

Feste Positionierung

Feste Positionierung wird sich als Alternative zu Frames in der Zukunft immer mehr durchsetzen, wenn es darum geht, Bereiche so auf einer Seite zu positionieren, dass diese nicht mit dem Rest der Seite scrollen. Dieser Artikel beschreibt, wie Sie mit Hilfe der CSS-Eigenschaft `position:fixed` und anderen Techniken feste Positionierung trotz mangelhafter Browserunterstützung bereits heute korrekt einsetzen.

- [Grundlagen der Positionierung mit CSS](#)
- [Probleme mit festen Blöcken](#)
 - [Fehlerhafte Browserimplementierung](#)
 - [Feste Positionierung auch im IE](#)
 - [Probleme mit geringen Auflösungen](#)
 - [Probleme mit Ankern](#)
 - [Abneigung gegen fest positionierte Blöcke](#)
- [Fazit](#)

Grundlagen der Positionierung mit CSS

Mit der CSS-Eigenschaft `position` bestimmen Sie, auf welche Weise Boxen auf einer Website positioniert werden. Dabei werden folgende Positionierungsarten unterschieden:

`position:static`

Bei der Box handelt es sich um eine **normale Box gemäß dem gewöhnlichen Fluss**. `static` ist der Initialwert der Eigenschaft `position`, das bedeutet, dass jede Box, für die Sie keine andere Positionierungsart definieren, statisch positioniert ist.

`position:relative`

Die Box wird **relativ zu ihrer statischen Position verschoben**. Die Position der nachfolgenden Boxen wird so berechnet, als ob die Box nicht verschoben worden wäre.

`position:absolute`

Die Box wird **relativ zu ihrem umschließenden Block verschoben**. Dabei gilt folgendes zu beachten: Der umschließende Block für eine absolut positionierte Box wird durch das nächste positionierte Elternelement oder, falls es kein solches gibt, durch den umschließenden Ausgangsblock (also den Viewport) gebildet. Die positionierte Box wird völlig aus dem normalen Fluss entfernt, hat also keinen Einfluss auf spätere gleichrangige Elemente.

`position:fixed`

Die Box wird **absolut positioniert**, darüber hinaus ist die Box **feststehend hinsichtlich einer bestimmten Referenz**. Auf dem Screen bedeutet das, die Box bleibt beim Scrollen der Seite stehen, beim Druck wird das Element auf jeder Seite an der gleichen Stelle wiederholt.

Man bezeichnet ein Element (oder seine Box) als »positioniert«, wenn dessen Eigenschaft `position` einen anderen Wert als `static` annimmt. Wie weit die Inhaltskanten der positionierten Box von den Kanten des umschließenden Blocks entfernt sind, um also die genaue Position angeben zu können, gibt es die

Eigenschaften top, right, bottom und left.

Elemente lassen sich auch übereinander positionieren. Für gewöhnlich überlappen sie sich dann in der Reihenfolge, wie sie im XHTML-Quelltext vorkommen. Die Eigenschaft z-index erlaubt es, zu bestimmen, wie die Elemente übereinander liegen. Dabei überdeckt ein Element mit einem höheren Z-Index einen mit einem niedrigeren. Wird kein Z-Index explizit angegeben, hat das Element den Z-Index 0.

Folgendes Beispiel wird Ihnen die Anwendung der Eigenschaft position verdeutlichen.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
<head>
    <title>Grundlagen der Positionierung mit CSS</title>
    <style type="text/css">
        #box1, #box2, #box3, #box4 {
            width: 200px;
            height: 100px;
            padding: .5em;
            border: 2px solid black;
            color: black;
        }
        #box1 {
            background-color: #ffd;
            /* statisch positioniert */
        }
        #box2 {
            background-color: #ffb;
            /* relativ positioniert */
            position: relative;
            top: 50px;
            left: 50px;
        }
        #box3 {
            background-color: #ff6;
            /* absolut positioniert */
            position: absolute;
            top: 50px;
            left: 70px;
        }
        #box4 {
            background-color: #ff3;
            /* absolut positioniert */
            position: absolute;
            bottom: 25px;
            right: 25px;
        }
    </style>
</head>
```

```

</style>
</head>

<body>
  <div id="box1">Box 1</div>
  <div id="box2">
    Box 2
    <div id="box3">Box 3</div>
  </div>
  <div id="box4">Box 4</div>
</body>
</html>

```

Beispieldokument ansehen: [beispiel1.xhtml](#)

Entsprechend einfach lässt sich ein fest positionierter Block bilden, wie das zweite Beispiel zeigt. Wundern Sie sich allerdings nicht, falls der positionierte Block in Ihrem Browser nicht stehen bleibt. Ich werde noch erläutern, dass dies nicht unbedingt ungewöhnlich ist.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Grundlagen der Positionierung mit CSS</title>
  <style type="text/css">
    #box1 {
      width: 200px;
      height: 100px;
      border: 2px solid black;
      background-color: #ffd;
      /* fest positioniert */
      position: fixed;
      top: 10px;
      left: 20px;
    }
  </style>
</head>

<body>

  <div id="box1"></div>

  <p>Lorem ipsum dolor sit amet ...</p>
  <p>Lorem ipsum dolor sit amet ...</p>
  <p>Lorem ipsum dolor sit amet ...</p>
  <p>Lorem ipsum dolor sit amet ...</p>

```

```
<p>Lorem ipsum dolor sit amet ...</p>
```

```
</body>
```

```
</html>
```

Beispieldokument ansehen: [beispiel2.xhtml](#)

Probleme mit festen Blöcken

Fehlerhafte Browser-Implementierung

Leider ist die Browserunterstützung für `position: fixed` noch nicht ausreichend. Gegenwärtig unterstützen nur aktuelle Mozilla-Versionen (also auch Netscape > 6.0 und andere Browser wie Firebird, die sich der Mozilla-Engine bedienen), Opera ab Version 5, Konqueror 2.x und der IE 5.x für MacOS die feste Positionierung von Blöcken mit `position: fixed`. Die Interpretation Ihrer Positionierung kann daher sehr unterschiedlich ausfallen.

Zudem ist die Implementierung in einigen Browsern fehlerhaft. Der Internet Explorer 5.0 für MacOS hat einen grauenhaften Bug: Elemente können zwar fest positioniert werden, aber sobald die Seite gescrollt wird, scrollt der Focus der Links mit. Danach ist weder der Linktext noch die Box des Focusses als Link aktivierbar. Der IE 5.1.3 unter MacOS X hat das gleiche Problem, auch wenn hier zumindest die Links aktivierbar sind.

Ein ähnliches Verhalten ist auch beim Opera zu beobachten: Hier scrollt die Focus-Box ebenfalls mit der Seite weg. Die Links sind weiterhin anklickbar, jedoch werden Eigenschaften, die Sie für die Zustände `:hover` und `:active` für Anker deklariert haben, nur dann ausgeführt, wenn der Focus sich noch auf seinem Platz befindet. Auch Javascript-Bildwechsel funktionieren nur innerhalb dieser Focus-Box.

Zumindest für den Internet Explorer gibt es einen kleinen Workaround. Da der IE sowohl für MacOS als auch für Windows `position: fixed` falsch interpretiert, sollten Sie diese Deklaration vor ihm verstecken. Dazu können Sie dessen [Attribut-Selektor-Bug](http://w3development.de/css/hide_css_from_browsers/attribute/) [http://w3development.de/css/hide_css_from_browsers/attribute/] ausnutzen. Angenommen Sie möchten ein Navigationselement fest positionieren:

```
#navigation {
    position: absolute;
    top: 30px;
    left: 40px;
    width: 300px;
    height: 400px;
}

#navigation[id] {
    position: fixed;
}
```

Das Element, hier mit der ID `navigation`, wird zunächst absolut positioniert, würde also mit Rest der Seite wegschrollen. Für alle Browser, die den Attribut-Selektor korrekt interpretieren, überschreiben Sie `position: absolute` in `position: fixed`. Dem IE wird also die Deklaration, die er ohnehin nicht versteht, vorenthalten, sodass er das Element zumindest korrekt positioniert, während andere Browser wie Mozilla oder Opera das Element nun fest positionieren. Vergleichen Sie die Darstellung des zweiten Beispiel mit der Darstellung in [Beispiel 3](#), für das ich diesen Bug ausgenutzt habe. In diesem Beispiel können Sie auch beobachten, was mit der Focus-Box des Links im Opera passiert, wenn Sie ein wenig scrollen und dann mit der über den Beispiel-Link fahren.

Feste Positionierung auch im IE

Mit Hilfe des proprietären `expression()`-Features ist auch der Internet Explorer ab Version 5.0 in der Lage, Blöcke fest zu positionieren. Zwei Voraussetzungen müssen jedoch erfüllt sein:

- JavaScript (also Active Scripting) muss aktiviert sein.
- Das Dokument darf nicht im *Standard Mode*, sondern muss im *Quirks Mode* ausgeführt werden, was allerdings Nachteile mit sich bringt. Mehr dazu in meinem Artikel [Pinselführung - Doctype-Switching: Wie Browser über die Darstellung von \(X\)HTML entscheiden \[http://www.heise.de/ix/artikel/2004/03/136/\]](http://www.heise.de/ix/artikel/2004/03/136/).

Ich möchte diesen Workaround an folgendem Beispiel erläutern:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="de">
<head>
  <title>Das expression()-Feature des Microsoft Internet Explorers</title>

  <style type="text/css">
    #box1, #box2 {
      width: 200px;
      height: 100px;
      padding: .5em;
      border: 2px solid black;
      color: black;
    }
    #box1 {
      background-color: #ffe;
      position: absolute;
      top: 0;
      left: 0;
      position: expression("absolute");
      top: expression(document.body.scrollTop - this.offsetHeight + this.offsetHeight);
    }
    #box1[id] {
      position: fixed;
    }
    #box2 {
      background-color: #ffd;
      position: fixed;
      top: 100px;
      left: 300px;
      position: expression("absolute");
      top: expression(parseInt(document.body.scrollTop + 100));
    }
  </style>
</head>
```

```

        #box2[id] {
            position: fixed;
        }
    </style>
</head>

<body>

    <div id="box1">Box 1</div>
    <div id="box2">Box 2</div>

    <p>Lorem ipsum dolor sit amet ...</p>
    <p>Lorem ipsum dolor sit amet ...</p>
    <p>Lorem ipsum dolor sit amet ...</p>
    <p>Lorem ipsum dolor sit amet ...</p>
    <p>Lorem ipsum dolor sit amet ...</p>

</body>
</html>

```

Beispieldokument ansehen: [beispiel4.xhtml](#)

Expressions erlauben, einer CSS-Eigenschaft JavaScript-Ausdrücke zuzuweisen. Dies wird in dem Beispiel ausgenutzt, um die Position der beiden Boxen exakt festzulegen. Im ersten Beispiel wird das Element an den oberen Rand des Viewports positioniert, im zweiten Beispiel mit einem Abstand von 100 Pixel, entsprechend dem Wert der Eigenschaft top. Die Boxen bleiben beim Scrollen stehen.

Eine andere Möglichkeit, um feste Positionierung im Internet Explorer ab Version 5.0 zu ermöglichen, erläutert Fabrice Pascal in seinem Artikel [Position: fixed \(fast\) alle Browser? Es geht doch! \[http://www.fabrice-pascal.de/artikel/posfixedie6/\]](#).

Ich persönlich rate von der Verwendung derartiger »Tricks« ab, da sie immer irgendwelche Nachteile mit sich bringen.

Probleme mit geringen Auflösungen

Ein fest positionierter Bereich scrollt nicht mit der Seite mit. Wenn der Bereich so groß wird, dass er in einer geringen Auflösung (vor allem bei 640 mal 480 oder 800 mal 600 Bildpunkten) oder bei einer entsprechend kleinen Fenstergröße nicht mehr in das Browserfenster passt, ist er nicht mehr zugänglich. Sie sollten den festen Bereich also entweder auf eine geringe Fläche beschränken oder darauf verzichten. Alternativ bietet sich die Anwendung der Eigenschaft `overflow` an.

overflow: visible

Der Inhalt der Box wird nicht beschnitten, er könnte also aus der Box herausfließen, wenn die festgelegten Dimensionen der Box nicht ausreichen.

overflow: hidden

Der Inhalt der Box wird beschnitten. Der User hat keine Möglichkeit, auf den abgeschnittenen Inhalt zuzugreifen.

overflow: scroll

Es werden Scroll-Mechanismen zur Verfügung gestellt, um den Inhalt der Box zugänglich zu machen, falls er beschnitten wird.

overflow:auto

Für gewöhnlich bieten User-Agents hierbei Scroll-Mechanismen an, falls diese notwendig werden.

Jedoch haben vor allem ältere Browser mit der Eigenschaft `overflow` große Probleme. Anhand von [Beispiel 5](#) können Sie ihren Browser testen.

Probleme mit Ankern

Wenn Sie eine umfangreiche Webseite geschrieben haben, mag es manchmal sinnvoll sein, Querverweise innerhalb dieser Webseite zu setzen. Oder Sie möchten von einer Quelle aus in die Mitte einer anderen Webseite verweisen. Dies bewerkstelligen Sie mit Ankern innerhalb des Dokumentes. Wird auf einen solchen Anker verwiesen, springt der Browser zu dieser Stelle, sodass diese direkt oben auf der Seite angezeigt wird.

Haben Sie, wie auch auf dieser Seite zu sehen, einen fest positionierten Bereich oben auf der Seite, der sich in die Breite erstreckt, ist es nicht möglich, mit Ankern zu arbeiten. Die von mir getesteten Mozilla und Opera springen zum Anker, stellen diesen auch ganz oben auf der Seite dar, jedoch hinter dem fest positionierten Bereich. [Beispiel 6](#) verdeutlicht das Problem. Eine gute Lösung, dieses Verhalten zu umgehen, ist mir bisher nicht bekannt. Mathias Schäfer listet auf der Seite [Probleme mit Ankern und position:fixed \(feste CSS-Positionierung\) \[http://molily.de/css-position-fixed\]](http://molily.de/css-position-fixed) mögliche Lösungsansätze auf.

Abneigung gegen fest positionierte Blöcke

Der Vorteil von fest positionierten Blöcken ist, dass sie sich ständig in Sicht- und Klick-Weite des Nutzers befinden. Allerdings gibt es vermehrt Nutzer, die fest positionierte Bereiche als Einschränkung empfinden. Schließlich könne der Platz, der zur Darstellung der Navigation verschwendet wird, viel besser zur Darstellung der Information verwendet werden. Gerade auf kleinen Bildschirmen wird dieser Nachteil deutlich. Leider ist es nicht möglich, `position:fixed` gezielt per User-Stylesheet zu überschreiben. Diesen Einwand sollten Sie beachten, wenn Sie mit fest positionierten Bereichen arbeiten. Sorgen Sie also dafür, dass diese Bereiche nicht zu viel Platz einnehmen!

Fazit

Trotz aller Probleme bietet `position:fixed` Webautoren eine gute Möglichkeit, Inhalte ständig im Blickwinkel des Benutzers bereit zu stellen, zum Beispiel den Navigationsbereich der Seite, und ist somit eine gute Alternative zu Frames. Die schlechte Browserunterstützung sollte Autoren nicht davon abhalten, feste Positionierung **sinnvoll** und auf eine Art und Weise einzusetzen, dass die Zugänglichkeit der Informationen nicht beeinträchtigt wird.