Florence Maurice

CSS3

Leitfaden für Webdesigner

dpi

ADDISON-WESLEY

3

Transparenzen – opacity, RGBA und Co.

Elemente halb transparent zu setzen ist eine beliebte Strategie, die Designs vielschichtig wirken lässt. Klassischerweise hat man das über halb transparente PNG-Grafiken realisiert oder mit nicht transparenten Bildern getrickst. Mit CSS3 kommen native und einfach einzusetzende Transparenzen: über die Eigenschaft `opacity` oder die Farbangaben mit `rgba()` und `hsla()`.

3.1 Klassisches

In CSS 2.1 stand keine native Möglichkeit zur Verfügung, um Bereiche mehr oder weniger als transparent zu definieren. Es gibt aber Methoden, um Transparenzen zu simulieren.

3.1.1 Simulation über zwei Hintergrundbilder

Abbildung 3.1 zeigt einen typischen Einsatz von transparenten Bereichen. Ein Text wird auf ein Bild gesetzt, aber zur besseren Lesbarkeit mit einem halb transparenten Hintergrund versehen.

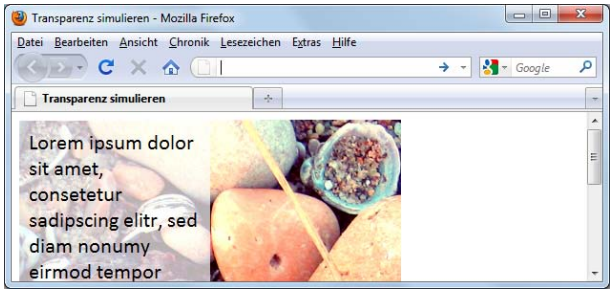


Abbildung 3.1
Der Text befindet sich auf einem teilweise transparenten Hintergrund

Eine Möglichkeit, so etwas zu simulieren, führt über den Einsatz von zwei Hintergrundbildern, von denen das eine normal ist und das andere transparenter aussieht (Abbildung 3.2). Es ist jedoch nicht wirklich transparent, sondern ein opakes aufgehelltes Bild.

Abbildung 3.2
Zwei Bilder, um Transparenzen zu simulieren

Im HTML-Code gibt es einen Wrapper und einen Absatz mit dem eigentlichen Textinhalt.

```
<div id="wrapper">
  <p>Lorem ipsum </p>
</div>
```

Per CSS wird jetzt dem umfassenden Container das opake Bild zugewiesen. Der darin stehende Absatz erhält eine Breite und Höhe und das halb transparente Bild:

```
#wrapper {
  width: 400px; /* Breite des Hintergrundbilds */
  height: 381px; /* Höhe des Hintergrundbilds */
  background-image: url(steine.jpg);
}
p {
  background: url(steine_halbtransparent.jpg) top left;
  width: 180px;
  height: 361px;
  margin: 0;
  padding: 10px;
}
```

Listing 3.1

Transparenzen mit zwei Hintergrundbildern simulieren (*transparenz_simulieren.html*)

Diese Lösung ist mit einem sehr großen Aufwand verbunden:

- Für jede Beschriftung auf einem halb transparenten Bereich muss ein zusätzliches Bild erstellt werden.
- Soll der Bereich etwas weniger transparent und etwas opaker sein, muss ein erneutes Bild erstellt werden.
- Auch die Performance leidet, da zwei Bilder für diesen Effekt geladen werden müssen. Diesen Punkt könnte man durch den Einsatz von Sprites verbessern.

Sprites sind eine CSS-Technik, bei der mehrere Bilder zu einem zusammengefasst werden. `background-position` sorgt dann dafür, dass immer der richtige Ausschnitt des großen Hintergrundbildes und damit das richtige Bild angezeigt wird. Mehr zu Sprites unter `http://css-tricks.com/css-sprites/`.

Etwas weniger Aufwand stellt eine zweite Methode dar: der Einsatz eines halb transparenten PNG als Hintergrundbild. Der Code ist derselbe wie im Beispiel zuvor, aber dieses Mal erhält der Absatz das halb transparente PNG-Bild:

```
p {
  background-image: url(trans.png);
}
```

Listing 3.2

Transparenzen simulieren über transparente PNGs (*transparenz_png.html*)

Diese Methode hat einige Vorteile, aber auch Nachteile:

- Ein Vorteil ist, dass **ein PNG-Bild** für beliebig viele Beschriftungen benutzt werden kann.
- Aber natürlich bedeutet auch dieses PNG-Bild einen zusätzlichen HTTP-Request.
- Eine gewünschte Änderung am Grad der Transparenz bedeutet wiederum, dass das Bild verändert werden muss.

Der IE6 kann alphatransparente Bilder nicht korrekt darstellen. Falls man da noch nachbessern wollte (aber der IE6 ist schon wirklich alt!), gibt es unterschiedliche Möglichkeiten wie das Format PNG8 (`http://blogs.sitepoint.com/2007/09/18/png8-the-clear-winner/`) oder aber den AlphaImageLoader-Filter (`http://24ways.org/2007/supersleight-transparent-png-in-ie6`).

3.2 Opacity – weniger ist mehr Transparenz

In CSS3 gibt es gleich mehrere Möglichkeiten, Transparenzen zu definieren. Das zeigt, dass die Entwickler von CSS3 anerkennen, dass Transparenz ein wichtiges Designanliegen ist.

Beginnen wir mit der CSS3-Eigenschaft `opacity`, mit der Sie die Transparenz von Elementen bestimmen können. Opazität heißt so viel wie Deckkraft und ist das Gegenteil von Transparenz.

3.2.1 So funktioniert opacity

Statt `opacity: 0.5` können Sie auch `opacity: .5` schreiben.

Der Standardwert ist `opacity: 1`, d. h. Elemente sind ganz deckend. Ein geringerer Wert als 1 macht ein Element durchsichtiger, `opacity: 0.5` ist folglich halb durchsichtig und `opacity: 0` vollständig durchsichtig.

Das folgende Beispiel führt ein paar Werte vor. Im HTML-Teil sind mehrere `div`-Elemente mit eindeutigen `id`s platziert:

```
<div id="eins">0</div>
<div id="zwei">0.2</div>
<div id="drei">0.4</div>
<div id="vier">0.6</div>
<div id="fuenf">0.8</div>
<div id="sechs">1</div>
```

Im Dokument wird eine Hintergrundgrafik eingebunden. Die einzelnen `div`-Elemente erhalten eine weiße Hintergrundfarbe, Breite sowie Höhe und werden nebeneinander gefloatet. Das Entscheidende: Für jedes Element wird ein anderer Grad an Opazität festgelegt:

LISTING 3.3

Ausschnitt aus dem Listing *opacity_prinzipiell.html*

```
#eins { opacity: 0; }
#zwei { opacity: 0.2; }
#drei { opacity: 0.4; }
#vier { opacity: 0.6; }
#fuenf { opacity: 0.8; }
#sechs { opacity: 1; }
```

Die Auswirkung demonstriert die folgende Abbildung.

Abbildung 3.3

Elemente mit einem unterschiedlichen Grad an Opazität

Das Element mit `opacity: 0` ist ganz transparent, damit unsichtbar, das Element mit `opacity: 1` ist vollständig opak, d. h. deckend. Die anderen zeigen Zwischenstufen. Eine Besonderheit sieht man am Text des Elements mit `opacity: 0.2` oder auch mit `0.4`:

Abbildung 3.4

Die Beschriftung der mehr oder minder transparenten Bereiche

Es sieht dann aus, als hätten die Texte selbst leicht unterschiedliche Farben, aber ein genauerer Blick zeigt, dass auch **durch den Text der Hintergrund durchschimmert**. Die Texte haben also den gleichen Grad an Transparenz wie die Elemente, in denen sie stehen.

Im Beispiel stehen die Texte direkt innerhalb der `div`-Elemente. Ein zusätzliches `p`-Element würde nichts daran ändern, dass die Texte transparenter sind. Auch würde es nichts helfen, bei `p` eine höhere Deckkraft anzugeben.

Diese Besonderheit an `opacity` ermöglicht beispielsweise das – etwa skriptgesteuerte oder über Transitions (*Kapitel 9*) bedingte – Ausblenden von Elementen durch ein Verringern der Deckkraft.

In manchen Fällen wie bei einer Bildbeschriftung auf einem halb transparenten Bereich ist es störend, dass die Transparenz vererbt wird, weil die Schrift weniger gut lesbar ist. Dann ist es besser, auf `rgba()` oder `hsla()` zurückzugreifen.

Das ist ein wichtiger Punkt bei `opacity`: Alle Elemente, die innerhalb des über `opacity`-modifizierten Elements stehen, erben den Grad der Transparenz. Ein Text innerhalb eines Elements mit `opacity: 0.2` ist ebenfalls nur zu 20 % sichtbar.

Es gibt einen Trick, um einen nicht transparenten Text auf ein mit `opacity` teiltransparent gesetztes Element zu platzieren. Dafür darf der Absatz mit dem Text kein Kindelement des Elements sein, das über `opacity` modifiziert wird. Die HTML-Struktur kann beispielsweise so aussehen:

```
<div class="wrapper">
  <p class="transparent"></p>
  <p class="opak">Text</p>
</div>
```

Dann müssen Text und transparenter Container noch positioniert werden, damit der Text sich auf dem halb transparenten Bereich befindet. Genaueres zeigt Dirk Jesse in einem Artikel unter `http://www.highresolution.info/weblog/entry/transparente_hintergruende_mit_css/`.

3.2.2 opacity in den heutigen Browsern

Der Firefox versteht `opacity` seit Version 0.9, nur für Versionen davor würden Sie `-moz-opacity` brauchen.

`opacity` wird von allen neuen Browsern unterstützt, d. h. Firefox, Safari, Chrome, Opera und Internet Explorer ab Version 9.

Was aber macht man mit dem Internet Explorer vor Version 9, der `opacity` noch nicht interpretiert? Abhilfe verspricht der Alpha-Filter.

```
filter: progid:DXImageTransform.Microsoft.Alpha(Opacity=50);
```

Bei diesem schreiben Sie den Grad der Opazität/Transparenz als Wert zwischen 0 und 100, wobei wiederum 0 ganz durchsichtig ist, 100 ganz sichtbar. Es gibt zusätzlich die Variante mit dem herstellerspezifischen Präfix `-ms-` und dem Wert in Anführungszeichen für den Internet Explorer 8 im Standardmodus:

```
-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";
```

Eine browserübergreifende Transparenz von 50 % lässt sich damit folgendermaßen realisieren – Sie sollten aber in Erwägung ziehen, alle Sonderangaben für den Internet Explorer über konditionale Kommentare auszulagern.

```
.transparent {
  filter: progid:DXImageTransform.Microsoft.Alpha(Opacity=50);
  -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=50)";
  opacity: 0.5;
}
```

Denken Sie aber daran, dass der Internet Explorer bis einschließlich Version 7 beim entsprechenden Element hasLayout benötigt, wofür im Zweifelsfall `zoom: 1` sorgt (Genaueres zu den Besonderheiten beim Einsatz von Filtern siehe *Kapitel 2*).

Alternativ dazu können Sie die Bibliothek Modernizr (siehe *Kapitel 2*) nutzen, um besondere Formatierungen für den Fall bereitzustellen, dass ein Browser `opacity` nicht versteht. Wenn `opacity` unterstützt wird, ergänzt Modernizr im `html`-Start-Tag `opacity` als Klasse, ansonsten `no-opacity`. Damit können Sie eine alternative Formatierung bei Nichtunterstützung bereitstellen:

```
.no-opacity .transparent {
  /* Alternative Formatierungen */
}
```

Wie die alternative Formatierung aussehen kann, hängt vom Kontext ab. Je nach Aufgabe könnte das ein halb transparentes PNG-Bild sein oder eine geänderte Hintergrundfarbe etc.

3.3 Farben notieren: rgb() und hsl()

Bevor es wieder um Transparenzen geht, zuerst allgemein zur Art, in CSS Farben anzugeben.

3.3.1 RGB und Co.

Meist werden Farben in CSS über hexadezimale RGB-Werte angegeben, also beispielsweise:

```
background-color: #ff0000;
```

Dabei bestimmen die ersten beiden Stellen den Rotwert, die zweiten beiden den Blauwert und die letzten beiden den Grünwert (Rot-Grün-Blau: RGB). Da die Werte im letzten Beispiel immer aus gleichen Zahlen/Buchstaben bestehen, ließe sich das verkürzen:

```
background-color: #f00;
```

Neben dieser Art, Farben anzugeben, kann man in CSS 2.1 die Farbangaben hinter das Schlüsselwort `rgb()` schreiben und zwar als durch Komma getrennte Dezimalwerte. Der höchstmögliche Wert ist dabei `255` und entspricht dem hexadezimalen `ff`:

```
background-color: rgb(255, 0, 0);
```

Anstelle von dezimalen Werten können Sie auch Prozentwerte nutzen:

```
background-color: rgb(100%, 0%, 0%);
```

Die folgenden Angaben bezeichnen alle einen roten Hintergrund:

```
background-color: #ff0000;
background-color: #f00;
background-color: rgb(255, 0, 0);
background-color: rgb(100%, 0%, 0%);
```

Diese Farbangaben sind nicht sehr intuitiv. Ich kenne niemanden, der sofort beim Anblick von `#f13dbe` sagen kann, um welche Farbe es sich handelt. (Es ist übrigens ein strahlendes Pink.)

3.3.2 HSL

Besser sind da die in CSS3 eingeführten Farbangaben mit **HSL**. HSL steht dabei für *hue* (Farbton), *saturation* (Sättigung) und *lightness* (Helligkeit).

So definieren Sie einen roten Hintergrund:

```
background-color: hsl(0, 100%, 50%);
```

Der Farbton hat den Wert 0, die Sättigung ist 100 % bei einer Helligkeit von 50 %.

Farbtöne wählen

Erst einmal zu den Farbtönen. Hue repräsentiert einen Winkel des Farbkreises.

Abbildung 3.5

Farbkreis, der als Basis für den Hue-Wert dient – Bild aus Wikipedia (`http://de.wikipedia.org/w/index.php?title=Datei:Color_circle_%28hue-sat%29.png`)

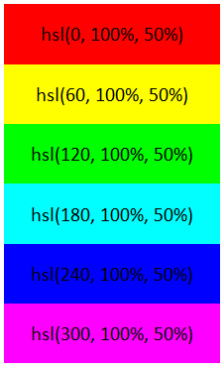
Rot ist ganz oben und entspricht dem Wert 0 oder 360. Weitere Entsprechungen sind:

- 0 red – rot
- 60 yellow – gelb
- 120 green – grün
- 180 cyan – aquamarin
- 240 blue – blau
- 300 magenta – magenta

Merken können Sie sich die Reihenfolge der englischen Farbtöne red – yellow – green – cyan – blue – magenta durch Sprüche wie „**R**ote **Y**acht **g**rüßt **C**abrio **b**ei **M**ondschein“ oder ähnlich.

Abbildung 3.6

Die Hauptfarbtöne mit ihren Werten



Zwischentöne wählen Sie durch Zwischenwerte. Orange beispielsweise liegt zwischen Rot und Gelb und hat entsprechend den Wert 30.

Für einen Farbton zwischen Grün und Hellblau kann man alle Werte zwischen 120 und 180 nehmen, beispielsweise 130, 140, 150 usw. Selbstverständlich sind nicht nur Zehnerschritte möglich.

Abbildung 3.7
Ein paar Zwischentöne zwischen Grün und Hellblau

Das ist schon einmal praktisch: Wenn bei einer Layoutpräsentation der Kunde sagt, er hätte gerne an dieser Stelle einen Grünton, der etwas mehr ins Bläuliche geht, so lässt sich das direkt realisieren, was mit den klassischen RGB-Werten kaum möglich wäre.

Sättigung

Kommen wir zur nächsten Angabe – der Sättigung. Das ist der zweite Wert, der hinter `hsl` angegeben wird, und den Sie als **Prozentzahl** notieren. Bisher wurde in den Beispielen immer von 100 % Sättigung ausgegangen, wodurch man die kräftigstmögliche Farbe erreicht. Durch niedrigere Prozentwerten sind Pastelltöne möglich.

Abbildung 3.8
Pastelltöne durch Anpassungen bei der Sättigung (von links nach rechts 0 %, 25 %, 50 % bis 100 %)

Bei 0 % Sättigung erhält man einen Grauton.

Helligkeit

Der letzte Faktor, den Sie beeinflussen können, ist die Helligkeit der Farbe. Bisher wurde hier immer 50 % gewählt, was einer normalen Helligkeit entspricht. Wählen Sie stattdessen 0 %, so haben Sie Schwarz, wählen Sie eine Helligkeit von 100 %, haben Sie Weiß. Dazwischen sind alle denkbaren Abstufungen möglich.

Abbildung 3.9
Abstufungen durch Veränderung der Helligkeit (links nach rechts 10 %, 25 %, 50 %, 75 %)

Und jetzt alle zusammen

Das Zusammenspiel der drei Faktoren veranschaulichen die folgenden Tabellen aus der W3C-Spezifikation.

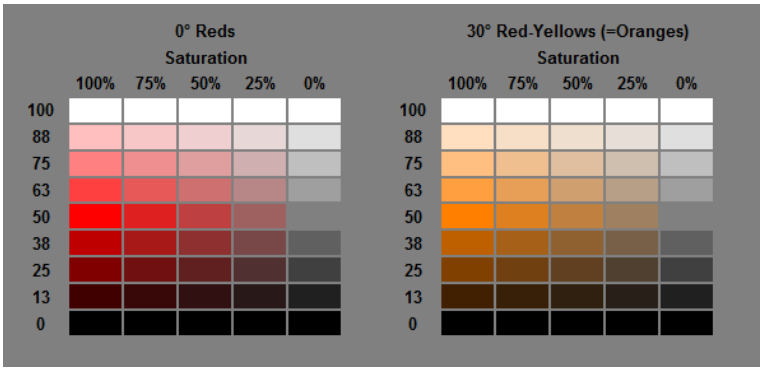


Abbildung 3.10

Zusammenspiel der HSL-Werte (`http://www.w3.org/TR/2010/PR-css3-color-20101028/#hsl-color`)

Diese Tabellen demonstrieren die Auswirkung von ausgewählten Sättigungs- und Helligkeitswerten bei **Rot und Orange**.

Die erste Tabelle zeigt die Farbe Rot, die ja den Farbton 0 hat. Die horizontale Beschriftung beschreibt den Grad der Sättigung. 100 %, d. h. ganz links, haben die Farben mit dem größten Grad an Intensivität, der nach rechts abnimmt.

Die vertikale Achse führt unterschiedliche Grade an Helligkeit vor. Es beginnt oben mit 100 % Helligkeit – ganz Weiß –, führt über mehrere Rosatöne bis zu 50 % Helligkeit (Normalwert) und wird nach unten immer dunkler bis zum Wert 0 %. Keine Helligkeit bedeutet Schwarz.

Genauso ist die rechte Tabelle mit den Werten zu Orange aufgebaut. Orange hat den Farbton 30, es liegt genau zwischen Rot (Farbton 0) und Gelb (Farbton 60).

Beim Betrachten der beiden Tabellen wird deutlich, dass das Ergebnis beim größten Grad an Helligkeit unabhängig vom Farbton immer Weiß ist, beim geringstmöglichen Grad an Helligkeit immer Schwarz. Genauso sind die Ergebnisse unabhängig vom Farbton immer Grau, wenn die Sättigung 0 % ist.

Auch wenn die Arbeit mit HSL zu Beginn ein bisschen Eingewöhnung braucht, so wird doch klar, dass sie auf Dauer wesentlich intuitiver ist und sich auch leichter Farbtonanpassungen direkt durchführen lassen.

Nützlich für die Arbeit mit HSL

Vielleicht kennen Sie die Firefox-Erweiterung Colorzilla (`https://addons.mozilla.org/en-us/firefox/addon/colorzilla/`). Einmal installiert, integriert sie eine kleine Pipette in der Statusleiste, über die man direkt die Farben von Webseitenelementen auswählen kann.

Ein rechter Mausklick auf das Pipettensymbol ermöglicht, direkt den Farbwert als HSL-Wert zu kopieren.

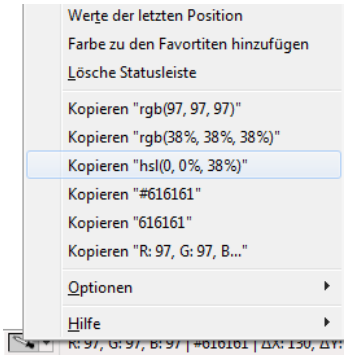


Abbildung 3.11
Über Colorzilla kann man den ausgewählten Wert auch als HSL kopieren.

Ein Doppelklick auf das Pipettensymbol öffnet einen Farbwähler, der einem ebenfalls die HSL-Werte angibt.

Sie sehen, HSL ist eine schöne intuitive Art, Farben zu bestimmen. Sie ahnen aber wahrscheinlich, dass dies nur von neueren Browsern unterstützt wird. Und dann ist der Preis für die Nutzung doch recht hoch: Ein Gewinn an intuitiver Nutzung bei der Webentwicklung bezahlt man damit, dass die Farben in älteren Browsern nicht mehr funktionieren. Aber an dieser Stelle kommt das `a` ins Spiel.

3.4 Am Ende steht a: rgba() und hsla()

Neben `hsl()` definiert CSS3 zwei neue Arten, Farben anzugeben, nämlich `rgba()` und `hsla()`. Diese funktionieren wie ihre Konterparts ohne `a`, aber mit einer zusätzlichen vierten Angabe – der Angabe der Transparenz. Wie bei `opacity` entspricht 0 vollkommen durchsichtig und 1 vollständig undurchsichtig. Interessant sind die Werte dazwischen.

3.4.1 Grundprinzip von hsla() und rgba()

Ein halb transparentes Rot lässt sich auf folgende Arten darstellen:

```
background-color: hsla(0, 100%, 50%, 0.5);
```

Oder über `rgba()`:

```
background-color: rgba(255, 0, 0, 0.5);
```

Hierzu ein Beispiel. Im HTML-Teil befinden sich mehrere `div`-Elemente mit `id`s:

```
<div id="eins">0</div>
<div id="zwei">0.2</div>
<div id="drei">0.4</div>
<div id="vier">0.6</div>
<div id="fuenf">0.8</div>
<div id="sechs">1</div>
```

Per CSS erhalten alle `div`s eine Breite und werden gefloatet.

Die Hintergrundfarbe samt Transparenzangabe wird bei den einzelnen Elementen über `hsla()` zugewiesen. Da wir eine weiße Hintergrundfarbe benötigen, ist der Farbton egal, wichtig ist, dass die größtmögliche Helligkeit, d. h. 100 % beim dritten Parameter gewählt ist. Der letzte Parameter variiert, um unterschiedliche Grade an Transparenz vorzuführen.

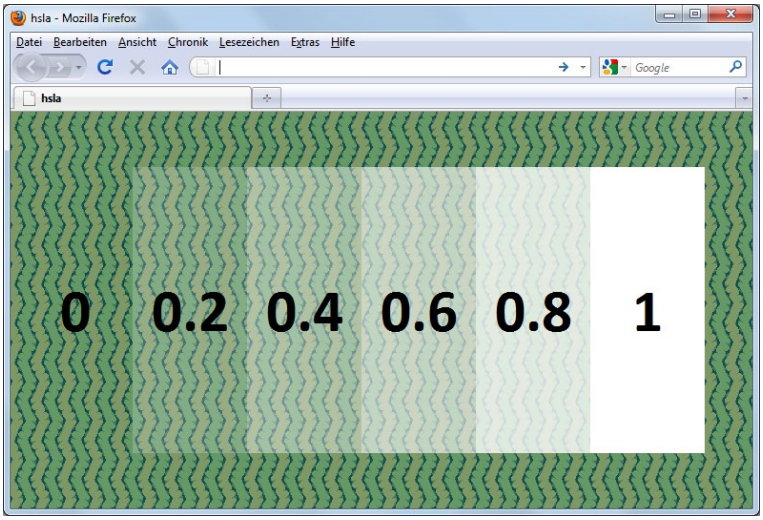
LISTING 3.4

Verschiedene Werte von `hsla()` im Vergleich (*hsla.html*)

```
#eins { background-color: hsla(0, 50%, 100%, 0); }
#zwei { background-color: hsla(0, 50%, 100%, 0.2); }
#drei { background-color:  hsla(0, 50%, 100%, 0.4); }
#vier { background-color:  hsla(0, 50%, 100%, 0.6); }
#fuenf { background-color: hsla(0, 50%, 100%, 0.8); }
#sechs { background-color: hsla(0, 50%, 100%, 1); }
```

ABBILDUNG 3.12

Bereiche mit unterschiedlich transparenten Hintergründen



Die Abbildung zeigt deutlich, dass die Bereiche unterschiedlich transparente Hintergrundfarben haben. Unberührt von der Transparenz der Hintergründe ist der Text, der gleichermaßen schwarz und opak ist. Das ist ein großer Unterschied im Vergleich zum `opacity`-Beispiel, siehe Seite 65 (Abschnitt 3.2.1).

Im Beispiel wurde `hsla()` eingesetzt. `rgba()` funktioniert im Prinzip genauso, hier ein Ausschnitt aus dem entsprechenden Listing:

LISTING 3.5

Verschiedene Grade der Transparenz bei `rgba()` (Ausschnitt aus *rgba.html*)

```
#eins { background-color: rgba(255, 255, 255, 0); }
#zwei { background-color: rgba(255, 255, 255, 0.2); }
#drei { background-color:  rgba(255, 255, 255, 0.4); }
```

Browser übersetzen übrigens `hsl()`- und `hsla()`-Angaben in `rgb()` bzw. `rgba()`. Wenn Sie sich die Formatierungen über Firebug ansehen, zeigt dieser anstelle der `hsla()`-die entsprechenden `rgba()`-Werte.

3.4.2 Nicht nur als Hintergrund

`hsla()` oder `rgba()` können Sie überall verwenden, wo Sie in CSS Farbangaben machen. Beispielsweise auch für eine Schriftfarbe:

Listing 3.6
Ausschnitt aus *hsla_schriftfarbe.html*

```
#drei { color:  hsla(0, 50%, 100%, 0.4); }
#vier { color:  hsla(0, 50%, 100%, 0.6); }
```

Abbildung 3.13
`hsla()` bei `color`

Oder für einen Rahmen:

Listing 3.7
Ausschnitt aus *hsla_rahmen.html*

```
#drei { border-color:  hsla(0, 50%, 100%, 0.4); }
#vier { border-color:  hsla(0, 50%, 100%, 0.6); }
```

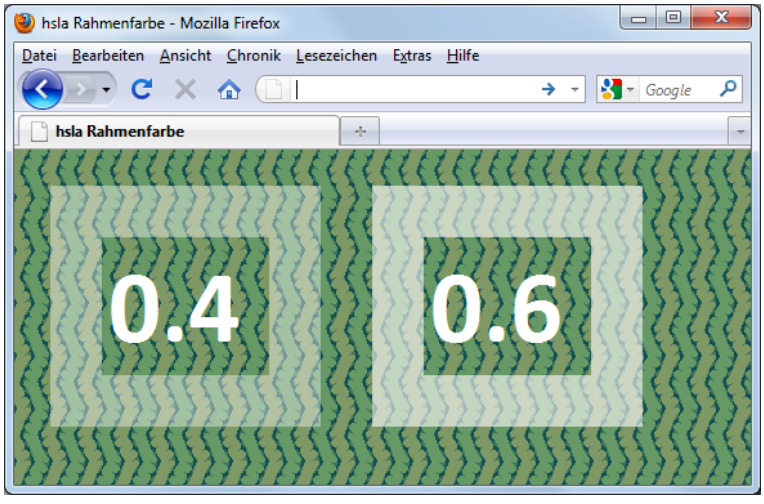


Abbildung 3.14
Teiltransparente Rahmen

Allerdings sehen diese Rahmen in Webkit-Browsern ein bisschen anders aus: An der Stelle, wo sich seitliche und obere/untere Rahmen überlappen, werden diese als weniger transparente Vierecke dargestellt.

Abbildung 3.15

In Webkit-Browsern sehen die Rahmen in den Ecken anders aus. Um das zu ändern, muss man leicht unterschiedliche Farben für die seitlichen und die Rahmenteile oben/unten spezifizieren (`http://snook.ca/archives/html_and_css/safari-transparent-borders`).

Ein teilweise transparenter Rahmen funktioniert aber nur, solange der Hintergrund des Elements ebenfalls transparent ist. Wählt man hingegen eine Hintergrundfarbe für das Element, dann schimmert diese hinter dem Rahmen durch, und der Rahmen wirkt nicht mehr transparent wie gewünscht.

Außerdem spielen `hsla()` und `rgba()` ihre Stärken gerade in der Kombination mit anderen CSS3-Eigenschaften aus: bspw. als halb transparente Schatten (`box-shadow` und `text-shadow`) u. v. m.

Wieder sind es mehrere `div`-Elemente mit `id`s, die formatiert werden sollen.

Und hier die entscheidenden Formatierungen:

Listing 3.8

Ausschnitt aus *hsla_rahmen2.html*

```
div {
/* Weitere Formatierungen ausgelassen */
  background-color: #066;
}
#eins { border-color:  hsla(0, 50%, 100%, 0); }
#zwei { border-color:  hsla(0, 50%, 100%, 0.2); }
#drei { border-color:  hsla(0, 50%, 100%, 0.4); }
#vier { border-color:  hsla(0, 50%, 100%, 0.6); }
#fuenf { border-color:  hsla(0, 50%, 100%, 0.8); }
#sechs { border-color:  hsla(0, 50%, 100%, 1); }
```

Abbildung 3.16

Teiltransparente weiße Rahmen auf eingefärbter Box

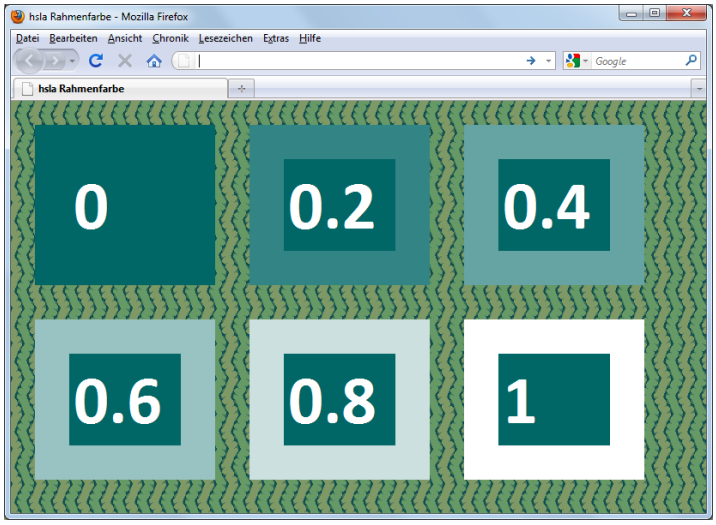


Abbildung 3.16 zeigt Rahmen mit unterschiedlichem Grad an Transparenz auf einem in einem Blauton eingefärbten Element – von Transparenz ist nichts zu sehen. Auch für dieses Problem gibt es eine Lösung über CSS3 mithilfe der Eigenschaft `background-clip`, wie sie in *Kapitel 6* besprochen wird.

Andererseits demonstriert das Beispiel eine weitere Möglichkeit von Transparenzen: Man kann sie nutzen, um eine Navigation oder ein ähnliches Element anders einzufärben, ohne den Farbton zu ändern.

Eine schöne Webseite, um sich mit dem Zusammenspiel der verschiedenen Eigenschaften von HSLA vertraut zu machen, ist `http://mothereffinghsl.com/`. Hier steuern Sie über Regler die einzelnen Werte und sehen direkt die Auswirkung.

3.4.3 rgba() und hsla() am Beispiel: Bildbeschriftung

Sehen wir uns jetzt ein praktisches Beispiel an: ein halb transparenter Bereich auf einem Bild, der einen opaken Text beinhaltet.

Abbildung 3.17

So soll die Bildbeschriftung aussehen.

Erst einmal der HTML-Code:

```
<div class="beschriftung">
  <img src="steine.jpg" alt="Steine" width="400" height="381" />
  <span>Steine, Muscheln und Sand</span>
</div>
```

Dann zum CSS-Code: Das umfassende Element erhält eine Breite und wird relativ positioniert. Diese relative Positionierung ist wichtig, damit es als Bezugspunkt für die darauf positionierte Beschriftung dienen kann.

```
.beschriftung {
  position: relative;
  width: 400px;
}
```

Das Bild wird zu einem Blockelement:

```
.beschriftung img {
  display: block;
}
```

Schließlich wird die Beschriftung absolut positioniert und erhält die gewünschte Hintergrundfarbe über `hsla()`:

```
.beschriftung span {
  position: absolute;
  bottom: 0;
  left: 0;
  width: 100%;
  background: hsla(0, 100%, 0%, 0.6);
  color: #fff;
  text-align: center;
  height: 4em;
  line-height: 4em;
  border-top: 1px solid #999;
  border-bottom: 1px solid #999;
}
```

Mangels Unterstützung für `hsla()` ist die Beschriftung im IE < 9 schlecht lesbar. Das müssen wir verbessern.

3.4.4 Strategien für ältere Browser

Die Unterstützung für `hsl()`, `hsla()` und `rgba()` ist dieselbe und kann deshalb gemeinsam behandelt werden. Sie funktionieren in allen modernen Browsern, im Internet Explorer ab Version 9.

Möglichkeit 1 – opake Farbe als Fallback zuerst notieren

Ältere IE ignorieren die über `rgba()` oder `hsla()` angegebene Hintergrundfarbe und wählen entsprechend transparent. Das kann bewirken, dass der Text nicht mehr zu lesen ist. Als Minimalversion sollten Sie deswegen statt der Farbe mit der Transparenz eine andere opake Farbe angeben:

```
background: #999;
background: hsla(0, 100%, 0%, 0.6);
```

LISTING 3.9

Im Listing *bildbeschriftung_fix_ie1.html* ist das schon einmal so implementiert.

Aber Achtung: Wenn Sie hier anstelle von `background` mit `background-color` arbeiten, ignorieren aus unerfindlichen Gründen sowohl der IE6 als auch der IE7 diese Angabe.

Wichtig ist, dass Sie zuerst die Fallbacklösung angeben und dann die `rgba()`- oder `hsla()`-Angabe: Nur so stellen Browser, die `hsla()` unterstützen, das auch dar.

Möglichkeit 2 – PNG-Bild als Fallback

An sich können Sie für den Fall, dass `rgba()` und `hsla()` nicht unterstützt werden, als Fallbacklösung ein semitransparentes PNG-Bild nehmen. Diese Lösung ist allerdings recht aufwändig, weil sie die Erstellung eines Bildes erfordert (und außerdem funktioniert sie nicht im IE6).

Möglichkeit 3 – Filter zur Hilfe

Es gibt einen Filter, um `rgba()`/`hsla()` im IE zu simulieren, den sogenannten Gradient-Filter. Für einen halb transparenten weißen Hintergrund benötigen Sie folgende Angabe:

```
filter:progid:DXImageTransform.Microsoft.gradient( startColorstr=#7BF
  FFFFF,endColorstr=#7BFFFFFF);
```

Ab Internet Explorer 8 kann der Filter auch durch die folgende Syntax wiedergegeben werden (zur allgemeinen Frage, ob man `-ms-filter` überhaupt braucht, siehe auch *Kapitel 2*):

```
-ms-filter:"progid:DXImageTransform.Microsoft.gradient(startColorstr=
   #7BFFFFFF,endColorstr=#7BFFFFFF)";
```

Der Gradient-Filter ist eigentlich für Farbverläufe vorgesehen, deswegen erwartet er einen Wert für die Farbe am Anfang und die Farbe am Ende. Wenn Sie den Filter für die Transparenz nutzen, sind die beiden Werte dieselben.

Bei der Angabe `#7BFFFFFF` handelt es sich um vier hexadezimale Werte: Die beiden ersten Zeichen (`7B`) benennen den Grad der Transparenz, die weiteren drei Zeichenpaare in gewohnter Manier die Farben Rot, Grün und Blau.

Je höher der erste Wert, desto opaker ist die Farbe. Um den richtigen Wert zu ermitteln, gehen Sie am besten von dezimalen Werten aus: 255 würde vollständig opak bedeuten, 0 ganz durchsichtig. Den gewählten Wert rechnen Sie dann in eine hexadezimale Zahl um, beispielsweise mithilfe des Rechners von Selfhtml unter `http://de.selfhtml.org/helferlein/dezhex.htm`.

Aber es gibt auch Tools, die Ihnen die Arbeit abnehmen:

- Bei `http://kimili.com/journal/rgba-hsla-css-generator-for-internet-explorer/` geben Sie einfach die `hsla()`- oder `rgba()`-Farbe an, und die benötigten Filterangaben werden automatisch erzeugt. Hier werden die `-ms-filter`- und die `filter`-Angabe erzeugt.
- Auch `http://css3please.com/` bietet die Option, `rgba()`-Werte in die entsprechenden Filterangaben für den IE zu konvertieren. Hier werden übrigens nur die `filter`-Angaben generiert, auf die `-ms-filter`-Angabe wird verzichtet.

Übrigens überprüft Modernizr ebenfalls die Unterstützung für `hsla()` und `rgba()`. Er ergänzt entsprechend die Klassen `rgba`, `no-rbga` und `hsla` und `no-hsla`.

Eine Erleichterung bei dieser Herangehensweise könnte das Skript von Lea Verou sein: Es steht unter `http://leaverou.me/rgba.php/` und erzeugt automatisch aus einer `rgba()`-Angabe serverseitig per PHP das benötigte PNG-Bild.

Die Kombi machts

Sehen wir uns die erforderlichen Schritte einmal praktisch anhand der halb transparenten Bildbeschriftung an: Zuerst setzen Sie eine nicht transparente Hintergrundfarbe für ältere Firefox- (Version 2 und abwärts) und Opera-Browser vor Version 10, dann kommt die `hsla()`-Angabe für neuere Browser:

```
<style>
.beschriftung span {
  background: #999;
  background: hsla(0, 100%, 0%, 0.6);}
</style>
```

Innerhalb von konditionalen Kommentaren folgen darunter die gesonderten Angaben für den Internet Explorer vor Version 9. Zuerst wird die Hintergrundfarbe auf `transparent` gesetzt, dann werden die Filter definiert.

LISTING 3.10

Ausschnitt aus *bildbeschriftung_fix_ie2.html*

```
<!--[if lt IE 9]>
<style type="text/css">
.transparent {
  background-color: transparent;
  filter:progid:DXImageTransform.Microsoft.gradient(startColorstr=
    #7BFFFFFF,endColorstr=#7BFFFFFF);
  -ms-filter:"progid:DXImageTransform.Microsoft.gradient(startColorstr=
      #7BFFFFFF,endColorstr=#7BFFFFFF)";
  zoom: 1;
}
</style>
<![endif]-->
```

Die restlichen CSS-Angaben bleiben wie gehabt.

Wahlweise sollten Sie auch in Erwägung ziehen, die besonderen Angaben für den Internet Explorer über die dem `html`-Start-Tag zugewiesenen Klassen zu machen (siehe *Kapitel 2*).

3.4.5 Nachbessern oder nicht

Dass Sie Filter einsetzen können, um diesen Effekt im IE zu simulieren, heißt natürlich nicht, dass Sie es sollen. Interessant ist in dieser Hinsicht die Seite `http://24ways.org`.

Abbildung 3.18

24Ways im Firefox – elegante Transparenzen überall

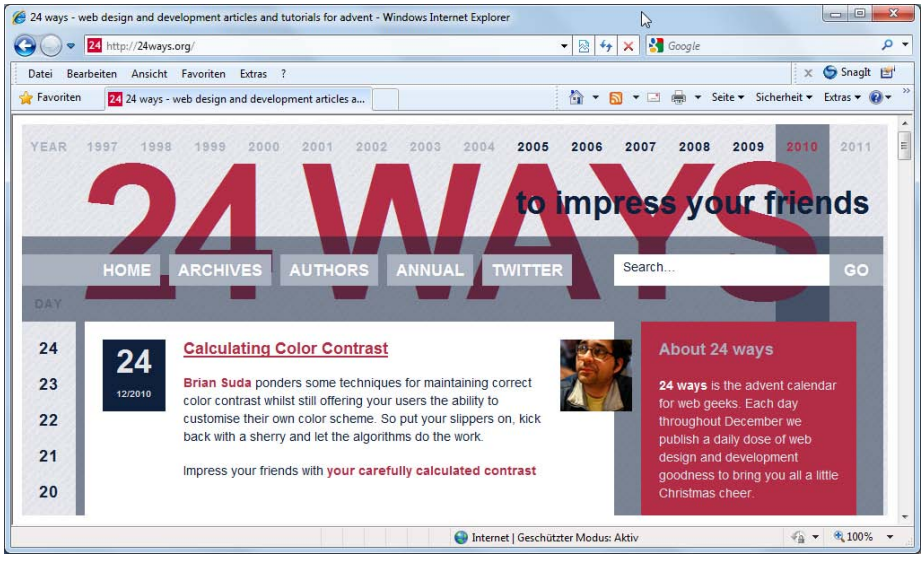


Abbildung 3.19

Dieselbe Seite im IE mit weniger Transparenzen, aber immer noch gut benutzbar und schön

Sieht man sich das genau im Vergleich an, so zeigt sich, dass ein Teil der Elemente auch im IE8 transparent ist. Die Ebene, die die Zahl 24 überlagert, ist im IE8 halb transparent. Die Buttons hingegen sind nur im Firefox & Co. transparent, im IE8 hingegen opak.

Abbildung 3.20

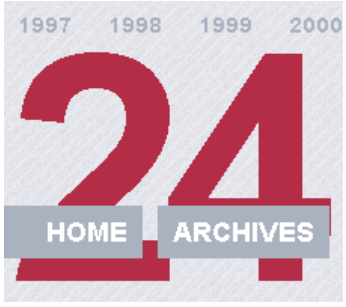
Teile des Designs sind auch im IE8 (rechts) transparent, aber nicht alles.

Diese Webseite setzt für IE8 und IE7 an ausgewählten Stellen transparente PNGs ein. Diese sind in einem eigenen Stylesheet definiert, das über konditionale Kommentare eingebunden ist. Für den IE6 werden keine transparenten PNGs eingesetzt, weil man da auf einen zusätzlichen Filter zurückgreifen müsste.

Wer sich Webstandards verpflichtet fühlt, wird versuchen, so weit wie möglich auf Filter zu verzichten.

Abbildung 3.21

Ganz ohne Transparenzen, aber benutzbar ist die Webseite von 24ways im IE6.



Dieses Beispiel zeigt schön, dass nachbessern nicht bedeutet, dass man überall und immer und für alle Browser nachbessern muss.

3.5 Steckbrief

`hsl()`, `hsla()`, `rgba()` und `opacity` werden im *CSS Color*-Modul definiert (`http://www.w3.org/TR/css3-color/`). Dieses hat bereits seit Juni 2011 den Status einer Proposed Recommendation.

Mit `opacity` machen Sie Elemente teiltransparent, wobei die Transparenz an Kindelemente vererbt wird.

Eigenschaft	Firefox	Safari	Chrome	Opera	IE	Alternativen
`opacity`	ja	ja	ja	ja	9	Alpha-Filter (kein Standard)

Tabelle 3.1 Browserunterstützung der CSS3-Eigenschaft `opacity`

Die Version, ab wann ein Browser ein bestimmtes Feature unterstützt, wird nur angegeben, wenn ein Browser das Feature erst vor Kurzem eingefügt hat.

Um einzelne Farben transparent zu setzen, sind `hsla()` und `rgba()` vorgesehen. Beide können überall dort eingesetzt werden, wo Farben definiert werden.

Farbangabe	Firefox	Safari	Chrome	Opera	IE	Alternativen
`hsl()`	ja	ja	ja	ja	9	rgb(), hexadezimale Farbangabe
`rgba()`	ja	ja	ja	ja	9	Gradient-Filter (kein Standard)
`hsla()`	ja	ja	ja	ja	9	Gradient-Filter (kein Standard)

Tabelle 3.2 Browserunterstützung für `hsl()`, `hsla()` und `rgba()`

Copyright