen noch folgende zwei Regeln:

```
body {
text-align: center;
}

div#haupt {
text-align: left;
}
```

Erstes Leben hauchen wir unserer Seite durch einen vernünftigen Seitenkopf ein. Wie im obigen Codeschnipsel demonstriert, handelt es sich um eine `h1`-Überschrift, und wir setzen die Phark-Methode auf dem `a`-Element ein, um ein Bild anzuzeigen. In Kapitel 16, »Newsletter-Formular«, wird übrigens näher auf »Image Replacement« à la Phark eingegangen.

Abbildung 1-3: 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 zweite 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, 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.

Abbildung 1-4: news.gif, 109 x 26 Pixel

Abbildung 1-5: news-an.gif, 109 x 26 Pixel

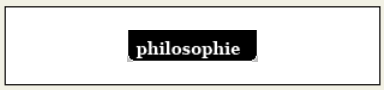

Abbildung 1-6: philosophie.gif, 109 x 26 Pixel

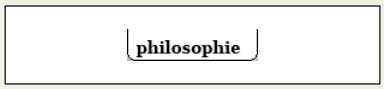

Abbildung 1-7: philosophie-an.gif, 109 x 26 Pixel

Abbildung 1-8: portfolio.gif, 109 x 26 Pixel

Abbildung 1-9: portfolio-an.gif, 109 x 26 Pixel

Abbildung 1-10: kontakt.gif, 109 x 26 Pixel

Abbildung 1-11: kontakt-an.gif, 109 x 26 Pixel

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);
}

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 just 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 – dabei aber gewissermaßen »automatisch« – gekennzeichnet, dass sie nicht verlinkt sind. Dies 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 dies 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;
}
```

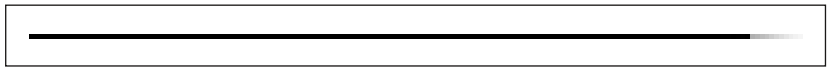

Abbildung 1-13: nav-schatten.gif, 161 x 1 Pixel

Abbildung 1-12: nav-hintergrund.jpg, 150 x 462 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: 27px;
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);
}
```

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

Das Ganze funktioniert allerdings nicht im Internet Explorer. Das nervt uns wirklich, aber wir sind pragmatisch: In unserer Truhe liegen noch ein paar Kindselektoren herum, die der Internet Explorer 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);
}
```

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

Damit sind wir bezüglich der linken Navigation aus dem Schneider. Im Internet Explorer sind aktive und »gehoverte« Punkte gefettet, und in – durchaus nicht nur brandneuen – 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 die linke Navigation bereits »floaten«, gehen wir hier analog vor:

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

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

Momentan sind beide gefloatete Container höher als der Hauptcontainer, ragen also über diesen hinaus. Somit reichen sowohl Hintergrundfarbe als auch Hintergrundbild nicht mehr über die volle Höhe. Wir müssen einen genauen Blick in die Spezifikation werfen bzw. auf Anne van Kesteren

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

Abbildung 1-16: blume-2.jpg, 150 x 100 Pixel

Abbildung 1-17: blume-3.jpg, 150 x 100 Pixel

Abbildung 1-18: blume-4.jpg, 150 x 100 Pixel

hören, worauf wir in Kapitel 17, »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&times;1200px</div>
  <div class="bild"><a href="#"><img src="bilder/blume-2.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&times;1200px</div>
  <div class="bild"><a href="#"><img src="bilder/blume-3.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&times;1200px</div>
  <div class="bild"><a href="#"><img src="bilder/blume-4.jpg"
  alt="Blume" /></a></div>
  <div>2005 PowerShot 1/120s 1600&times;1200px</div>
</div>
```

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 nahe liegend, die Container fließen zu lassen. Durch Zugriff auf die `bild`-Klasse können wir Ausnahmen bei 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: 85px;
}

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:

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

Abbildung 1-19: schatten.gif, 158 x 108 Pixel

Anschließend brauchen wir eine neue Breite dieses Containers, denn sein Inhalt, also jedes Bild, wird nun mit einem Rahmen sowie einem Innenabstand versehen, den wir durch eine Hintergrundfarbe weiß gestalten. Dies funktioniert leider nicht so recht im Internet Explorer 5.5, aber mit ein, zwei Kniffen ist unser Design auch dort flottgemacht.

```
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: 108px;` 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 nach links und oben um jeweils fünf Pixel verschoben, wodurch unser Schatten freigelegt wird.

Bonus

Jetzt wird es nochmal spannend. Wir bauen in unsere Seite einen kleinen »Effekt« ein, nämlich einen Inhaltsbereich, der scrollbar und gewissermaßen ein CSS-Frame ist und über den die Topnavigation – es wurde bereits angedeutet – »rüberlappt«.

Unbedenklich ist das nicht: Dieses Vorgehen wird die Benutzerfreundlichkeit der Seite stark einschränken, denn wir zeigen auf vermutlich jeder Plattform entweder zu viel (wo dann zweimal gescrollt werden muss, und zwar regulär sowie im Inhaltscontainer) oder zu wenig Inhalt (weil wir diesen »einfach so« abschneiden). Sie sollten dies also nicht unbedingt nachahmen.

Durch ein Fehlverhalten des Internet Explorer in Bezug auf die vorher relativ positionierten Bilder sind wir jedoch gezwungen, das Vorgehen bereits im Vorfeld durch eine »MOS«-Erweiterung zu entschärfen (»MOS« steht für Mozilla, Opera, Safari):

```
div#haupt > div#inhalt {
height: 437px;
overflow: auto;
}
```

Die Höhe orientiert sich hier an der von uns festgelegten Mindesthöhe für die linke Navigation.

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 ja 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, nämlich linke Navigation und Inhaltsbereich, mit der `clear`-Eigenschaft festlegen und dafür sorgen, dass die ganze Schrift in Weiß dargestellt wird.

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

Referenzen

- Paul Bohman (WebAIM): Eine zugängliche Methode, um HTML-Inhalte zu verstecken

  http://www.webaim.org/techniques/articles/hiddentext

- 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