Animation mit CSS

Zwei Typen von Animationen

Grundsätzlich lassen sich in CSS zwei Typen von Animationen definieren: nicht Interaktive (**CSS-Animation**) und Interaktive (**CSS-Transitions**). CSS-Animationen starten automatisch, CSS-Transitions sind abhängig vom Zustand eines Elements, siehe «**:hover und :active**».

Was lässt sich animieren?

Faustregel 1: Es lassen sich alle Eigenschaften animieren, die mit numerischen Werten definiert werden.
Faustregel 2: Interpolieren von Position, Grösse, Drehung und Deckkraft verbraucht wenig Rechenleistung:

```
transform: translate(npx, npx); /* Position, Pixelwert  */
transform: scale(n);            /* Grösse, Faktor       */
transform: rotate(ndeg);        /* Rotation, Winkel     */
opacity:   n;                   /* Deckkraft, 0-1       */
```

CSS-Animation

Die CSS Eigenschaft **animation** fasst folgende Eigenschaften zusammen:

animation-name	[Referenz für **@keyframes**, siehe unten]
animation-duration	[Sekunden oder Millisekunden]
animation-timing-function	[linear, ease, ease-in, ease-in-out, ease-out]
animation-delay	[Sekunden oder Millisekunden]
animation-iteration-count	[0-infinite]
animation-direction	[normal, reverse, alternate, alternate-reverse]

@keyframes

In der CSS-Regel **@keyframes**, gefolgt vom Namen einer Animation, werden die Werte definiert, zwischen denen interpoliert wird: Start, Schluss und bei Bedarf Zwischenschritte. Erlaubt sind Angaben in Prozenten von 0-100, dazu **from** und **to** für Anfang und Ende der Animation.

```
@keyframes grow {
     from {transform: scale(1);}    /* statt 'from' ginge auch '0%' */
     to   {transform: scale(2);}    /* statt 'to' ginge auch '100%' */
}

.cas-dt {
     animation: grow 2s ease 0 infinite alternate;
}
```

Anmerkungen

Die Reihenfolge der Werte ist nur teilweise geregelt. Als erstes muss der Name aufgeführt werden. Falls Dauer und Verzögerung definiert werden, muss die Dauer zuerst angegeben werden.
Für ältere Browser-Versionen sind Vendor-Prefixes notwendig:

```
-webkit-animation {}  @-webkit-keyframes {}   /* Chrome/Safari     */
-moz-animation    {}  @-moz-keyframes    {}   /* Firefox           */
-ms-animation     {}  @-ms-keyframes     {}   /* Internet Explorer */
-o-animation      {}  @-o-keyframes      {}   /* Opera             */
```

CSS-Transition

Die CSS Eigenschaft **transition** fasst folgende Eigenschaften zusammen:

transition-property	[Eine CSS-Eigenschaft]
transition-duration	[Sekunden oder Millisekunden]
transition-timing-function	[linear, ease, ease-in, ease-in-out, ease-out]
transition-delay	[Sekunden oder Millisekunden]

```
.cas-dt {
     color: black;
     opacity: 0.5;
     transition: opacity 0.5s ease-out 0;
}

.cas-dt:hover {
     opacity: 1;
}
```

Timing-Function / Ease

Eine Bewegung muss nicht mit konstanter Geschwindigkeit ablaufen. Die verschiedenen Varianten von «ease»-Werten entsprechen kurvenförmigen Beschleunigungswerten über die Dauer einer Animation.

linear	[konstante Geschwindigkeit]
ease	[langsamer Start, langsames Ende]
ease-in	[langsamer Start, Beschleuigung am Ende]
ease-out	[Schneller Start, Verlangsamung gegen Ende]
ease-in-out	[langsamer Start, langsames Ende (Variante)]
cubic-bezier(n,n,n,n)	[Eigene Kurve]

:hover und :active

Die CSS Pseudoklassen **:hover** und **:active** bezeichnen Zustände eines HTML-Elements. **:hover** bezeichnet ein Element, über dem der Cursor schwebt, **:active** steht für ein Element, das angeklickt wird (Maustaste gedrückt). Ein Problem im Zusammenhang mit CSS-Transitions stellt sich bei Touch-Devices. Das Tippen mit dem Finger entspricht dem Klick mit der Maus oder dem Trackpad. Die Pseudoklasse **:hover** existiert auf Tablets und Smartphones nicht: es gibt nur **:active**.

Nun lässt sich mit JavaScript erreichen, dass das einmalige Antippen eines Elements als **:hover** interpretiert wird, und erst das zweite Tippen als **:active**.

Mit JavaScript lässt sich nicht nur beliebig manipulieren, wie der Browser Quelltext verarbeitet, sondern auch wie er Interaktionen prozessiert. Es stellt sich natürlich die Frage, wie sinnvoll es ist, eine Art der Interaktion, die auf den Eigenschaften des Cursors basiert, in eine Umgebung ohne Cursor zu Übersetzen.