

## Kapitel 14



### Der Flow und »position«

*Worin Sie sehen, dass die Kästchen auf einer Seite einem natürlichen Fluss folgen. Überdies lernen Sie, wie Sie den Verlauf dieses Flusses manipulieren und die Kästchen mit einer Eigenschaft namens »position« verschieben können.*

Die Themen im Überblick:

- »Flow«: Die Seite ist ein langer, ruhiger Fluss, Seite 282
- Der normale Fluss der Boxen, Seite 283
- Versetzt weiterfließen: »position: relative«, Seite 285
- Raus aus dem Fluss: »position: absolute«, Seite 286
- Absolute Positionierung auf der Beispielsite, Seite 289
- Wie ein Fels in der Brandung: »position: fixed«, Seite 294
- Positionierte Boxen und der »z-index«, Seite 296
- Auf einen Blick, Seite 298

Es gibt drei Möglichkeiten, Boxen auf einer Webseite zu positionieren:

1. **Normaler »Flow«:** Die Boxen werden in der Reihenfolge ihres Vorkommens im Quellcode dargestellt. Dafür gibt es die Eigenschaften `position: static` (der Standardwert) und `position: relative`.
2. **Absolute Positionierung:** Absolut positionierte Elemente werden aus dem Fluss herausgehoben und sind für die anderen Elemente nicht vorhanden. Dazu zählen `position: absolute` und `position: fixed`.
3. **Float:** Schwebende – im CSS-Jargon *gefloatede* – Elemente sind ein Zwischending. Zuerst wird die Box normal positioniert, dann wird sie zum Teil aus dem Fluss herausgehoben und schwebt so weit wie möglich nach rechts oder links.

Zunächst lernen Sie die Grundlagen der Positionierung mit `position` kennen, danach die Grundlagen zu `float` und anschließend wenden Sie all dieses Wissen in mehrspaltigen Layouts praktisch an. Aber los geht es zunächst mit ein paar Erläuterungen zum Normalzustand einer Webseite, dem *document flow*.

## 14.1 »Flow«: Die Seite ist ein langer, ruhiger Fluss

Normalerweise folgen die Elemente im sichtbaren Bereich des Browserfensters dem *document flow*, dem »Fluss des Dokuments«. Alle HTML-Elemente schwimmen in diesem Fluss. Er ist der natürliche Zustand einer Webseite und für Webdesigner so wichtig wie die Schwerkraft für Architekten. Man unterscheidet dabei wie gehabt zwischen Block- und Inline-Elementen.

Zunächst zu den Block-Elementen:

- Das erste Block-Element wird so weit wie möglich links und oben platziert.
- Block-Elemente werden immer so breit wie möglich, also so breit wie das umgebende Element, und haben immer einen integrierten Zeilenumbruch. Nachfolgende Block-Elemente stehen immer *unterhalb* des vorhergehenden Block-Elements.

Inline-Elemente verhalten sich etwas anders:

- Das erste Inline-Element wird ebenfalls so weit wie möglich links und oben platziert.
- Folgende Inline-Elemente werden jeweils rechts davon angeordnet, und zwar so lange, bis kein Platz mehr ist.
- Wenn rechts kein Platz mehr ist, rutschen sie eine Zeile tiefer und beginnen wieder ganz links.

Panta rhei – alles fließt. Der Flow macht Webseiten so flexibel wie die im ersten Kapitel auf Seite 44 beschriebene Zeitung. Als Webdesigner sollten Sie diesen Flow *verstehen* und versuchen, mit ihm zu arbeiten, nicht gegen ihn.

## 14.2 Der normale Fluss der Boxen

Um ein bisschen Gefühl für den Flow zu bekommen, studieren Sie vor der Zähmung der Widerspenstigen beim Layouten zunächst das natürliche Verhalten der Boxen in freier Wildbahn.

### Drei Boxen im Fluss

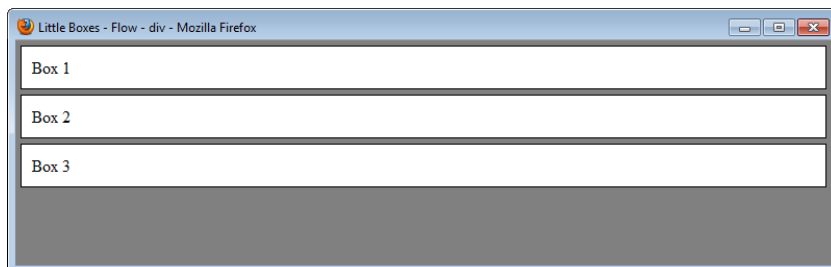
Schauen Sie sich folgendes HTML und CSS an:

HTML	CSS
<pre>&lt;body&gt;   &lt;div&gt;Box 1&lt;/div&gt;   &lt;div&gt;Box 2&lt;/div&gt;   &lt;div&gt;Box 3&lt;/div&gt; &lt;/body&gt;</pre>	<pre>body {   color: black;   background-color: gray;   padding: 0;   margin: 0; } div {   background-color: white;   padding: 10px;   border: 1px solid black;   margin: 5px; }</pre>

Tabelle 14.1:  
Drei Boxen  
in Fluss des  
Dokuments

Weil `div` ein Block-Element ist, stehen die drei Kästen in Abbildung 14.1 untereinander, auch wenn sie nur wenig Inhalt haben.

Abbildung 14.1:  
Block-Elemente  
werden so breit,  
wie es geht, und  
stehen unter-  
einander.



Ein Block-Element ist von Natur aus raumgreifend veranlagt und nimmt immer die verfügbare Breite des Eltern-Elements ein, in diesem Fall also von body.

### Drei verkürzte Boxen im Fluss

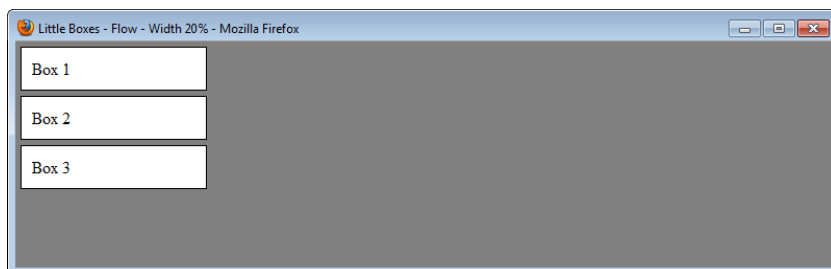
Im nächsten Schritt werden die drei div-Geschwister mit `width` auf 20% verkürzt:

Listing 14.1:  
Verkürzung von  
»div« auf 20%

```
div {  
    width: 20%;  
    background-color: white;  
    padding: 10px;  
    border: 1px solid black;  
    margin: 5px;  
}
```

Alles andere bleibt unverändert. 3 mal 20 sind 60. Stehen die Boxen jetzt nebeneinander? Nein, das tun sie nicht. Abbildung 14.2 zeigt, dass auch die drei kurzen Boxen untereinander stehen. `div` ist ein Block-Element, und Block-Elemente haben einen integrierten Zeilenumbruch. Die Eigenschaft `width` verkürzt die Box zwar, der Zeilenumbruch aber bleibt.

Abbildung 14.2:  
Auch die kurzen  
Boxen bleiben  
untereinander  
stehen.



Dieses normale Verhalten der Boxen im Flow wird wie erwähnt auch als `position:static` bezeichnet. Ausgerechnet die Positionierung, die dem natürlichen Fluss der Elemente freien Lauf lässt, bekommt den Namen *static*. So spielt das Leben.

### 14.3 Versetzt weiterfließen: »position: relative«

Die relative Positionierung mit `position: relative` macht zwei Dinge:

- Sie verschiebt die Box von ihrer normalen Position im Fluss.
- Sie markiert den ursprünglichen Platz des Elements als geschützt.

Die anderen Elemente im Dokument verhalten sich so, als ob das Element noch an seinem *ursprünglichen* Platz im normalen Fluss stehen würde.

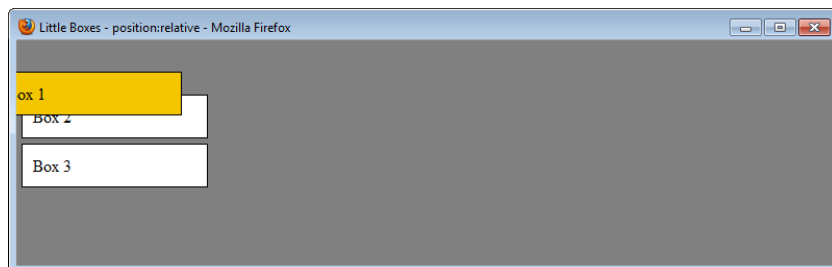
HTML und CSS werden für dieses Beispiel ein klein wenig geändert:

HTML	CSS
<pre> &lt;body&gt;   &lt;div id="anders"&gt;Box 1&lt;/div&gt;   &lt;div&gt;Box 2&lt;/div&gt;   &lt;div&gt;Box 3&lt;/div&gt; &lt;/body&gt; </pre>	<pre> body {   color: black;   background-color: gray;   padding: 0;   margin: 0; } div {   width: 20%;   background-color: white;   padding: 10px;   border: 1px solid black;   margin: 5px; } #anders {   position: relative;   top: 25px;   right: 25px;   background-color: #f3c600; } </pre>

Tabelle 14.2:  
Relative  
Positionierung

Abbildung 14.3 zeigt das Ergebnis dieser Änderung im Browserfenster.

Abbildung 14.3:  
Relative Positionierung in Aktion



Beachten Sie, dass Box 2 und 3 sich überhaupt nicht verändert haben. Bei der relativen Positionierung bleibt die ursprüngliche Position des Elements wie gesagt geschützt und wird nicht von den nachfolgenden Elementen beansprucht.

Ein positioniertes Element bekommt seine genauen Koordinaten mit den Eigenschaften `top`, `right`, `bottom` und `left`, die bei der relativen Positionierung von der ursprünglichen Position der Box im Flow aus gemessen werden:

- `top:25px` drückt die Box nach *unten*. An der normalen Position der Box werden oben 25 Pixel eingefügt.
- `right:25px` schiebt die Box nach *links*, sodass sie zum Teil verschwindet. Die Box wird also von rechts um 25 Pixel verschoben.

Die Eigenschaften `top`, `right`, `bottom` und `left` geben die Position in einem Koordinatensystem an, und der Bezugspunkt ist bei der relativen Positionierung die ursprüngliche Position der Box im Flow. Das ist in sich ganz logisch, aber trotzdem wirkt `position:relative` auf den ersten Blick ein bisschen wie von hinten durch die Brust ins Auge.

## 14.4 Raus aus dem Fluss: »position: absolute«

Im Gegensatz zur relativen nimmt die *absolute Positionierung* das Element komplett aus dem Fluss heraus. Das Element wird – bildlich gesprochen – hochgezogen, und alle anderen Elemente auf der Seite verhalten sich so, als ob es gar nicht da wäre.

Das HTML für dieses Beispiel ist absolut identisch mit dem für die relative Positionierung, und im CSS wird genau ein Wort geändert:

HTML	CSS
<pre>&lt;body&gt; &lt;div id="anders"&gt;Box 1&lt;/div&gt; &lt;div&gt;Box 2&lt;/div&gt; &lt;div&gt;Box 3&lt;/div&gt; &lt;/body&gt;</pre>	<pre>body {   color: black;   background-color: gray;   padding: 0;   margin: 0; } div {   width: 20%;   background-color: white;   padding: 10px;   border: 1px solid black;   margin: 5px; } #anders {   position: <b>absolute</b>;   top: 25px;   right: 25px;   background-color: #f3c600; }</pre>

Tabelle 14.3:  
Absolute  
Positionierung

Nur ein einziges Wort wurde im CSS geändert, aber die Wirkung ist enorm. Box 1 steht plötzlich rechts, und bemerkenswerterweise rutschen die Boxen 2 und 3 nach oben:

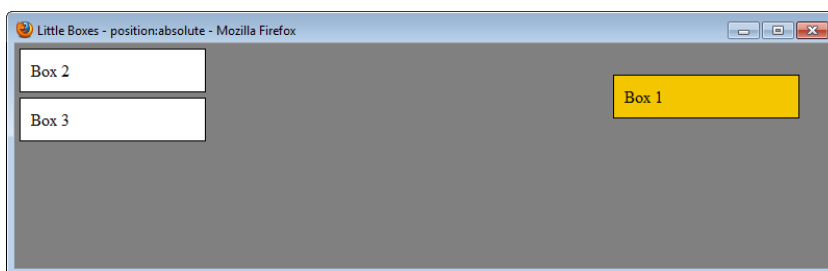


Abbildung 14.4:  
Nur ein einziges  
Wort geändert –  
absolute Positionierung

Absolut positionierte Elemente werden aus dem Fluss herausgehoben und liegen *über* den anderen Elementen. Bei normal fließenden Elementen ist der Browser dafür verantwortlich, dass die Boxen sich nicht überlappen, bei absolut positionierten Elementen hingegen ist

der Webdesigner verantwortlich, der die absolute Positionierung veranlasst hat.

Die genaue Position eines absolut positionierten Elements wird wieder durch `top`, `right`, `bottom` oder `left` angegeben, aber die Werte für diese vier Eigenschaften orientieren sich nicht mehr an der ursprünglichen Position der Box im Fluss:

1. Die absolute Positionierung eines Elements bezieht sich auf das nächste umgebende Element (*containing block*), das mit `relative`, `absolute` oder `fixed` positioniert ist.
2. Falls kein positioniertes umgebendes Element vorhanden ist (was in der Praxis häufig der Fall ist), erfolgt die Positionierung relativ zum obersten Element des Dokumentbaums, und das ist nicht `body`, sondern `html` (*initial containing block*). In der Regel bedeutet dies so viel wie »vom Rand des Browserfensters aus gerechnet«.

Diese beiden Aussagen kann man so zusammenfassen:

- Absolute Positionierung ist *relativ* zu einem ganz bestimmten Bezugspunkt.
- Es gibt zwei *verschiedene* mögliche Bezugspunkte:
  - ein umgebendes positioniertes Element
  - das Stammelement `html` (der Rand des Browserfensters)

Diese Tatsachen sorgen im Alltag für viel Verwirrung und sollen deshalb im Folgenden anhand eines Beispiels genauer unter die Lupe genommen werden.

## Tipp

### Überlappungen kontrollieren mit »z-index«

Falls sich absolut positionierte Elemente überlappen, können Sie mithilfe der Eigenschaft `z-index` festlegen, welche Elemente vorne und welche hinten liegen. Mehr dazu finden Sie ab Seite 296.



## 14.5 Absolute Positionierung auf der Beispielsite

In diesem Abschnitt möchte ich Ihnen zeigen, wie Sie die absolute Positionierung in der Praxis einsetzen können.

Auf der »Little Boxes«-Startseite steht das Logo im Kopfbereich zwischen `<h1>` und `</h1>`. Darunter steht der Absatz »Webseiten gestalten mit HTML und CSS. Grundlagen.«, der in diesem Abschnitt mithilfe absoluter Positionierung rechts neben die Grafik verschoben werden soll.

### 1. Absolute Positionierung relativ zum Rand des Browserfensters

In der ersten Variante ist keines der umgebenden Elemente relativ positioniert, und daher werden die Werte für `top` und `left` im folgenden ToDo vom Rand des Browserfensters aus berechnet – oder genauer gesagt: vom Stammelement `html`.

#### ToDo: Positionierung des Absatzes relativ zum Stammelement `html`

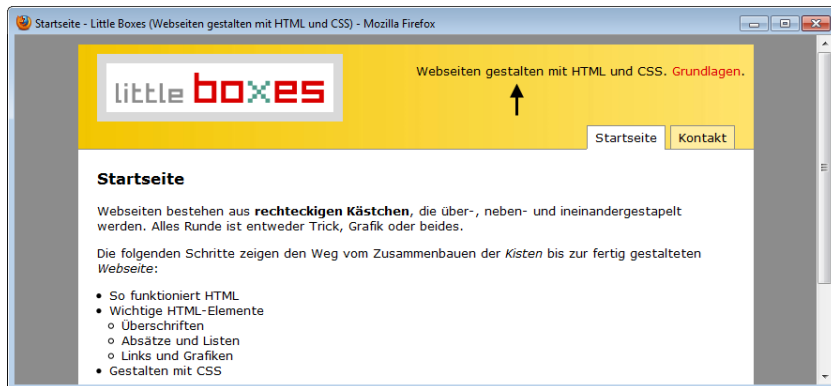
1. Ändern Sie die Regel für `p#slogan` wie folgt:

```
p#slogan {
  position: absolute;
  top: 25px ;
  left: 50%;
  padding: 5px 0 5px 0;
  margin-bottom: 0;
}
```

2. Speichern Sie das Stylesheet, und betrachten Sie die Webseiten im Browser.

Die Webseite sieht mit dieser Regel im Browser ungefähr so aus:

Abbildung 14.5:  
Absolut positioniert,  
scheinbar okay



Auf den ersten Blick sieht die Seite ganz okay aus. Die folgenden Dinge sind passiert:

- Zuerst wird `p#slogan` komplett aus dem Fluss herausgehoben und ist somit für andere Elemente nicht mehr vorhanden. Dadurch rutscht der Navigationsbereich ein Stück nach oben.
- Da es kein umgebendes positioniertes Element gibt, beziehen sich die Angaben `top:25px` und `left:50%` auf das Stammelement `html`. Der Absatz beginnt 25px von oben und genau auf der Hälfte des Browserfensters.

Die vertikale Positionierung vom oberen Rand des Browserfensters ist auf diese Weise noch einigermaßen genau möglich, aber die horizontale Positionierung mit 50% von links beruht auf Versuch und Irrtum und ist reine Glückssache.

Auch wenn die Webseite in Abbildung 14.5 okay aussieht, ist sie ein Kartenhaus und nicht besonders stabil. Abbildung 14.6 zeigt, was passiert, wenn zum Beispiel der Wrapper nicht mehr im Browserfenster zentriert wird. Dem positionierten Absatz sind solche Änderungen egal. Er bleibt horizontal einfach bei 50% des Browserfensters stehen, weil sein Bezugspunkt das Stammelement `html` ist. Er weiß nichts von einem Kopfbereich.

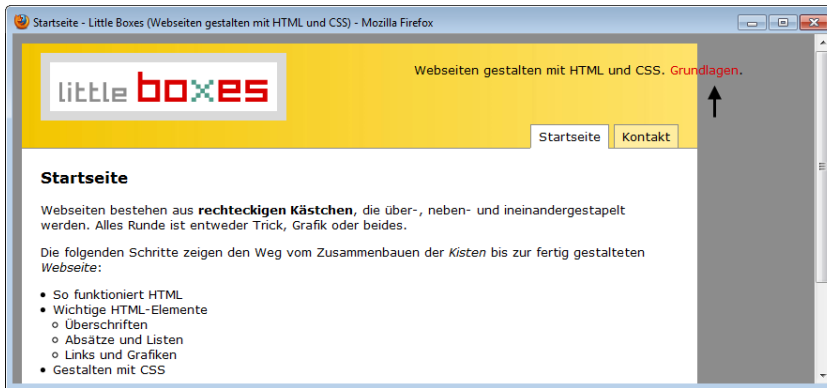


Abbildung 14.6:  
Absolut positioniert, nicht mehr okay

Um Missverständnissen vorzubeugen: Die absolute Positionierung zum Stammelement `html` hat durchaus ihre Berechtigung, zum Beispiel in den ab Seite 343 gezeigten mehrspaltigen Layouts mit absoluter Positionierung, aber in diesem Beispiel ist die im folgenden Abschnitt gezeigte Kombination von absoluter Positionierung mit einem relativ positionierten umgebenden Element die bessere Lösung.

## 2. Absolute Positionierung mit einem umgebenden, relativ positionierten Element

Besser und vor allem zuverlässiger wäre es im vorliegenden Fall, den Absatz *relativ zum Kopfbereich* zu positionieren, und genau das erreichen Sie durch einen einfachen Trick. Wie hieß es etwas weiter oben so schön:

*Die absolute Positionierung eines Elements bezieht sich auf das nächste umgebende Element (containing block), das mit relative, absolute oder fixed positioniert ist.*

Und jetzt der Trick: Wenn Sie den Kopfbereich mit `position: relative` versehen, ihm aber keinerlei Werte für `top` & Co mit auf den Weg geben, bleibt er im Flow, wird nicht verschoben und wird zum neuen Bezugspunkt für die absolute Positionierung von `p#slogan`.

Klingt kompliziert? Probieren Sie es aus. Dieser Trick ist eine echte Basistechnik beim Layouten mit CSS. Im folgenden ToDo wird `p#slogan` positioniert, und durch das `position: relative` bekommen die Werte für `top` und `right` den Kopfbereich als neuen Bezugspunkt.

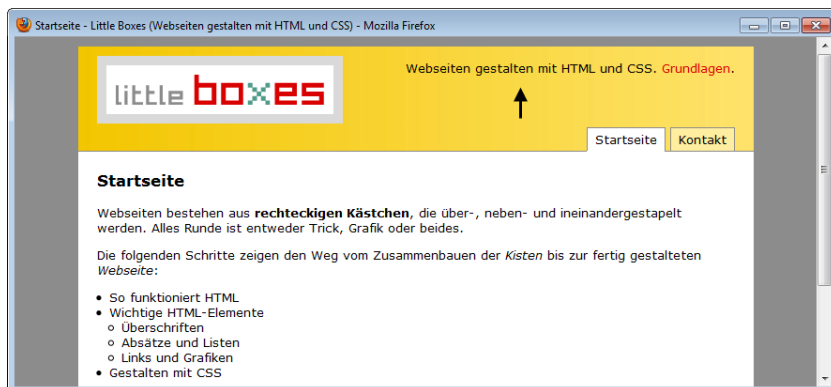
## ToDo: Positionierung des Absatzes relativ zum Kopfbereich

1. Ändern Sie das Stylesheet *bildschirm.css* wie folgt:

```
#kopfbereich {
    position: relative; /* positioniert, aber bleibt im Fluss */
    background: #ffe574 url(farbverlauf.jpg) repeat-y left top;
    color: black;
    padding: 10px 20px 0 20px;
}
```

Auf den ersten Blick sieht das Ergebnis in Abbildung 14.7 gar nicht so viel anders aus als in Abbildung 14.5.

Abbildung 14.7:  
Absolut positioniert, relativ zum Kopfbereich



Die Kombination von absoluter und relativer Positionierung bietet einige Vorteile:

*Der Bezugspunkt für den Absatz ist jetzt nicht mehr der Rand des Browserfensters, sondern #kopfbereich, und zwar genau genommen die äußere Kante des padding (»the padding edge«) oben und rechts.*

Wenn zum Beispiel die Seite selbst nicht mehr zentriert wird, macht das nichts, der Absatz wird den Kopfbereich niemals verlassen. Die *absolute* Positionierung wird durch diesen kleinen Trick sehr nützlich und flexibel einsetzbar.

Wenn der Wrapper nicht mehr im Browserfenster zentriert wird, bleibt der Absatz trotzdem innerhalb des Kopfbereichs (siehe Abbildung 14.8).

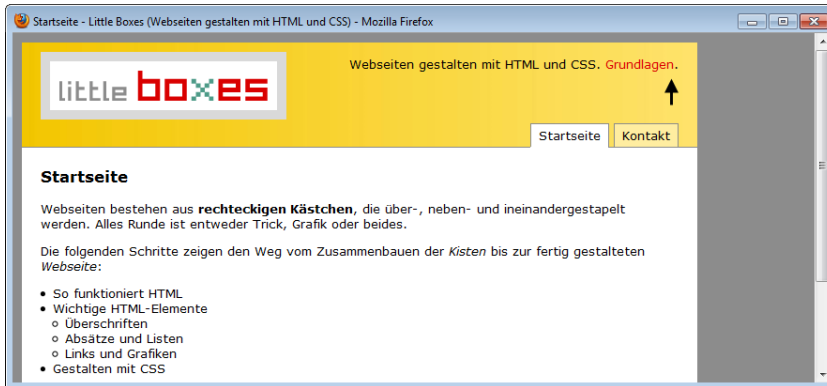


Abbildung 14.8:  
Absolut positioniert, relativ zum Kopfbereich

Mit einem einfachen `display: block` für das `span`-Element können Sie das rot eingefärbte Wort »Grundlagen« übrigens eine Zeile tiefer setzen. Und mit einem `text-align: right` für das umgebende `p#slogan` werden beide Zeilen rechts ausgerichtet werden. Probieren Sie es ruhig einmal aus.

### Grundlagentechnik: absolute und relative Positionierung kombinieren

#### Tipp

Die in diesem Beispiel gezeigte Kombination von absoluter Positionierung in Verbindung mit einem umgebenden, relativ positionierten Element ist eine sehr nützliche Grundlagentechnik zur Positionierung, die auf Webseiten häufig eingesetzt wird.

## 14.6 Wie ein Fels in der Brandung: »position: fixed«

Die feste Positionierung mit `position:fixed` verhält sich fast genau wie `position:absolute`, mit einem kleinen, aber feinen Unterschied: Ein fixiertes Element scrollt nicht mit.

Absolut positionierte Elemente sind relativ zu einem Bezugspunkt im Dokument und scrollen daher mit. Bei fixierten Elementen ist das anders:

*Das umgebende Element (containing block) ist für fixierte Elemente immer das Browserfenster (der sogenannte »Viewport«) und nicht das Stamm-element html innerhalb dieses Fensters.*

Da das Browserfenster selbst nicht mitscrollt, bleiben auf der Seite fixierte Elemente stehen. Wie bei der absoluten Positionierung ist nicht mehr der Browser, sondern der Webdesigner dafür verantwortlich, dass fixierte Elemente sich nicht danebenbenehmen.

Der Internet Explorer 6 ignoriert `position:fixed` übrigens komplett, was für das gewünschte Layout zum Teil drastische Folgen haben kann. Hier sehen Sie ein kleines Beispiel. Gegeben sei folgendes HTML, in dem das fett gedruckte `div`-Element per CSS fixiert wird.

Listing 14.2:  
Ein bisschen  
HTML mit zu  
fixierendem  
»div«-Element

```
<body>
<div id="fixiert">Fixierte Box</div>
<p>Absatz 1. Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Morbi rhoncus volutpat nisl. Praesent elementum odio ac nibh.
Duis at quam nec dolor consequat blandit. Sed libero. Vivamus
faucibus purus non purus. Suspendisse id ante ut nulla facilisis
porta.</p>
<p>Absatz 2. Nullam vulputate hendrerit nunc. Nullam dapibus blandit
orci. Nunc metus. Sed sed ante. Cras interdum, erat at pharetra
sodales, elit ligula nonummy nisi, sit amet auctor purus leo vel
urna. Pellentesque ac augue sit amet ipsum nonummy sodales. Sed
libero augue, ultricies et, tristique ut, posuere commodo, ligula.
Integer aliquet. Donec varius lectus. </p>
</body>
```

Und hier ist das CSS dazu:

```
p { margin-left: 25%; }
#fixiert {
  position: fixed;
  top: 10px;
  left: 10px;
  background-color: #f3c600;
  width: 15%;
  padding: 10px;
  border: 1px solid black;
}
```

Diese beiden Quelltext-Schnipsel ergeben folgendes Bild, links im Firefox, rechts im Internet Explorer 6:

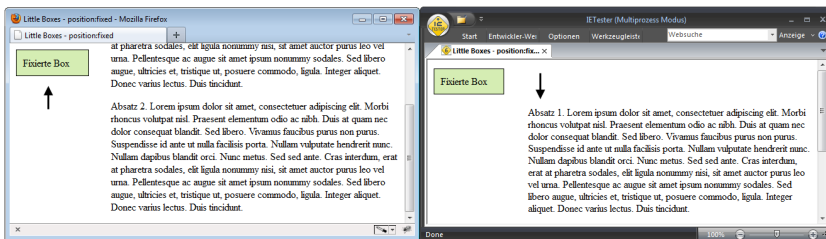


Abbildung 14.9:  
»position:fixed«  
im Firefox (links)  
und im Internet  
Explorer 6 (rechts)

Wie Sie sehen, ignoriert der Internet Explorer 6 `position:fixed` komplett, und die HTML-Elemente fließen ganz normal weiter. Diese Tatsache macht den Einsatz von `position:fixed` zwar nicht prinzipiell unmöglich, erhöht den Aufwand aber doch beträchtlich. Ab Version 7 interpretiert der IE die Anweisung `position: fixed` übrigens korrekt.

## Workarounds für IE6 und »position: fixed«

Wenn Sie trotzdem nicht auf den Einsatz von fixierten Bereichen verzichten möchten, gibt es im Web zahlreiche Tutorials zu Workarounds, zum Beispiel bei *The Styleworks*:

■ [thestyleworks.de/tut-art/iewinfixed.shtml](http://thestyleworks.de/tut-art/iewinfixed.shtml)

## Tipp

## 14.7 Positionierte Boxen und der »z-index«

Absolut positionierte Boxen haben zwei bemerkenswerte Eigenschaften:

1. Sie existieren, wie Sie gesehen haben, unabhängig von allen anderen Boxen und werden von deren Fluss überhaupt nicht beeinflusst – völlig losgelöst, sozusagen.
2. Absolut positionierte Boxen können sich überlappen und haben deshalb eine zusätzliche Dimension, die mit der Eigenschaft `z-index` geregelt werden kann.

Sie wissen bereits, dass der Autor einer Webseite bei einer absolut positionierten Box selbst dafür verantwortlich ist, dass diese sich nicht unabsichtlich mit anderen ins Gehege kommt. Was hat es aber nun mit diesem `z-index` auf sich?

In mathematischen Diagrammen gibt es neben der x-Achse (die von links nach rechts verläuft) und der y-Achse (von oben nach unten) noch eine z-Achse, die sich von vorne nach hinten erstreckt, also bildlich gesprochen vom Betrachter aus ins Papier hinein. Positionierte Elemente können einen `z-index` bekommen, der die Position des Elements auf dieser z-Achse beschreibt. Ohne `z-index` liegen innere Elemente *vor* äußeren, weshalb zum Beispiel die Hintergrundfarbe von `#wrapper` die von `body` überdeckt.

Die Eigenschaft `z-index` hat zwei Besonderheiten:

- `z-index` gilt *nur* für positionierte Elemente.
- `z-index` ist *nur* relevant, wenn Elemente sich überlappen.

Je höher der Wert von `z-index` ist, desto dichter ist das Element am Leser.



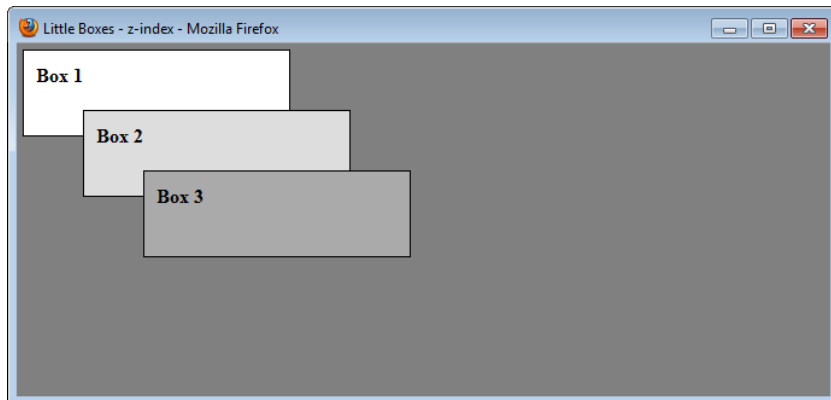
Hier sehen Sie ein Beispiel:

HTML	CSS
<pre> &lt;div id="eins"&gt;Box 1&lt;/div&gt; &lt;div id="zwei"&gt;Box 2&lt;/div&gt; &lt;div id="drei"&gt;Box 3&lt;/div&gt; </pre>	<pre> body {   color: black;   background-color: gray;   padding: 0;   margin: 0; } div {   position: absolute;   font-weight: bold;   width: 200px;   height: 50px;   padding: 10px;   border: 1px solid black;   margin: 5px; } #eins {   top: 0;   left: 0;   <b>z-index: 100;</b>   background-color: #fff; } #zwei {   top: 50px;   left: 50px;   <b>z-index: 200;</b>   background-color: #ddd; } #drei {   top: 100px;   left: 100px;   <b>z-index: 300;</b>   background-color: #aaa; } </pre>

Listing 14.4:  
HTML und CSS  
für die drei  
Beispielboxen

Im Browser sieht dieses Beispiel so aus:

Abbildung 14.10:  
»z-index« mit den  
Werten 100, 200  
und 300



Box 3 hat den höchsten z-index und ist dementsprechend am dichtesten am Leser. Die Werte für z-index können übrigens beliebige ganze Zahlen sein, sogar negative sind erlaubt. Anstelle von 100, 200 und 300 hätten 1, 2 und 3 im Beispiel denselben Effekt. Der Vorteil der Hunderter ist, dass man noch ein bisschen Raum hat, um später eventuell Elemente dazwischen zu positionieren.

## 14.8 Auf einen Blick

Hier sind noch einmal die wichtigsten Punkte dieses Kapitels im Überblick:

- Die HTML-Elemente einer Webseite folgen dem *document flow*, dem natürlichen Fluss des Dokuments:
  - Ein einzelnes Element beginnt so weit wie möglich links und oben.
  - Weitere Elemente werden rechts davon angeordnet, bis kein Platz mehr ist.
  - Falls kein Platz ist, rutschen sie eine Zeile tiefer und beginnen wieder links.

- Block-Elemente haben einen integrierten Zeilenumbruch und beanspruchen unabhängig von ihrer Breite immer eine ganze Zeile.
- Es gibt drei Werte zur CSS-Eigenschaft `position`, und zwar `relative`, `absolute` und `fixed`. Die genauen Koordinaten zur Positionierung werden jeweils mit den Eigenschaften `top`, `right`, `bottom` und `left` angegeben.
- Relative Positionierung verschiebt das Element relativ zu seiner ursprünglichen Position im *flow*. Die ursprüngliche Position bleibt geschützt.
- Absolute Positionierung nimmt das Element aus dem Fluss. Die Werte für `top`, `right`, `bottom` und `left` beziehen sich entweder auf das Stammelement `html` oder auf das nächste umgebende Element, das ebenfalls positioniert ist.
- Fixierte Elemente bleiben an ihrer Position stehen, scrollen nicht mit und sollten sehr vorsichtig eingesetzt werden.
- Der Internet Explorer 6 versteht `position: fixed` nicht und lässt fixierte Elemente normal weiterfließen.
- Positionierte Boxen können einen `z-index` bekommen, der ihre Position auf der z-Achse beschreibt.

