


Zeitschriftenartikel*Begutachtet***Begutachtet:**

Dr. Lutz Gollan 
Landesbetrieb Verkehr
Hamburg
Deutschland

Erhalten: 31. Mai 2020**Akzeptiert:** 04. Juni 2020**Publiziert:** 30. Juni 2020**Copyright:**

© Dr.-Ing. Maika Büschenfeldt.
Dieses Werk steht unter der Lizenz
Creative Commons Namens-
nennung 4.0 International (CC BY 4.0).

**Empfohlene Zitierung:**

BÜSCHENFELDT, Maika, 2020:
Animationen mit CSS: Ein kleines
Tutorial für Animationen mit CSS 3.
In: *API Magazin* 1(2) [Online]
Verfügbar unter: [DOI 10.15460/
apimagazin.2020.1.2.40](https://doi.org/10.15460/apimagazin.2020.1.2.40)

Animationen mit CSS

Ein kleines Tutorial für Animationen mit CSS 3

Dr.-Ing. Maika Büschenfeldt^{1*} ¹ Hochschule für Angewandte Wissenschaften, Hamburg, Deutschland

Wissenschaftliche Mitarbeiterin am Department Information

* Korrespondenz: redaktion-api@haw-hamburg.de

Zusammenfassung

Der kleine Workshop bringt mit CSS 3-Animationen ein wenig Bewegung in HTML-Seiten. Es werden die Animationsarten Transitions und Keyframe-Animationen vorgestellt. Das Tutorial setzt Grundkenntnisse in HTML und CSS voraus.

Schlagwörter: CSS 3, HTML, Webseite, Animation

Abstract

This short workshop uses CSS 3 animations to bring a little movement to HTML pages. The animation types transitions and keyframe animations are introduced. This tutorial requires basic knowledge of HTML and CSS.

Keywords: CSS 3, HTML, Website, Animation

1 Allgemeines

In diesem kleinen Workshop werden wir mit CSS 3-Animationen ein wenig Bewegung in unsere HTML-Seiten bringen. Animation soll hier heißen, dass wir HTML-Elemente in einem festgelegten Zeitraum mittels CSS (Cascading Style Sheets) verändern. Eine solche Veränderung würde beispielsweise bedeuten, dass sich eine Überschrift von A nach B bewegt und dabei ihre Farbe und ihre Größe verändert. CSS 3-Animationen haben – wie so vieles – Vor- und Nachteile:

Vorteile:

- CSS-Animationen können in eine CSS-Datei ausgelagert werden und sind damit mehrfach verwendbar.
- CSS-Animationen sind hardwarebeschleunigt, sie laufen deshalb sehr schnell, glatt und ruckelfrei.
- CSS-Animationen sind einfach und auch ohne JavaScript-Kenntnisse zu erstellen. Wer sich dennoch an JavaScript herantraut, sollte wissen, dass CSS-Animationen im Zusammenwirken mit der JavaScript Animations-API zu einem sehr mächtigen und schnellen Animationswerkzeug werden können.

Nachteil:

- CSS-Animationen können nur eingeschränkt eingesetzt werden. Sie sind Bestandteil von CSS 3 und immer noch eine experimentelle Technologie. Der Einsatz bei älteren Browsern ist deshalb schwierig und es ist auch nicht auszuschließen, dass sich die Syntax im Zuge der Weiterentwicklung ändert.

2 Vorbereitungen

Bevor wir beginnen, legen wir die Dateien für ein kleines HTML-Projekt an. Wir benötigen zunächst eine HTML-Datei und eine CSS-Datei für die ausgelagerten CSS Style-Anweisungen.

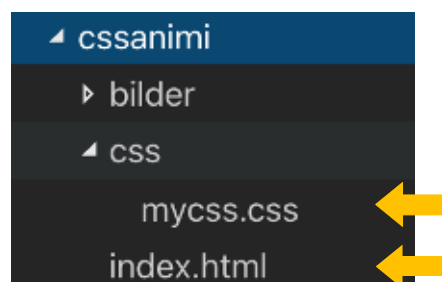


Abb. 1: Verzeichnisstruktur

Im HTML-Code (index.html) binden wir die Stylesheet-Datei mycss.css ein und fügen als erstes Element einen leeren <div>-Container hinzu.

```
<!DOCTYPE html>
<html lang="de">
<head>
  <title>HTML - Der Einstieg</title>
  <meta charset="utf-8">
  <link rel="StyleSheet" href="css/mycss.css"
    type="text/css" media="screen" />
</head>
<body>
  <div id="box1"></div>
</body>
</html>
```

In der CSS-Datei (mycss.css) definieren wir die CSS-Eigenschaften für den <div> Container. Wir verwenden den ID-Selektor für "box1".

```
#box1{
  height: 100px;
  width: 100px;
  background: orangered;
}
```

Im Ergebnis erhalten wir ein orangerotes Quadrat.



Abb. 2: Ein Quadrat mit CSS zeichnen

3 Transitions

CSS 3 bietet zwei Animationsarten an: Transitions und Keyframe-Animationen. Beide Animationsarten sind einander ähnlich. Transitions sind jedoch einfacher anzuwenden, dafür sind sie in ihren Möglichkeiten etwas eingeschränkter. Eine Transition ist der Übergang von einem Anfangszustand in einen Endzustand. In unserem Beispiel ist das orangerote Quadrat der Ausgangszustand.

```
#box1{
  display: block;
  height: 100px;
  width: 100px;
  background: orangered;
  transition-property: background-color width;
  transition-duration: 5s;
  transition-delay: .5s;
}
```

Für die Definition einer Transition benötigen wir mindestens zwei Angaben:

`transition-property` gibt an, welche CSS-Eigenschaften verändert werden sollen.

`transition-duration` gibt an, wie lange die Animation dauert.

`transition-delay` gibt an, wieviel Zeit vergeht, bis die Animation startet.

`transition-timing-function` steuert den Ablauf der Animation.

Folgende Angaben sind möglich:

<code>linear</code>	Gleichbleibende Geschwindigkeit (Standardeinstellung).
<code>ease</code>	Langsamer Beginn, dann schnelles und langsames Ende.
<code>ease-in</code>	Langsamer Beginn, schnelles Ende.
<code>ease-out</code>	Schneller Start, langsames Ende.
<code>ease-in-out</code>	Langsamer Start und langsames Ende.



Den Endzustand definieren Pseudoklassen. Die Art der Pseudoklasse gibt an, welches Ereignis die Animation auslöst:

```
#box1:hover{
  height: 100px;
  width: 150px;
  background: blue;
}
```

<code>:active</code>	Anklicken.
<code>:hover</code>	Mit der Maus überfahren.

Ergebnis:

Beim Überfahren mit der Maus wechselt das Quadrat seine Farbe und wird breiter. Aus einem orangeroten Quadrat wird ein blaues Rechteck. Dieser Vorgang dauert fünf Sekunden. Wenn die Maus das Element wieder verlässt, verwandelt sich das blaue Rechteck zurück in ein rotes Quadrat.

Anfangszustand	Endzustand
	

4 Keyframe-Animation

Wie bei den Transitions geht es bei Keyframe-Animationen um animierte Übergänge zwischen unterschiedlichen Zuständen der CSS-Styles. Im Unterschied zu den Transitions läuft die Keyframe-Animation nicht nur von einem Anfangs- in einen Endzustand, sondern über mehrere Schlüsselbilder. Jedes Schlüsselbild repräsentiert einen Zwischenstand der Animation. Das ermöglicht vielschichtige Veränderungen, macht den Quellcode aber umfangreicher.

Um eine Animation zu erstellen, fügen wir dem zu animierenden Element die Eigenschaft `animation` hinzu und ergänzen diese durch eine Reihe von Sub-Eigenschaften. Sub-Eigenschaften beschreiben den Style der Animation. Die Tabelle zeigt eine Übersicht der animation Sub-Eigenschaften:

Tab. 1: Eigenschaften für CSS-Animationen

Eigenschaft	Erläuterung
<code>animation-name</code>	Legt den Namen für die <code>@keyframes</code> Regel fest.
<code>animation-duration</code>	Legt die Dauer einer Animation in Sekunden fest.
<code>animation-timing-function</code>	Gibt an, wie eine CSS-Animation über die Dauer eines Zyklus (Übergang) verlaufen soll. Mögliche Werte:
<code>ease</code>	Langsamer Start, beschleunigt und endet langsam (Standardwert) \Leftrightarrow cubic-bezier(0.25,0.1,0.25,1)
<code>linear</code>	Gleiche Geschwindigkeit \Leftrightarrow cubic-bezier(0,0,0.1,1)
<code>ease-in</code>	Langsamer Start \Leftrightarrow cubic-bezier(0.42,0,1,1)
<code>ease-in-out</code>	Langsamer Start und Ende \Leftrightarrow cubic-bezier(0.42,0,0.58,1)
<code>cubic-bezier(n,n,n,n)</code>	Erlaubt individuelle Einstellungen zum Ablauf der Animation. Eingabe numerischer Werte (auch Fließkommazahlen) zwischen 0 und 1 für die Punkte: (Start, Beschleunigung, Beschleunigung, Stopp)
<code>initial</code>	Werte werden auf den Standardwert zurückgesetzt.
<code>inherit</code>	Werte werden vom Elternelement übernommen
<code>animation-delay</code>	Gibt in Sekunden an, wann die Animation nach dem vollständigen Laden der Seite starten soll. Der Standardwert ist 0s.
<code>animation-iteration-count</code>	Legt die Anzahl Wiederholungen der Animation fest. Als Wert wird ein Ganzzahlwert für die Anzahl der Wiederholungen angegeben. Bei Angabe des Schlüsselwertes <code>infinite</code> läuft die Animation als Endlosschleife.
<code>animation-direction</code>	Legt die Anzahl Wiederholungen der Animation fest. Als Wert wird ein Ganzzahlwert für die Anzahl der Wiederholungen angegeben. Bei Angabe des Schlüsselwertes <code>infinite</code> läuft die Animation als Endlosschleife.

Wir verwenden die Eigenschaft `animations`, um einen Ladebalken zu erzeugen. Für die Darstellung des Ladebalkens erzeugen wir einen leeren `<div>`-Container mit der id „ladebalken“ im HTML Dokument.

```
<div id="ladebalken"></div>
```

Den Anfangszustand der `<div>`-Elemente und den Animationsablauf definieren wir in der CSS-Datei.

```
#ladebalken{
  display: block;
  background: darkgreen;
  height: 20px;
  width: 350px;
  border: 1px solid darkgray;
  animation-name: fortschrittsbalken;
  animation-duration: 4s;
  animation-direction: alternate;
  animation-timing-function: linear;
  animation-iteration-count: infinite;
}
```

Für die Definition des Animationsablaufs wählen wir den ID-Selektor `#ladebalken`. Dabei legen wir die folgenden Details des Animationsablaufs fest:

Tab. 2: Animationsablauf Fortschrittsbalken

<code>animation-name: fortschrittsbalken;</code>	Name der CSS-Definition, referenziert auf die Keyframes.
<code>animation-duration: 4s;</code>	Die Animation beginnt nach 4 Sekunden.
<code>animation-direction: alternate;</code>	Die Animation läuft abwechselnd vorwärts und rückwärts.
<code>transform-origin: 50% 50%</code>	Die Rotation der Animation wird mittig ausgerichtet.
<code>animation-iteration-count: infinite;</code>	Die Animation läuft endlos.

Während wir über die Animation-, Transform- und Transition-Eigenschaften die Details des Animationsablaufes bestimmen, erfolgt die eigentliche Darstellung der Animation durch die sogenannten `@keyframes`.

Syntax eines Keyframes:

```
@keyframes <bezeichner> {
  [ [ from | to | <Prozentzahl> ] [, from | to | <Prozentzahl> ]* block ]*
}
```

In unserem Beispiel wählen wir für die Keyframes die Prozentzahlen 0% bis 100%. Für jede Prozentzahl legen wir die CSS-Eigenschaften für die Länge des Balkens und die Farbe fest:

- Der Balken startet mit einer Länge von 20px und der Farbe darkgreen.
- Der Balken endet mit einer Länge von 200px und der Farbe lime.

Im Ergebnis wird der Balken nicht nur länger und wechselnd wieder kürzer, sondern zeigt auch einen Farbverlaufseffekt.

```
@keyframes fortschrittsbalken {
  0% {
    fill: darkgreen;
    width: 20px;
  }
  100% {
    fill: lime;
    width: 200px;
  }
}
```

Tip: Bei den Prozentangaben können auch mehrere Angaben gemacht werden. Zum Beispiel: 50% {fill: orange; width: 90px;}

5 Die Sub-Klasse „Transform“

Animationen werden ebenfalls durch die Eigenschaft Transform unterstützt. Die bereits bekannte Eigenschaft Transition unterstützt effektvolle Übergänge.

Tab. 3: Eigenschaften der Sub-Klasse Transform

Eigenschaft	Erläuterung
<code>transform: rotate(Grad deg)</code>	Rotation, Angabe der Gradzahl
<code>transform-origin</code>	Änderung der Position von transformierten Elementen. Werte können in Prozent angegeben werden. Standardwert ist: 50% 50% 0
Transition ändert den Wert einer CSS-Eigenschaft über die Zeit. Das ermöglicht Übergänge wie das Ein- und Ausblenden oder das Bewegen von HTML-Elementen.	

Im nächsten Schritt werden wir das orangefarbene Quadrat um die eigene Achse rotieren lassen. Dazu ergänzen wir die Eigenschaften der ID #box1 (siehe Abschnitt 2 und 3).

```
#box1{
  height: 100px;
  width: 100px;
  background: orangered;
  transition-property: background-color width;
  transition-duration: 7s;
  transition-delay: .5s;
  animation-name: rotatebox;
  animation-duration: 3s;
  animation-iteration-count: infinite;
  transform-origin: 50% 50%;
}
```

In dieser CSS-Anweisung stehen zunächst die Eigenschaften für den Animationsablauf.

Tab. 4: Animationsablauf für ein rotierendes Quadrat

<code>animation-name: rotatebox;</code>	Name der CSS-Definition, referenziert auf die Keyframes.
<code>animation-duration: 10s;</code>	Die Animation beginnt nach 10 Sekunden.
<code>animation-iteration-count: infinite;</code>	Die Animation läuft endlos.
<code>transform-origin: 50% 50%;</code>	Die Rotationsachse wird mittig ausgerichtet.

Für die Definition der Keyframes verwenden wir die Schlüsselwörter `from` und `to`:

```
@keyframes rotatebox {
  from { transform: rotate(0deg); }
  to   { transform: rotate(360deg); }
}
```

`from`: Für den Ausgangspunkt wird mit der Anweisung `transform: rotate(0deg)` ein Rotationswinkel von 0 Grad festgelegt.

`to`: Für den Endpunkt der Animation wird mit der Anweisung `transform: rotate(360deg)` ein Rotationswinkel von 360 Grad festgelegt.

Im Ergebnis dreht sich das rote Quadrat um die eigene Achse. Der in Abschnitt 3 entwickelte Hover-Effekt bleibt bestehen.

Weiterführende Links

Das aus der animation-timing-function resultierende Verhalten wird sehr schön in einer Beispiel-Animation der Plattform selfhtml (FRICKL) dargestellt.

https://wiki.selfhtml.org/extensions/Selfhtml/frickl.php/Beispiel:CSS3_animation-3a.html#view_result

Hinweis zur Verwendung von transition-property: Die Spezifikationen des W3C (World Wide Web Consortium) und eine Liste mit 50 animierbaren CSS-Eigenschaften können hier nachgeschlagen werden:

<https://www.w3.org/TR/css-transitions-1/#animatable-css>

CSS-Animationen im SELFHTML-Wiki:

<https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Animation>

Im SELFHTML-Wiki finden sich weitere hilfreiche Tutorials rund um CSS:

<https://wiki.selfhtml.org/wiki/CSS/Tutorials>