Fließlayout »World of Fish«

4

Schwierigkeitsgrad: mittel

»World of Fish« stellt ein kleines Informationsangebot zu einem begehbaren Aquarium dar, und dieses Angebot soll von uns umgesetzt werden. Im Vorfeld haben wir wie immer ein Konzept erstellt, die Inhalte gesichtet und zurechtgelegt, alles mit den Beteiligten abgestimmt und uns dann alle benötigten Bilder zurechtgeschnitten.

Abbildung 4-1: Fließlayout »World of Fish«

Fangen wir mit Hintergrund und Seitenüberschrift an:

```
html {
background: #4F7EC9;
color: #FFF;
font: 70%/1.2 verdana, arial, helvetica, sans-serif;
}

h1 {
background: url(bilder/world-of-fish.gif) top center no-repeat;
height: 168px;
text-indent: -999em;
}
```

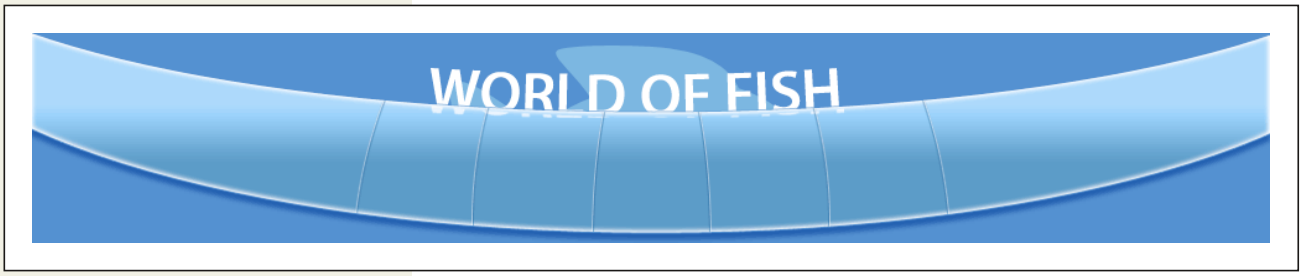

Abbildung 4-2: world-of-fish.gif, 1000 x 168 Pixel

Was machen wir hier mit unserer Überschrift, und wie sieht deren Markup überhaupt aus?

```
<h1 xml:lang="en">World of Fish</h1>
```

Auch wenn wir eine deutschsprachige Seite umsetzen, verwenden wir hier englischen Text, und der muss entsprechend gekennzeichnet werden. Wie im Vorwort bereits angesprochen, verwenden wir XHTML 1.0. Damit ist uns das `lang`-Attribut erlaubt, aber wir gebrauchen es nur im `xml`-Namensraum, um einen problemlosen Wechsel auf XHTML 1.1 zu ermöglichen.

Die Formatierung der Überschrift muss dem Umstand Rechnung tragen, dass wir ein sehr großes Hintergrundbild verwenden, aber auch bei kleineren Auflösungen gut dastehen wollen. Dies erledigen wir, indem wir dem `h1`-Element keine feste (also die volle) Breite vorgeben und ihm ein zentriertes Hintergrundbild zuweisen. So wird bei kleineren Auflösungen alles Relevante zu sehen sein, ohne dass Scrollbalken entstehen.

Navigation

Etwas anders wird es nun bei der Navigation und den Inhalten ablaufen.

```
<div id="nav">
  <ul>
    <li id="tour"><a href="#">Tour</a></li>
    <li id="angebot"><a href="#">Angebot</a></li>
    <li id="ueber"><a href="#">&Uuml;ber uns</a></li>
    <li id="wissen"><a href="#">Wissen</a></li>
```

```
    <li id="kontakt"><a href="#">Kontakt</a></li>
  </ul>
</div>
```

Die Navigation erhält eine Breite von 761 Pixeln, so dass wir hier erst mal Hand anlegen und den nav-Container zentrieren müssen. Durch den Internet Explorer müssen wir den üblichen Workaround schaffen, da wir nicht ohne weiteres nur `margin: auto;` heranziehen können:

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

div#nav {
margin: auto;
text-align: left;
width: 761px;
}
```

Anschließend widmen wir uns der Liste:

```
div#nav ul {
list-style: none;
width: 470px;
}

div#nav li {
float: left;
width: 94px;
}
```

Bevor wir die Liste in Position bringen, kümmern wir uns um die darin enthaltenen Links. Wir weisen ihnen Maße zu, nehmen die Unterstreichung weg, schaffen ihren Text aus dem Weg (wir werden auch hier die von uns favorisierte Phark-Methode einsetzen) und machen ein Block-Element aus ihnen.

```
div#nav li a {
background-repeat: no-repeat;
display: block;
height: 88px;
text-decoration: none;
text-indent: -999em;
}
```

HINWEIS

Das Geheimnis der Deklaration `background-repeat: no-repeat;` *wird in Kürze gelüftet.*

Die Liste schieben wir an die richtige Stelle, indem wir dem Elternelement (`div#nav`) die Deklaration `position: relative;` zuweisen und dann absolut dazu die Liste positionieren.

```
div#nav {
position: relative;
}

div#nav ul {
left: 146px;
position: absolute;
top: -84px;
}
```

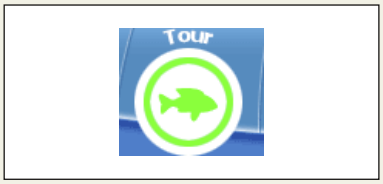

Abbildung 4-3: nav-tour.gif, 94 x 83 Pixel

Abbildung 4-4: nav-angebot.gif, 94 x 88 Pixel

Abbildung 4-5: nav-ueber.gif, 94 x 86 Pixel

Abbildung 4-6: nav-wissen.gif, 94 x 85 Pixel

Abbildung 4-7: nav-kontakt.gif, 94 x 85 Pixel

Im nächsten Schritt folgen die Zuweisungen für die einzelnen Navigationsbilder. Um die Bilder bündig zum vom Überschriftenhintergrund beschriebenen Bogen zu machen und die unterschiedliche Höhe der Bilder auszugleichen, justieren wir mit der margin-top-Eigenschaft nach.

```
div#nav li#tour a {
background-image: url(bilder/nav-tour.gif);
margin-top: 2px;
}

div#nav li#angebot a {
background-image: url(bilder/nav-angebot.gif);
margin-top: 7px;
}

div#nav li#ueber a {
background-image: url(bilder/nav-ueber.gif);
margin-top: 16px;
}

div#nav li#wissen a {
background-image: url(bilder/nav-wissen.gif);
margin-top: 10px;
}

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

Und damit ist auch klar, warum wir zuvor ein no-repeat für den Linkhintergrund gesetzt haben.

Haken und Ösen

Haken und Ösen gibt es auch in diesem Design, diesmal im wahrsten Sinne des Wortes. Den am Haken zappelnden »Jetzt NEU in München«-Schriftzug wollen wir an den von der Überschrift gebildeten Bogen heften. Und da wir die gleich im Anschluss folgende »Tagline« über die ganze Breite verwenden, aber einen festen Punkt zur Positionierung benötigen, ergänzen wir den nav-Container:

```
<div id="nav">
   ...
   <div>Jetzt NEU in M&uuml;nchen</div>
</div>
```

Wir setzen erneut die bewährte Phark-Bildersatztechnik ein, um den verwendeten Text durch unser Bild zu ersetzen.

```
div#nav div {
background: url(bilder/neu.gif);
height: 141px;
left: 655px;
position: absolute;
text-indent: -999em;
```

```
width: 105px;
top: -30px;
}
```

Bevor das untergeht, vereinfachen wir jetzt ein wenig den Code, da wir doch die text-indent-Eigenschaft schon zum wiederholten Male gebrauchen (bislang an drei Stellen):

```
h1, div#nav * {
text-indent: -999em;
}
```

Das ist doch gleich platzsparender. Wir machen uns dabei den Umstand zunutze, dass jeglicher in Block-Elementen – nur hierauf können wir text-indent anwenden – stehender Text im Navigationscontainer ausgeblendet bzw. ersetzt werden soll. Ein Fall für den universellen Selektor *, den wir im Navigationskontext (div#nav) einsetzen.

Zur Integration der Tagline erstellen wir einen entsprechend simplen Hilfscontainer – da sie semantisch keinen wirklichen Absatz darstellt, verwenden wir ein div-Element:

```
<div id="tagline">Das begehbare Aquarium in M&uuml;nchen</div>
```

Der bereits justierte »NEU«-Haken dient nun als gute Orientierung, um den tagline-Container über die margin-top-Eigenschaft ebenfalls zurechtzurücken. Für das gleichzeitig zu setzende Hintergrundbild definieren wir zudem die entsprechende Höhe:

```
div#tagline {
background: url(bilder/tagline-hintergrund.gif);
height: 36px;
margin-top: 47px;
}
```

Nun ist es einfach, die Schrift ebenfalls an unsere Bedürfnisse anzupassen. Um sie vertikal auszurichten, ziehen wir die line-height-Eigenschaft heran. Das ist in erster Linie einfacher, als einen Innenabstand zu spezifizieren.

```
div#tagline {
font-size: 1.2em;
font-weight: bold;
line-height: 1.8em;
}
```

Spannender wird es wieder bei dem Wellenbild, das ebenfalls noch in die Tagline soll. Wir könnten einfach ein img-Element heranziehen, aber da dem Bild keine Bedeutung zukommt, scheidet diese Variante aus. Eine andere Möglichkeit wäre es, ebenfalls von der Navigation aus einen Container »herabzuseilen«, der ein Hintergrundbild mitbringt, wobei gleichzeitig ein rechter Innenabstand im tagline-Container gesetzt wird. Oder wir spielen ein wenig mit einem span-Element und hängen hinter den Tagline-Text einfach .

Abbildung 4-8: neu.gif, 105 x 141 Pixel

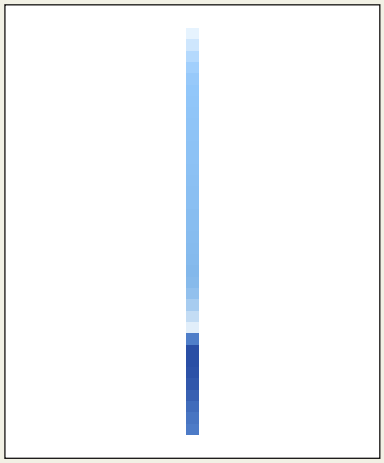

Abbildung 4-9: tagline-hintergrund.gif, 1 x 36 Pixel (hier vergrößert dargestellt)

Abbildung 4-10: tagline-aqua.gif, 51 x 17 Pixel

```
div#tagline span {
background: url(bilder/tagline-aqua.gif) top right no-repeat;
display: block;
height: 17px;
width: 51px;
}
```

Auf diese Weise erhalten wir zunächst ein 51 x 17 Pixel großes Behältnis für unser Bild. Als Block-Element »rutscht« es unter die Tagline, was uns nicht gefällt. Den Hebel setzen wir bei einem entsprechenden Außenabstand an, der gleichzeitig die Zentrierung sowie das »Nach-oben-Holen« durch einen negativen Wert für `margin-top` übernimmt. Über einen Innenabstand geben wir dem sonst überlappten Text den nötigen Raum.

```
div#tagline span {
margin: -1.4em auto 0;
padding-left: 340px;
}
```

Das war es aber noch nicht ganz: Damit der um das Bild ergänzte Tagline-Inhalt wirklich zentriert wirkt, muss der Außencontainer etwas Innenabstand erhalten, der die Bildbreite sowie etwas Abstand umfasst:

```
div#tagline {
padding-right: 60px;
}
```

Inhalt

Für den Inhalt der Webseite bzw. ihrer Homepage sind drei Bereiche vorgesehen, die gleich breit und hoch sein sowie nebeneinander stehen sollen.

```
<div id="inhalt">
  <div>
    <h2>Er&ouml;ffnungsangebot</h2>
    <p>Lorem ipsum [...]</p>
  </div>
  <div id="mitte">
    <h2>Wissenswertes</h2>
    <p>Lorem ipsum [...]</p>
  </div>
  <div>
    <h2>Fanartikel</h2>
    <p>Lorem ipsum [...]</p>
  </div>
</div>
```

Um diese Art der Visualisierung zu erreichen, lassen wir die Container fließen. Beginnen wir mit ein paar generellen Regeln.

```
div#inhalt {
margin: auto;
padding: 25px 0;
text-align: left;
width: 645px;
}
```

```
div#inhalt div {
float: left;
width: 195px;
}
```

Damit zentrieren wir den Inhaltsbereich und heben die dem `body`-Element zugewiesene Textzentrierung mittels `text-align` auf. Da die Deklarationen der Navigation teilweise identisch sind, fassen wir diese noch zusammen. (Die bei der Navigation angegebene Breite würde aber dann durch die `div#inhalt`-Regel »überschrieben« werden, während die relative Positionierung hier harmlos ist). Die umschlossenen div-Elemente stellen wir über die `float`-Eigenschaft nebeneinander.

Wenn es um die Überschriften geht, können wir uns ebenfalls »Synergien« zunutze machen, da die Schrift der Tagline auf dieselbe Art formatiert wurde. Wir fassen also auch hier unsere CSS-Zuweisungen zusammen, sobald wir die von uns gewünschte Darstellung erreicht haben:

```
h2 {
background: url(bilder/ueberschrift.gif);
font-size: 1.2em;
font-weight: bold;
height: 36px;
line-height: 1.8em;
padding-right: 7px;
text-align: right;
}
```

Abbildung 4-11: ueberschrift.gif, 195 x 36 Pixel

Der Umstand, dass die einzelnen Abschnitte im Inhalt jeweils nur einen Absatz umfassen, versetzt uns in die glückliche Lage, nicht noch ein gesondertes Hilfskonstrukt zusammenschrauben zu müssen, sondern einfach den `p`-Elementselektor nutzen zu können. Über diesen Selektor weisen wir den Absätzen eine Hintergrundfarbe, einen Rahmen (jedoch nicht nach oben), den gewünschten Innenabstand (nach oben ebenfalls modifiziert) sowie eine feste Höhe zu (sollten die Inhalte unterschiedlich lang sein, wären die Blöcke auch unterschiedlich hoch).

```
p {
background: #74A9DC;
border: 1px solid #EEF5FB;
border-top: 0;
height: 100px;
padding: 7px 14px 14px;
}
```

Fehlt also nur noch eins: Die drei Kästen brauchen etwas Abstand zueinander.

```
div#inhalt div#mitte {
margin: 0 30px;
}
```

HINWEIS

Wie bei der Tagline ist es auch hier legitim, den Farbkontrast zu hinterfragen. Während wir diese Webseite aber einfach als Test ansehen, würden wir diese Problematik in einer echten Version natürlich berücksichtigen.

Referenzen

- CSS-Spezifikation: Die `text-indent`-Eigenschaft

  http://www.w3.org/TR/CSS21/text.html#indentation-prop