Technology
Arts Sciences
TH Köln

Animierte SVGs – Workshop

Lioba Brandhoff – Webtechnologien – TH Köln

28.05.2024

Fragerunde

- Wer weiß was SVGs sind?
- Wer hat schon mit SVGs im Web gearbeitet?

- Wer weiß was animierte SVGs sind?
- Wer hat schon SVGs im Web animiert?

- Welche Methoden kennt ihr, um SVGs zu animieren?

Inhalt

• Was ist ein SVG?	**~10 Minuten**
• Animationsgrundlagen	**~10 Minuten**
• Überblick über alle Möglichkeiten	
• SMIL-Animation	**~5-10 Minuten**
• CSS-Animationen (Transitions)	**~10 Minuten**
Aufgabe 1	**~10-15 Minuten**
Pause	**~10 Minuten**
• CSS-Animationen (Keyframes)	**~15 Minuten**
Aufgabe 2	**~10-15 Minuten**
• JavaScript Animation	**~10 Minuten**
Aufgabe 3	**~20 Minuten**
• JSON-Animation	**~10-15 Minuten**
• Zusammenfassung und Ausblick	**~5-10 Minuten**

Was ist ein SVG?

- SVG = Scalable Vector Graphic

- Vektorgrafikformat auf XML-Basis (Extensible Markup Language)
- Vektor = eine Linie zwischen zwei Punkten
- Vektorgrafik besteht aus Pfaden und geometrischen Formen
  → Keine Pixel !
- 2D-Format
- Alle wichtigen Webbrowser unterstützen SVG-Rendering

- 2001 vom World Wide Web Consortium (W3C) entwickelt
- W3C = Gremium zur Standardisierung der Techniken im World Wide Web

- Bewegungen/Animationen werden immer beliebter im Web
  → Bewegungen machen die Internetseite dynamischer

Quelle: https://www.luke-media.de/blog/svg-animationen

Warum verwenden wir keine GIFs oder MP4?

GIF

- große Dateien → lange Ladezeiten
- Pixelbasiert → bei Vergrößerung schlechte Auflösung
- Animation wird in der Datei gespeichert (keine Änderungen möglich)

MP4

- große Dateien → lange Ladezeiten
- Pixelbasiert → bei Vergrößerung schlechte Auflösung
- Rechteckiges Format (keine Transparenzen)

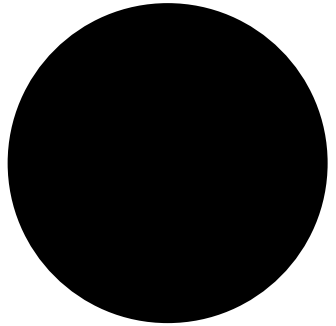
Vorteile:

- MP4 hat auch Audio → geht weder bei GIFs noch bei SVGs
- Es können Fotos und Screenshots angezeigt werden → bei SVGs nicht möglich

Quelle: https://www.pixx.io/blog/animierte-bilder-erstellen

Beispiel SVGs

```
<svg width="200px" height="200px">
    <circle cx="100px" cy="100px" r="100px"></circle>
</svg>
```

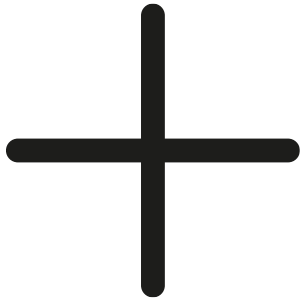


```
<svg width="200" height="200">
    <rect width="100%" height="100%" fill="red"></rect>
</svg>
```

Quelle: https://www.mlte.de/artikel/svg/

Beispiel SVGs

```
<svg id="icon-plus" viewBox="0 0 148 148" width="200" height="200">
    <defs>
        <style>
            .cls-1{
                fill:none;
                stroke:#1d1d1d;
                stroke-linecap:round;
                stroke-miterlimit:10;
                stroke-width:12px}
        </style>
    </defs>
    <g id="Ebene_1-2">
        <path class="cls-1" d="M6 74.03h136.06M74.03 6v136.06"/>
    </g>
</svg>
```



Beispiel SVGs

```
<svg id="icon-ball-rolling" width="200" height="200" viewBox="0 0 1200 1200">
    <path d="M600 1200.66c76.14 0 150.11-14.11 218.94-41.06a590.043 590.043 0
            0 0 37.2-16.02c61.65-29.14 118.47-69.01 168.12-118.66 99.98-99.98
            160.3-229.04 173.13-368.1 1.71-18.56 2.61-37.29 2.61-56.16
            0-43.97-4.73-87.21-13.88-129.16-24.22-110.96-79.62-212.87-161.
            85-295.11C910.94 63.07 760.27.66 600 .66S289.06 63.07 175.74
            176.39C62.41 289.72 0 440.39 0 600.66s62.41 310.94 175.74
            424.26c113.33 113.33 264 175.74 424.26 175.74Zm0-1135.27c8.04 0
            16.05.2 24.04.55l-48.72 56.11c-128.4-.05-225.52 59.51-258.28
            82.56l-68.47-7.69C346.01 111.86 469.43 65.4 600 65.4Zm378.49
            156.77c64.49 64.49 110.46 142.56 135.33 227.74l-66.31-19.81c-14.
            42-44.49-45.71-93.45-93.12-145.68-33.42-36.81-63.42-62.47-70.
            37-68.28l-8.28-74.5c36.95 22.29 71.46 49.22 102.76 80.53Zm-29.75
            784.6-239.57-13.25-55-145.63 148.57-164.15c12.29-3.22 136.72-35.
            88 188.38-50.53l90.58 201.39c-25.75 52.9-60.39 101.73-103.21
            144.56a547.712 547.712 0 0 1-29.75 27.61ZM321.36 …. 50.81Z"
            style="fill:#000;stroke-width:0"/>
</svg>
```

Quelle (Ball SVG): https://de.freepik.com/vektoren-kostenlos/sport-baelle-sammlung_943117.htm#fromView=search&page=1&position=22&uuid=03c304e1-68ef-4d3a-b447-a3ac5ae9b8fb

Animationsgrundlagen

Easing

Duration

Delay

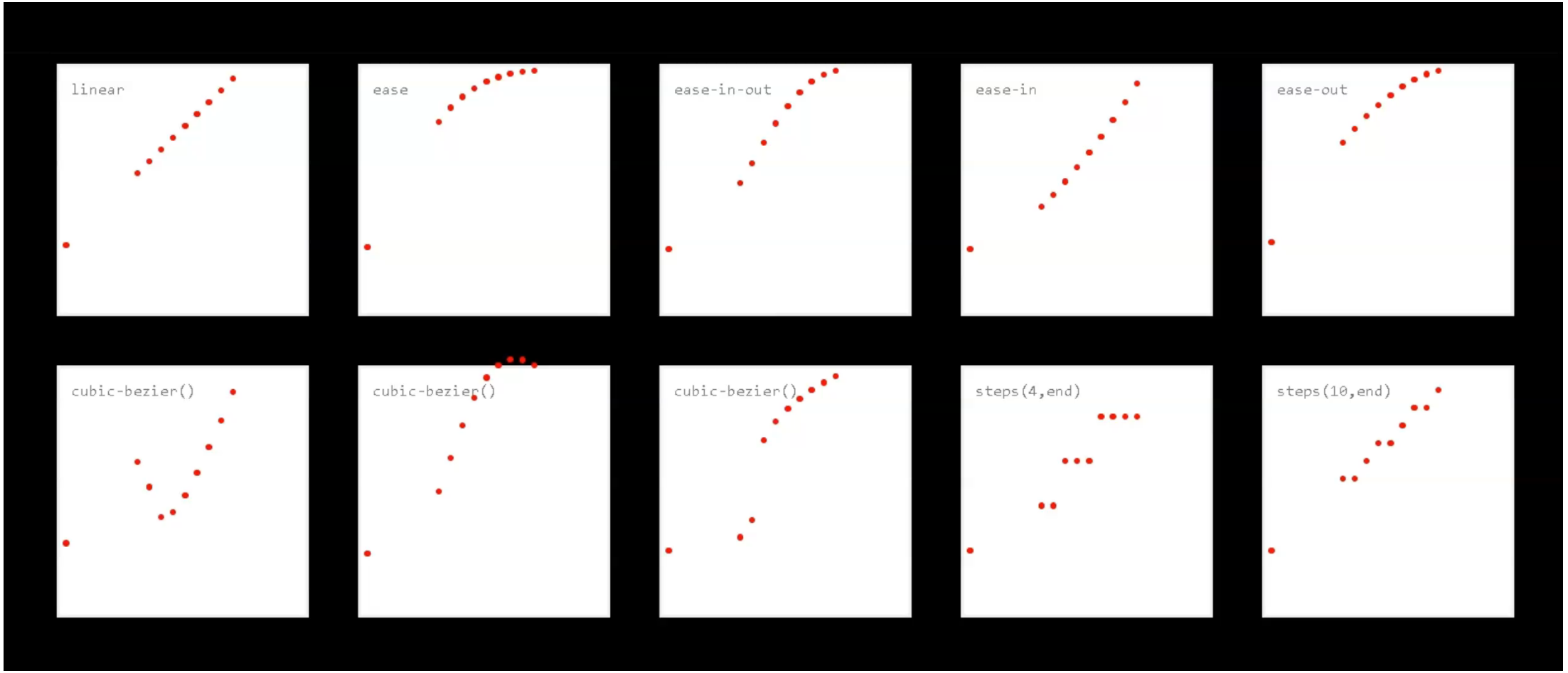
Start- & Endpunkt (Keyframes)

Animationsgrundlagen - Easing

linear	Gleichbleibende Geschwindigkeit
ease	Die Animation beginnt langsam, wird schnell und endet langsam.
ease-in	Die Animation beginnt langsam, endet schnell.
ease-out	Die Animation startet schnell, endet langsam.
ease-in-out	Die Animation beginnt langsam, wird schnell und endet wieder langsam.
cubic-bezier (x1,y1,x2,y2)	Die Animation wird nach eigenen Angaben gesteuert. Die ersten zwei Werte bestimmen die Startgeschwindigkeit, der dritte und vierte Wert die Endgeschwindigkeit.
steps (nr, [start|end])	Die Animation hat einen schrittweisen Übergang. Der erste Wert bestimmt die Anzahl an Schritten. Der zweite Parameter hat die möglichen Werte: start oder end. Damit wird bestimmt, ob ein Schritt vor oder nach einem Zeitintervall dargestellt werden soll.

Quelle: https://www.webdesign-journal.de/css3-animationen-erstellen/

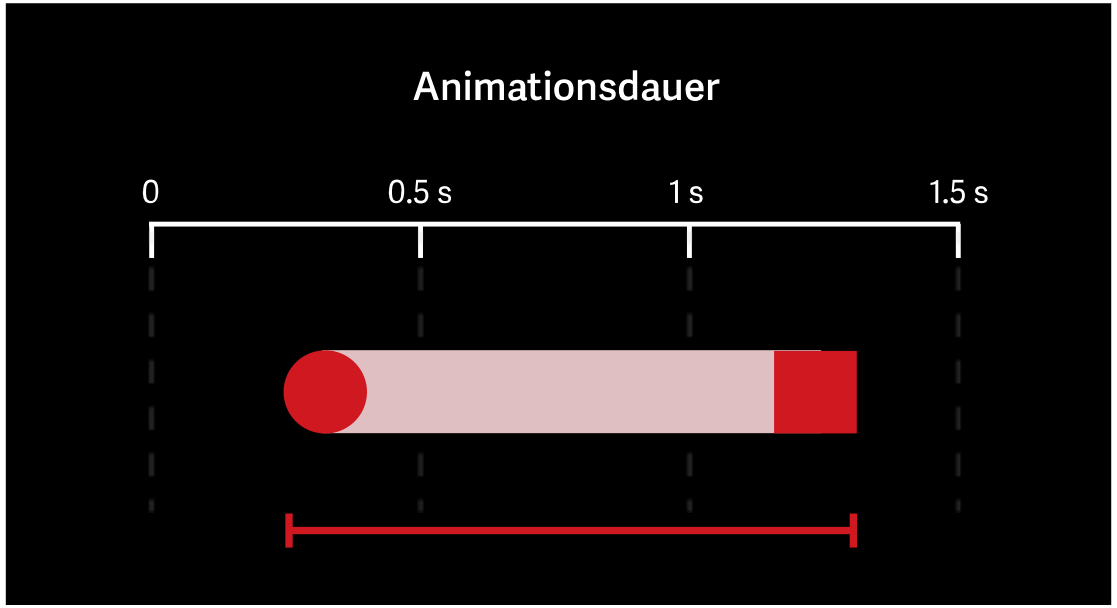
Animationsgrundlagen - Easing



Quelle: https://www.webdesign-journal.de/css3-animationen-erstellen/

Animationsgrundlagen - Duration

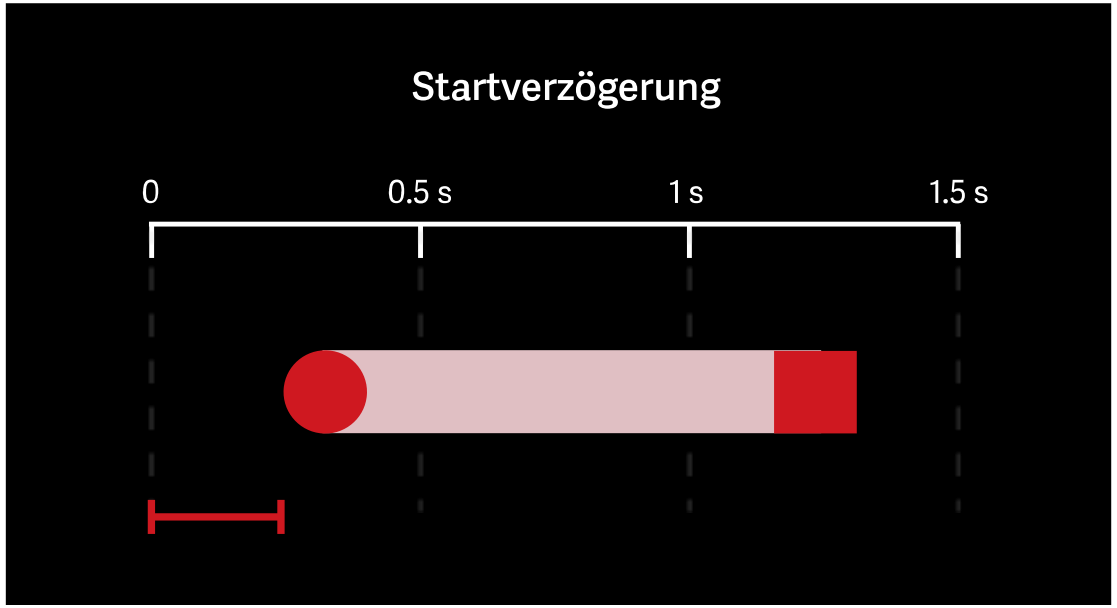
- Duration ist die Dauer einer Animation
- Wird in Sekunden angegeben (bzw. bei JavaScript in Millisekunden)



Quelle: https://www.webdesign-journal.de/css3-animationen-erstellen/

Animationsgrundlagen – Delay

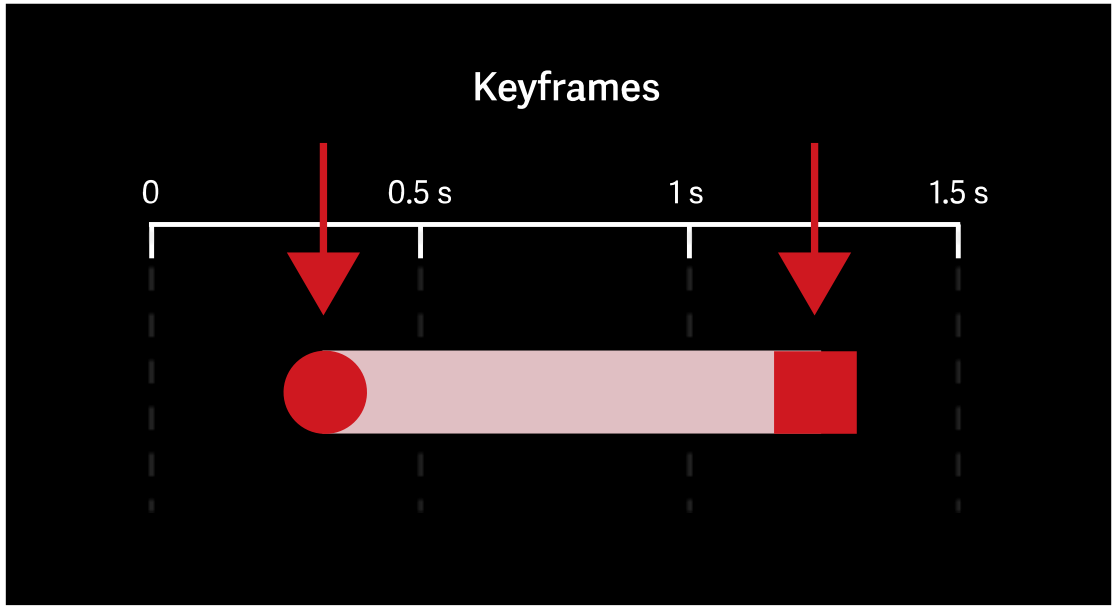
- Delay ist die Verzögerung der Animation
- gibt an, wann die Animation beginnt
- Werte sind 0, positive und negative Zahlen
- Wird in Sekunden angegeben (bzw. bei JavaScript in Millisekunden)



Quelle: https://www.webdesign-journal.de/css3-animationen-erstellen/

Animationsgrundlagen - Keyframes

- Keyframe = „Schlüsselbild“
- sind die Markierungen, die den Start- und Endpunkt einer Aktion festlegen
- Vorher-/Nachher-Zustand



Quelle: https://www.webdesign-journal.de/css3-animationen-erstellen/

Möglichkeiten der Animation eines SVGs

SMIL

CSS

JavaScript

JSON

SMIL

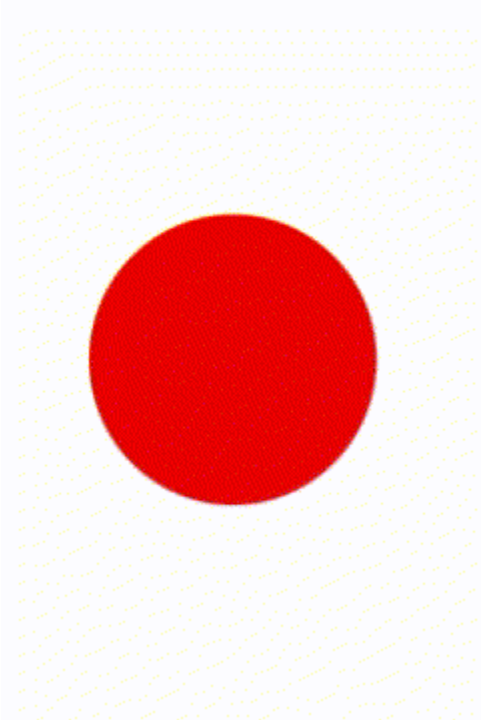
- Synchronized Multimedia Integration Language (SMIL; wie englisch: smile – lächeln)
- SMIL ist ein eigenes Animationsmodell eines SVGs

- 1998 vom W3C entwickelt (SVG erst 2001)
- Für SVG-Animationen, aber auch für Menüs auf DVDs und Youtube-Untertitel
- Wurde eine lange Zeit nur von wenigen Browsern unterstützt
  - Browserunterstützung könnte auch in Zukunft wieder abnehmen

Quelle: https://www.mediaevent.de/tutorial/svg-animation-css-javascript.html, https://wiki.selfhtml.org/wiki/SMIL

Beispiel SMIL – animate-Element

```
<svg id="icon-ellipse" width="200" height="200" viewBox="0 0 148 148">
    <ellipse cx="75" cy="75" rx="40" ry="40" fill="red">
        <animate attributeName="ry" to="70"
                 dur="3s" repeatCount="99"></animate>
    </ellipse>
</svg>
```

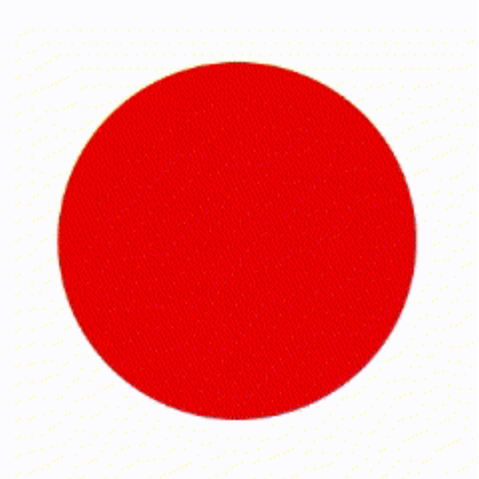


Quelle: https://wiki.selfhtml.org/wiki/SMIL

Beispiel SMIL – Verknüpfung der id

```
<svg width="200" height="200">
    <circle id="icon-circle" cx="100" cy="100" r="100" fill="red">
    </circle>

    <animate href="#icon-circle" attributeName="fill"
             values="red; darkorange; purple; black"
             calcMode="discrete" dur="4s" repeatCount="99">
    </animate>
</svg>
```



Vor- und Nachteile von SMIL-Animationen

Vorteile:

- Kleine Dateigrößen
- Skalierbar und responsiv
- Bewegung entlang eines Pfades
- Eventsteuerung mit hover oder click

Nachteile:

- Komplexere Animationen werden hiermit schwierig
- Geringe Browserunterstützung
- Der Einsatz wird nicht empfohlen

Quelle: https://www.h2d2.de/de/blog/2021/animationstechniken-im-vergleich/

Möglichkeiten der Animation mit CSS

CSS Transitions

CSS Keyframes

CSS Transitions

- CSS „transition“
- Ein- und Ausblenden und Bewegen von HTML-Elementen ohne Javascript
- Über 50 Eigenschaften können angepasst werden

- Transitions nur bei absoluten und relativen Werten
  → z.B. von height: 0px zu height: 100px, von 0% zu 90%,
  aber nicht von height: 0px zu height: auto

Quelle: https://www.mediaevent.de/css/transition.html

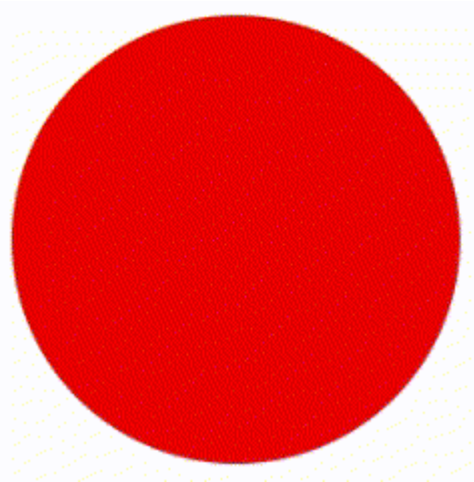
Beispiel CSS Transitions

```svg
<svg width="200" height="200">
    <circle id="icon-circle" cx="100" cy="100" r="100" fill="red"></circle>
</svg>
```

```css
#icon-circle {
    fill: red;
    transition-property: fill;
    transition-duration: 1s;
}

#icon-circle:hover {
    fill: orange;
}
```

```css
#icon-circle {
    fill: red;
    transition: fill 1s;
}
```



CSS Transitions-Eigenschaften

transition	Für die Einstellung aller nachfolgenden Übergangsseigenschaften
transition-property	Die Eigenschaft, die während des Übergangs geändert wird.
transition-duration	Dauer des Übergangs (in Sekunden)
transition-timing-function	Animationskurve für das Bewegungsmuster (ease, ease-in, ease-in-out, linear, cubic-bezier(n,n,n,n), step-start, step-end, steps(int,start|end))
transition-delay	Verzögerung (in Sekunden), bevor der Übergang startet

Quelle: https://www.mediaevent.de/css/transition.html

Technology
Arts Sciences
TH Köln

Aufgabe 1 – CSS Transitions

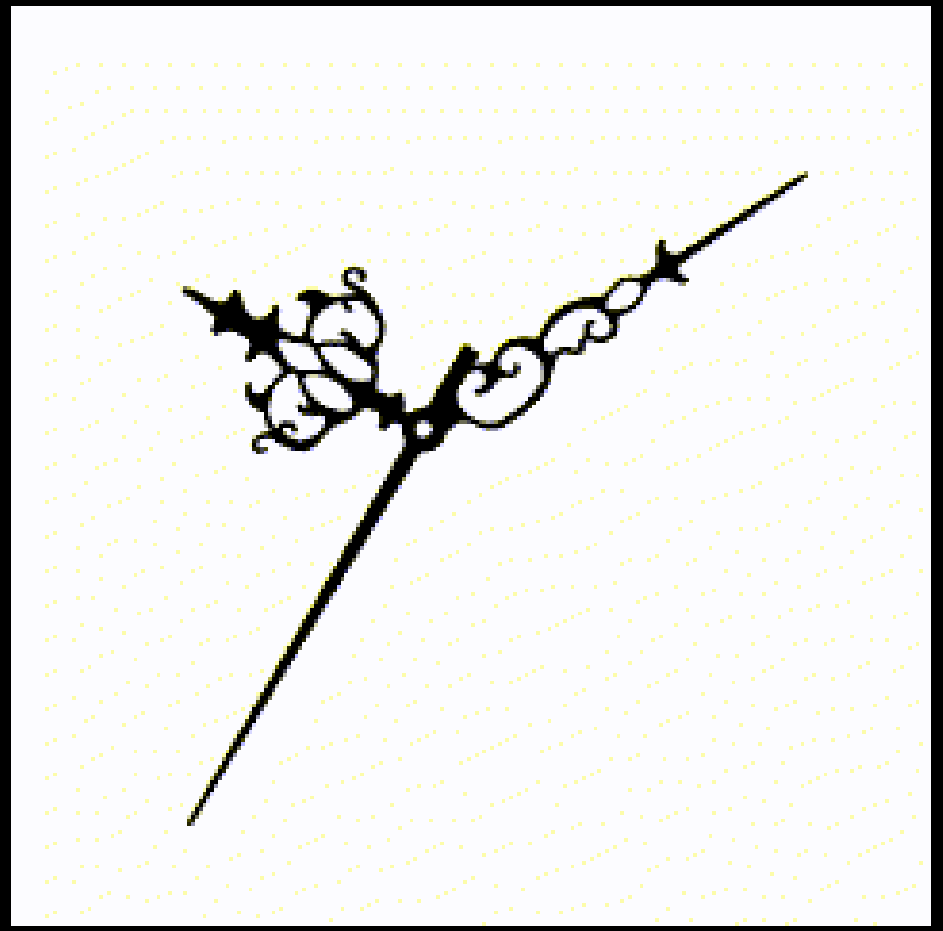
10 Minuten

Aufgabe 1 – CSS Transitions

- Erstellung einer CSS Transition anhand des Videos

Kriterien:

- Vorher sind alle Zeiger schwarz
- Bei hover auf einen Zeiger soll dieser die Farbe ändern



Lösung zu Aufgabe 1 – CSS Transitions

```css
#icon-transition #stundenzeiger {
    transition: fill 1s;
}

#icon-transition #stundenzeiger:hover {
    fill: red;
}

#icon-transition #minutenzeiger {
    transition: fill 1s;
}

#icon-transition #minutenzeiger:hover {
    fill: orange;
}
```

```css
#icon-transition #sekundenzeiger {
    transition: fill 1s;
}

#icon-transition #sekundenzeiger:hover {
    fill: purple;
}
```

Technology
Arts Sciences
TH Köln

Pause – 10 Minuten

CSS Keyframes

- CSS Eigenschaft „animation“
- Ein- und Ausblenden und Bewegen von HTML-Elementen ohne JavaScript

- @keyframes Liste
- Den zu animierenden CSS Eigenschaften werden unterschiedliche Werte zu bestimmten Zeitpunkten zugeordnet
- Zwischen diesen festgelegten Zeitpunkten wird der Verlauf automatisch berechnet

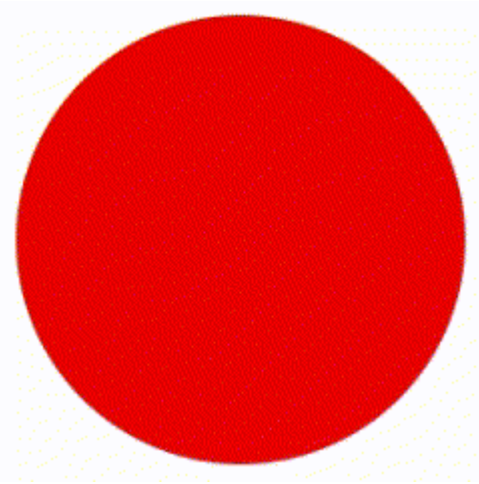
Quelle: https://wiki.selfhtml.org/wiki/SVG/Tutorials/Einstieg/SVG_mit_CSS_animieren

Beispiel CSS Keyframes

```svg
<svg width="200" height="200">
    <circle id="icon-circle-color" cx="100" cy="100" r="100" fill="red"></circle>
</svg>
```

```css
#icon-circle-color {
    animation-name:             color-change;
    animation-duration:         4s;
    animation-direction:        alternate;
    animation-timing-function:  ease-in-out;
    animation-iteration-count:  infinite;
}

@keyframes color-change {
    from {
        fill: red;
    }
    to {
        fill: darkorange;
    }
}
```



Beispiel CSS Keyframes

```css
#icon-circle-color {
    animation-name:             color-change;
    animation-duration:         4s;
    animation-direction:        alternate;
    animation-timing-function:  ease-in-out;
    animation-iteration-count:  infinite;
}
```

```css
#icon-circle-color {
    animation: color-change 4s alternate ease-in-out infinite;
}
```

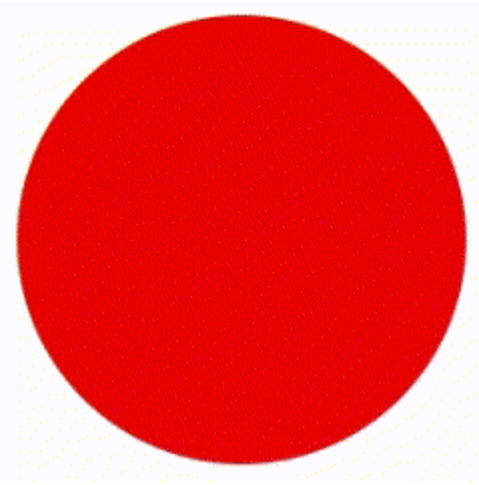
CSS Animations-Eigenschaften

animation	Für die Einstellung aller nachfolgenden Animationseigenschaften
animation-name	Name der Animation
animation-delay	Verzögerung (in Sekunden), bevor die Animation startet
animation-direction	Richtung der Animation (forwards, backwards, alternate)
animation-duration	Dauer der Animation (in Sekunden)
animation-fill-mode	Zustand der Animation vor und/oder nach dem Start (soll es wieder an der Stelle enden, wo es begonnen hat, oder am Endpunkt stehen bleiben?)
animation-iteration-count	Anzahl der Wiederholungen der Animation
animation-play-state	gibt an, ob die Animation gerade abgespielt wird oder pausiert (bspw. für hover-states)
animation-timing-function	Animationskurve für das Bewegungsmuster (ease, ease-in, ease-in-out, linear, cubic-bezier(n,n,n,n), step-start, step-end, steps(int,start|end))

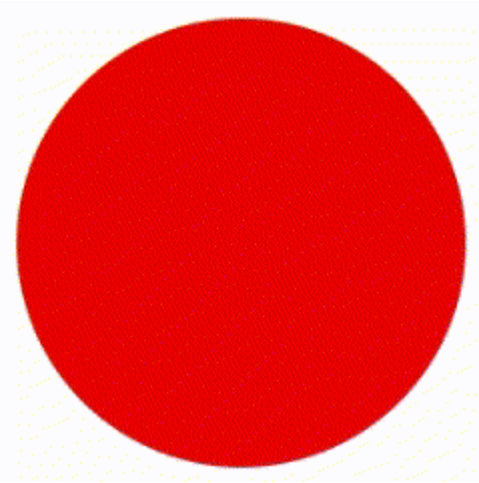
Quelle: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animations/Using_CSS_animations

Beispiel CSS Keyframes

```
@keyframes color-change {
    from {
        fill: red;
    }
    to {
        fill: darkorange;
    }
}
```



```
@keyframes color-change2 {
    0% {
        fill: red;
    }
    50% {
        fill: darkorange;
    }
    100% {
        fill: purple;
    }
}
```



Beispiel CSS Keyframes

```css
#icon-ball-rolling {
    animation: roll 4s normal ease-in-out infinite;
}

@keyframes roll {
    0% {
        transform: translate(0px, 15px) rotate(0) ;

    }
    100% {
        transform: translate(300px, 15px) rotate(360deg) ;
    }
}
```

Beispiel CSS Keyframes - Problemstellung

```css
#icon-ball-rolling {
    animation: roll2 4s normal ease-in-out infinite;
}

@keyframes roll2 {
    0% {
        transform: rotate(0) translate(0px, 15px);

    }
    100% {
        transform: rotate(360deg) translate(300px, 15px);
    }
}
```

Achtung! Richtige Reihenfolge beachten!
→ Konflikt bei der Matrizenmultiplikation

CSS Keyframes – Matrizen Exkurs

```
transform="matrix(a,b,c,d,e,f)"
```

$$\begin{pmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{pmatrix}$$

```
translate(tx,ty)                matrix(1,0,0,1,tx,ty)
rotate(α)                       matrix(cos(α),sin(α),-sin(α),cos(α),0,0)

Mgesamt = M2 * M1
transform="matrix(a2 b2 c2 d2 e2 f2) matrix(a1 b1 c1 d1 e1 f1)"
```

- Die Matrizen sind „falsch herum“ aufgeschrieben.
- In einer Matrizenmultiplikation entspricht die Matrix, die am weitesten rechts steht, der zuerst angewendeten Transformation
- Verkettete Transform-Funktionen in einer transform-Eigenschaft werden von rechts nach links ausgeführt

Für weitere Infos:
https://wiki.selfhtml.org/wiki/CSS/Funktionen/matrix()#matrix.28.29_und_matrix3D.28.29

Vor- und Nachteile von CSS Animationen

Vorteile:

- Kein JavaScript notwendig
  - → Leichtere Lernkurve
  - → Schneller und ruckelfreier, da die Methode nicht so rechenintensiv ist
- Unterstützung der meisten Browser
- Weniger Code in CSS-Stylesheets (im Vergleich zu JavaScript-Dateien)
  - → Schnellere Entwicklungszeit
- Können mehrfach aufgerufen werden (im Gegensatz zu SMIL-Animationen) – nur bei CSS Keyframes

Nachteile:

- Komplexere Animationen sind teilweise schwer umzusetzen (besonders bei CSS Transitions)
- Animation startet nach Laden der Seite (bzw. bei Transition nur bei hover, active etc.)
  - → hier wäre wieder JavaScript notwendig
- nur zwei statt beliebig vielen steps - nur bei CSS Transitions

Quelle: https://www.wpfox.de/coding/animationen-im-web-css-vs-javascript/

Technology
Arts Sciences
TH Köln

Aufgabe 2 – CSS Keyframes

15 Minuten

Aufgabe 2 – CSS Keyframes

- Schreibe Code in CSS, damit sich der Sekundenzeiger dreht

Tipp:
Gib dem SVG eine eigene ID, damit es nicht mit dem SVG aus der vorherigen Aufgabe in Konflikt geraten kann

Lösung zu Aufgabe 2 – CSS Keyframes

```css
#icon-keyframes #sekundenzeiger {
    animation: rotate 10s normal linear infinite;
}

@keyframes rotate {
    0% {
        transform: rotate(0) ;

    }
    100% {
        transform: rotate(360deg) ;
    }
}
```

JavaScript Animationen

- Web Animation API: element.animate ()
- Ähnlicher Aufbau wie CSS Keyframes

animate(keyframes, options)

- zwei Argumente: ein Array von Keyframes und ein Objekt mit den Optionen für die Steuerung des Ablaufs

Kleine Änderungen im Vergleich zu CSS Keyframes

- Zeitangaben in Millisekunden (statt Sekunden)
- Eigenschaften werden in CamelCase geschrieben (statt mit Bindestrichen)
- Easing Voreinstellung ist linear (statt ease-in)

Quelle: https://www.mediaevent.de/javascript/animation-api.html

Beispiel JavaScript Animation

```
function roll() {

    const item = document.getElementById("icon-ball-js");

    const keyframes = [
        { offset: 0, transform: "translate(0px, 15px) rotate(0)"},
        { offset: 0.25, transform: "translate(100px, -30px) rotate(90deg)"},
        { offset: 0.5, transform: "translate(200px, 15px) rotate(180deg)"},
        { offset: 0.75, transform: "translate(300px, -30px) rotate(270deg)"},
        { offset: 1, transform: "translate(400px, 15px) rotate(360deg)"},
    ];

    item.animate ( keyframes, {
        duration: 8000,
        delay: 0,
        fill: "forwards",
        easing: "ease-out",
        iterations: Infinity
    });
}
```

Bessere Steuerung von JavaScript Animationen

```
document.addEventListener("DOMContentLoaded", () => {
    roll();
});
```

```
document.addEventListener("DOMContentLoaded", () => {

    document.getElementById("icon-ball-js").addEventListener("click", () => {
        roll();
    })

});
```

Berechnungen bei JavaScript Animationen

```
const item = document.getElementById("icon-ball-js");

const keyframes = [
    { offset: 0, transform: "translate(0px, 15px) rotate(0)"},
    { offset: 1, transform: "translate(400px, 15px) rotate(360deg)"},
];
```

```
const item = document.getElementById("icon-ball-js");

const width = window.innerWidth - item.getBoundingClientRect().width;
const umdrehungen = 2

const keyframes = [
    { offset: 0, transform: "translate(0px, 15px) rotate(0)"},
    { offset: 0, transform: "translate("+ width + "px, 15px) rotate("+ (umdrehungen * 360) +"deg"},
];
```

Vor- und Nachteile von JavaScript Animationen

Vorteile:

- Auch komplexe Animationen können umgesetzt werden
- Es können Berechnungen durchgeführt werden
- Start der Animation kann durch ein Event eingestellt werden (bspw. bei Klick)

Nachteile:

- Höhere Programmierkenntnisse sind erforderlich
- Größere Dateien → längere Ladezeiten

Quelle: https://www.mediaevent.de/javascript/animation-api.html

Technology
Arts Sciences
TH Köln

Aufgabe 3 – JavaScript Animationen

20 Minuten

Aufgabe 3 – JavaScript Animationen

- Ändere den Code aus Aufgabe 2 für JavaScript um
- Der Sekundenzeiger soll sich drehen

Tipp:
Gib dem SVG eine eigene ID

Bonusaufgaben:
1. Ändere den Code, sodass er nicht linear sondern in steps läuft
(wie bei einem normalen Sekundenzeiger)

2. Schreibe den Code so um, dass alle Zeiger entsprechend ihrer Zeitangabe richtig weiter laufen
(Sekundenzeiger jede Sekunde, Minutenzeiger jede Minute, …)

Lösung zu Aufgabe 3 – JavaScript Animationen

```
function animate() {

    const svg = document.getElementById("icon-javascript");
    const item = svg.getElementById("sekundenzeiger");

    const keyframes = [
        { offset: 0, transform: "rotate(0)"},
        { offset: 1, transform: "rotate(360deg)"},
    ];

    item.animate ( keyframes, {
        duration: 6000,
        delay: 0,
        fill: "forwards",
        easing: "linear",
        iterations: Infinity
    });
}
```

Animationen mit JSON

- JSON = **J**ava**S**cript **O**bject **N**otation
- Datenformat in einer einfach lesbaren Textform
- für den Datenaustausch zwischen Anwendungen
- Programmiersprachen unabhängig

- In der JSON-Datei befindet sich das SVG und die Animation

- Datei kann bspw. ein Motion Designer in After Effects erstellen
- Ohne Animationskenntnisse: man kann das SVG in verschiedene Tools hochladen und darüber animieren (bspw. lottiefiles.com, svgartista.net)

- Datei wird dem Webprojekt hinzugefügt und mit JavaScript (LottieFiles Library) animiert

Quelle: https://de.wikipedia.org/wiki/JavaScript_Object_Notation , https://www.mediaevent.de/javascript/animation-api.html

Was ist Lottie?

- LottieFiles ist eine Programmbibliothek (Library)
- Entwickelt von Mitarbeitern von Airbnb
- Veröffentlicht seit 2017

- Eine Lottie ist ein vektorbasiertes Grafikformat, das auf SVG und JSON basiert
- Erleichtert das Einfügen von Animationen in Webseiten

- Die erstellte Datei wird auch „LottieFiles“ genannt (Dateiformat JSON)

Quelle: https://de.wikipedia.org/wiki/Lottie_(Grafikformat), https://lottiefiles.com/de/what-is-lottie

Beispiel mit Lottie

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/bodymovin/5.7.11/lottie.min.js"></script>

<div id="animation-container" data-lottie-animation="clock.json"></div>
```

```
function anim() {

    const animationContainer = document.getElementById('animation-container');
    const animationPath = animationContainer.getAttribute('data-lottie-animation');

    const animation = lottie.loadAnimation({
        container: animationContainer,
        renderer: 'svg', // Renderer Auswahl (z. B. 'svg' oder 'canvas')
        loop: true,
        autoplay: true,
        path: animationPath
    });
    return animation;
}

window.onload = anim;
```

Vor- und Nachteile von JSON/Lottie-Animationen

Vorteile:

- Auch komplexe Animationen können umgesetzt werden
- viele zusätzliche Optionen und Methoden, um die Animation zu steuern
- Start der Animation kann durch ein Event eingestellt werden (bspw. bei Klick)
- JSON-Dateien basieren auch auf Vektoren → kleine Dateien

  ! dennoch Verwendung von JavaScript

Nachteile:

- Komplexe Animationen nur mit Profis möglich (Tools haben nur standardisierte Animationen)
- Höhere Programmierkenntnisse sind erforderlich
- (Abhängigkeit von einer Library)

Quelle: https://raiok.com/2023/10/09/lottie-animationen-ein-blick-in-die-zukunft-der-web-animationen/

Heute haben wir folgendes gelernt

- Was SVGs sind
- Welche Methoden es gibt, um SVGs zu animieren
  - SMIL
  - CSS Transitions und Keyframes
  - JavaScript
  - JSON bzw. Lottie-Animationen
- Vor- und Nachteile dieser Methoden

Ausblick

Weitere Tools:

- dotLottie (.Lottie)
- GSAP
- Animate.css
- Skia
- Three.js (3D)
- ….

Technology
Arts Sciences
TH Köln

Vielen Dank für eure Aufmerksamkeit!