Scalable Vector Graphics
SVG

Manfred Kaul

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Verbundprojekt
work&study
Offene Hochschulen Rhein-Saar

Was ist SVG?

umgangssprachlich: *"SVG ist das HTML für Vektor-Graphik"*

Teil des DOM

Vektorgrafik ist skalierbar

Vektorgrafik

Bitmap

Was ist SVG?

"Scalable Vector Graphics (SVG) is an **XML***-based markup language for describing* **two dimensional** *based* **vector graphics**. *SVG is essentially to graphics what HTML is to text.*

SVG is a **text-based open Web standard**. *It is explicitly designed to work with other web standards such as CSS, DOM, and SMIL.*

SVG images and their related behaviors are defined in XML text files which means they can be searched, indexed, scripted and compressed. Additionally this means they can be created and edited with any text editor and with drawing software.

SVG is an open standard developed by the World Wide Web consortium(W3C) since 1999."

Use

The

Platform

Inline SVG wie HTML nutzen

Inline SVG

```
<!doctype html>
<h1>SVG Demo</h1>

<svg viewBox="0 0 300 200">

  <rect width="100%" height="100%" fill="red" />
  <circle cx="150" cy="100" r="80" fill="green" />

  <text x="150" y="125"
      font-size="60" text-anchor="middle"
      fill="white">SVG</text>

</svg>
```

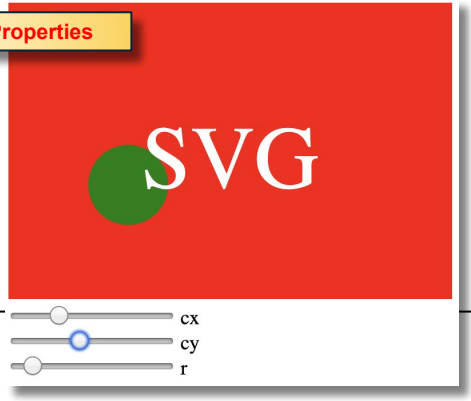
SVG-Element als Web-Komponente

```
<script type="module">
 import { LitElement, svg } from 'https://unpkg.com/lit-element/lit-element.js?module';
 customElements.define('draw-demo', class extends LitElement {
   render() {
     return svg`
       <svg viewBox="0 0 300 200">
         <rect width="100%" height="100%" fill="red" />
         <circle cx="${this.cx}" cy="${this.cy}" r="${this.r}" fill="green" />
         <text x="150" y="125"
             font-size="60" text-anchor="middle"
             fill="white">SVG</text>
       </svg>`;
   }
 static get properties() {
  return {
    cx: {type: Number},
    cy: {type: Number},
    r: {type: Number}
  }
 }
```

Tagged Template Literal

Parametrisierung

Managed Properties




```
 constructor(){
     super();
     this.cx = 150;
     this.cy = 100;
     this.r = 80;
   }
 });

 const demo = document.createElement('draw-demo');
 document.body.appendChild( demo );
 ['cx','cy','r'].forEach( prop => {
   const div = document.createElement('div');
   div.innerHTML = `
      <input type="range" name="${prop}" min="0" max="300">
      <label for="${prop}">${prop}</label>
   `;
   const input = div.querySelector('input');
   input.addEventListener('input', e => {
     demo[prop] = input.value;
   });
   document.body.appendChild( div );
 });
</script>
```

Incremental Re-Rendering triggered by changes of managed properties

Interactive SVG in Web Component

SVG-Koordinatensystem identisch mit dem DOM-Koordinatensystem

```
render() {
 return svg`
   <svg viewBox="0 0 300 200" width="300px" height="200px">
     <rect width="100%" height="100%" fill="red" />
     <circle cx="${this.cx}" cy="${this.cy}" r="80" fill="green" />
     <text x="150" y="125"
         font-size="60" text-anchor="middle"
         fill="white">SVG</text>
   </svg>`;
}
static get properties() {
 return {
   cx: {type: Number},
   cy: {type: Number}
 }
}
```

Bewege den Kreis mit der Maus

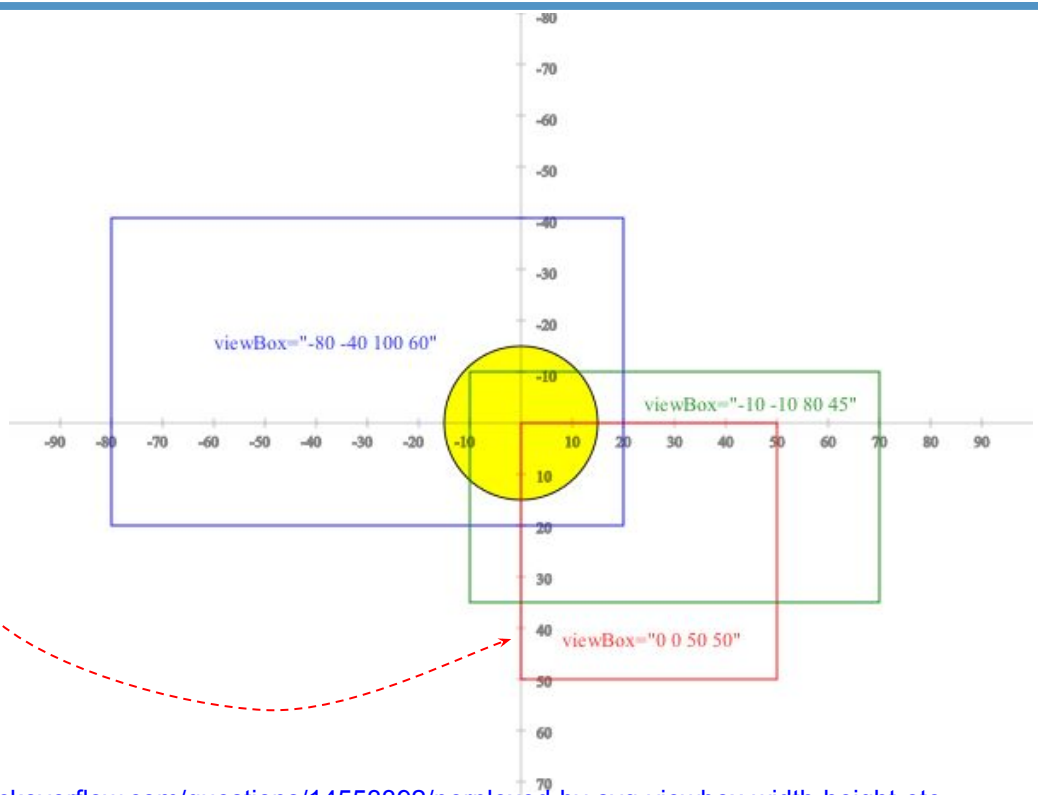
```
constructor(){
 super();
 this.cx = 150;
 this.cy = 100;
 this.moving = false;
 this.addEventListener('mousedown', e => {
   this.moving = true;
 });
 this.addEventListener('mouseup', e => {
   this.moving = false;
 });
 this.addEventListener('mousemove', event => {
   if ( this.moving ){
     this.cx = event.x;
     this.cy = event.y;
   }
 });
}
```

SVG-Koordinaten des Zentrums des Kreises werden mit den DOM-Koordinaten des Events überschrieben

SVG-Koordinatensystem

viewBox= "min-x min-y width height"

- 2D (x,y)
- unendlich in x und y
- viewBox = Ausschnitt
- `<svg viewBox="0 0 50 50">`



JSFiddle

https://stackoverflow.com/questions/14553392/perplexed-by-svg-viewbox-width-height-etc

https://software.intel.com/en-us/xdk/blog/viewbox-a-window-into-the-soul-of-svg

SVG-Element im DOM-Koordinatensystem

- Das SVG-Wurzelelement wird als Rechteck wie jedes HTML-Element auf der Seite platziert
- Default **width="100%"** ⇒ Default ist **Responsivität**
- Gleichschaltung von SVG-Koordinatensystem mit DOM-Koordinatensystem über width & height möglich, verhindert dadurch aber auch Responsivität
  - `<svg viewBox="0 0 50 50" width="50px" height="50px">`

Gleichschaltung

Transformation von DOM-Koordinaten ins SVG-Koordinaten

Cumulative Transformation Matrix (CTM)

$$\begin{bmatrix} x_{\text{viewport}} \\ y_{\text{viewport}} \\ 1 \end{bmatrix} = \text{CTM} \cdot \begin{bmatrix} x_{\text{userspace}} \\ y_{\text{userspace}} \\ 1 \end{bmatrix}$$

$$\text{CTM} = \begin{bmatrix} a_{\text{ctm}} & c_{\text{ctm}} & e_{\text{ctm}} \\ b_{\text{ctm}} & d_{\text{ctm}} & f_{\text{ctm}} \\ 0 & 0 & 1 \end{bmatrix}$$

SVGElement.getScreenCTM()

Returns a cumulative transformation matrix to convert from the user coordinate system of this element to the “screen” or client coordinates used to represent points on the root-level document. This method is useful in event handlers when converting between mouse/pointer coordinates and the coordinate system of your graphics.

```
render() {
 return svg`
   <svg viewBox="0 0 300 200">
     ...
   </svg>`;
}
constructor(){
 super();
 ...
 this.addEventListener('mousemove', e => {
   if ( this.moving ){
     const svg = this.shadowRoot.querySelector('svg');
     const pt = svg.createSVGPoint();
     pt.x = e.clientX;
     pt.y = e.clientY;
     const svgP = pt.matrixTransform(svg.getScreenCTM().inverse());
     this.cx = svgP.x;
     this.cy = svgP.y;
   }
 });
}
```

ohne width gilt width=100%

Transformation via Matrix-Multiplikation

CTM for mapping SVG units to screen coordinates

Matrix-Invertierung

https://www.sitepoint.com/how-to-translate-from-dom-to-svg-coordinates-and-back-again/

Einbettung von SVG in HTML

- **Inline SVG** in **HTML5**
  - Advantage: SVG elements are *first-class* DOM elements

- **SVG Image** `<img src="bild.svg">`

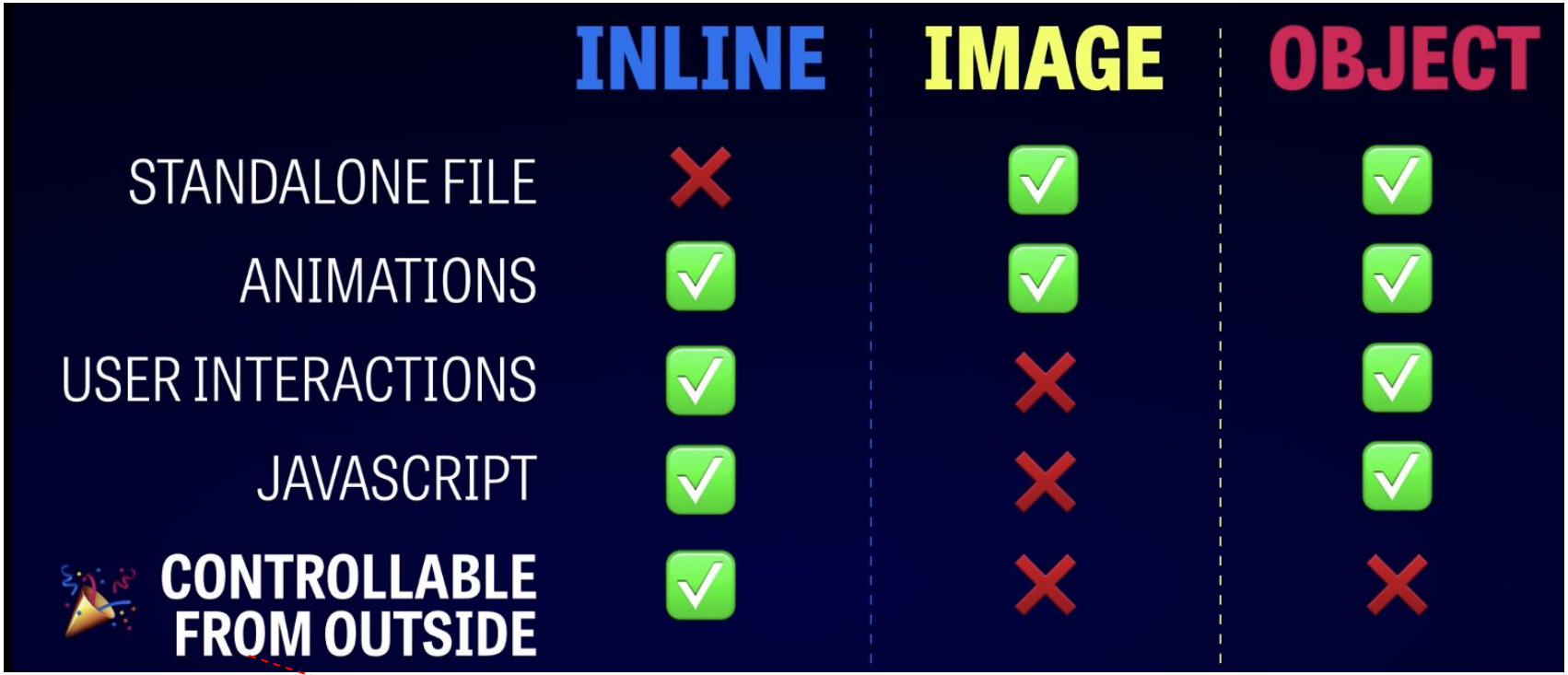
- **SVG as Background** in CSS stylesheet
  - `background: url( bild.svg );`

Anime.js Tutorial - Morphing SVG's in Web Design

- SVG Objekt <embed src=...> <iframe src=...> <object data=...>

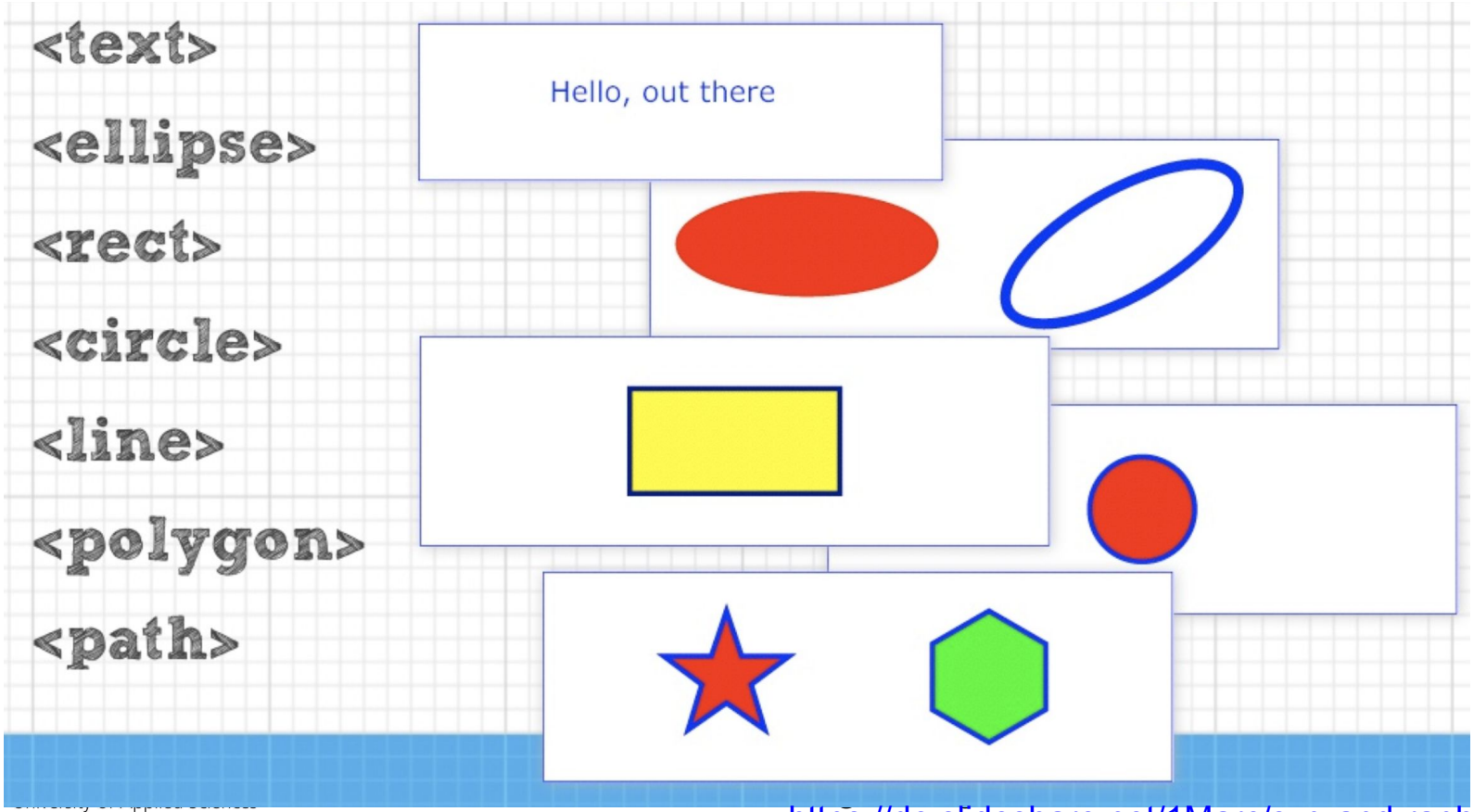
https://www.youtube.com/watch?v=af4ZQJ14yu8

SVG in HTML einbinden



https://www.youtube.com/watch?v=af4ZQJ14yu8

Die wichtigsten 7 SVG-Elemente



https://de.slideshare.net/1Marc/svg-and-raphael-js

Alle 85 SVG-Elemente

https://developer.mozilla.org/en-US/docs/Web/SVG/Element

SVG elements A to Z

A
<a>
<altGlyph>
<altGlyphDef>
<altGlyphItem>
<animate>
<animateColor>
<animateMotion>
<animateTransform>
<audio>

B C
<canvas>
<circle>
<clipPath>
<color-profile>
<cursor>

D
<defs>
<desc>
<discard>

E
<ellipse>

F
<feBlend>
<feColorMatrix>
<feComponentTransfer>
<feComposite>
<feConvolveMatrix>
<feDiffuseLighting>
<feDisplacementMap>
<feDistantLight>
<feDropShadow>
<feFlood>
<feFuncA>
<feFuncB>
<feFuncG>
<feFuncR>
<feGaussianBlur>
<feImage>
<feMerge>
<feMergeNode>
<feMorphology>
<feOffset>
<fePointLight>
<feSpecularLighting>
<feSpotLight>
<feTile>
<feTurbulence>
<filter>
<font>
<font-face>
<font-face-format>
<font-face-name>
<font-face-src>
<font-face-uri>
<foreignObject>

G
<g>
<glyph>
<glyphRef>

H
<hatch>
<hatchpath>
<hkern>

I
<iframe>
<image>

J K L
<line>
<linearGradient>

M
<marker>
<mask>
<mesh>
<meshgradient>
<meshpatch>
<meshrow>
<metadata>
<missing-glyph>
<mpath>

N O P
<path>
<pattern>
<polygon>
<polyline>

Q R
<radialGradient>
<rect>

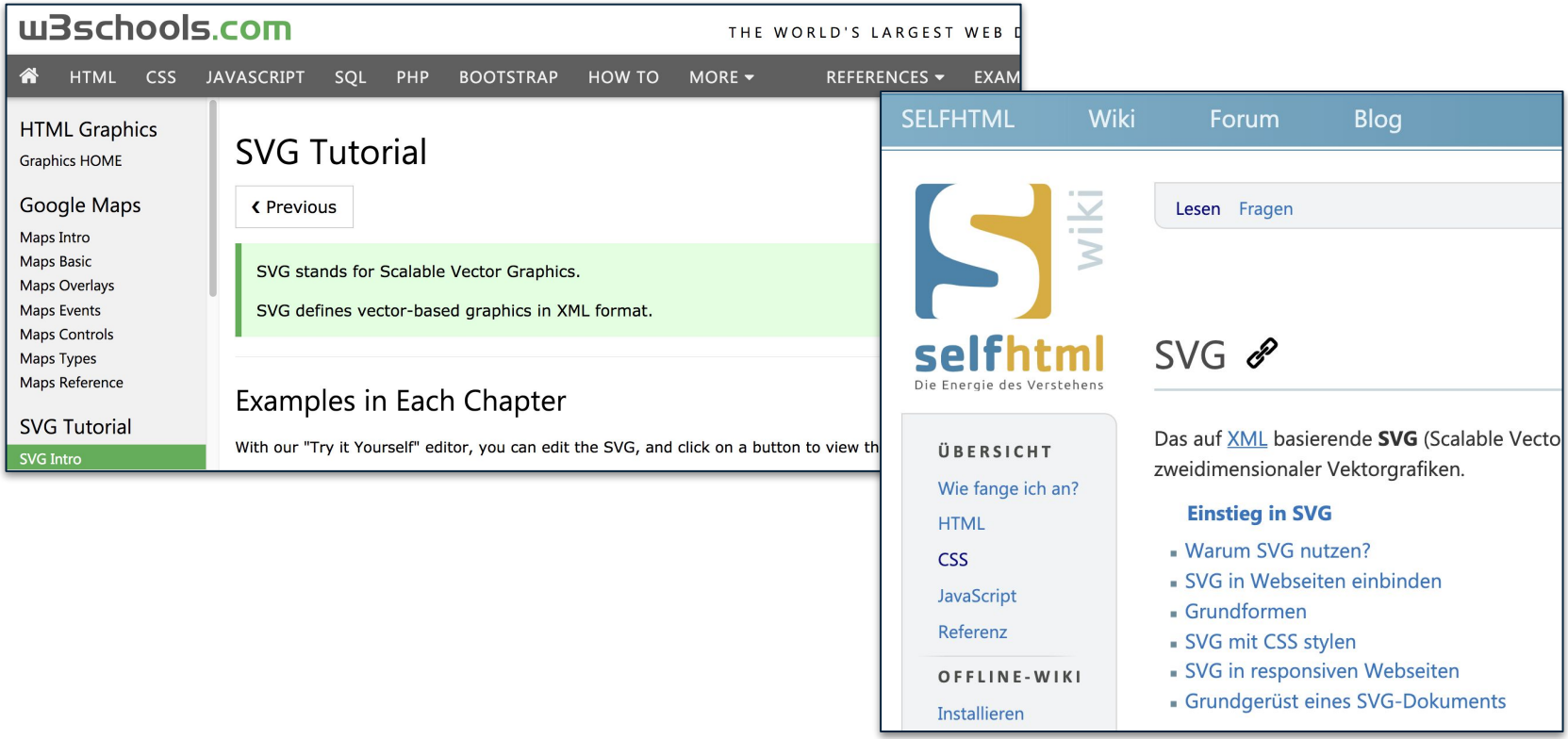
S
<script>
<set>
<solidcolor>
<stop>
<style>
<svg>
<switch>
<symbol>

T
<text>
<textPath>
<title>
<tref>
<tspan>

U
<unknown>
<use>

V — Z
<video>
<view>
<vkern>

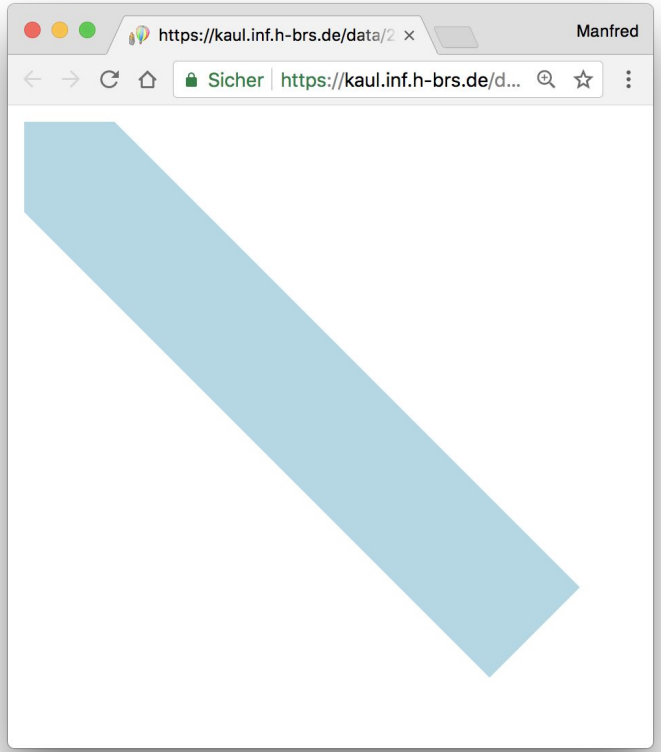
SVG Tutorials



SVG Element <line>

```
<svg viewBox="0 0 120 120">
  <line
        x1="0"
        y1="0"
        x2="100"
        y2="100"
        stroke="lightblue"
        stroke-width="25">
  </line>
</svg>
```

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/line



SVG Element <text>

```svg
<svg xmlns="http://www.w3.org/2000/svg"
     width="500" height="40" viewBox="0 0 500 40">

  <text x="0" y="35" font-family="Verdana" font-size="35">
    Hello, out there
  </text>
</svg>
```

Result

No word wrap

Hello, out there

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/text

SVG Element <foreignObject>

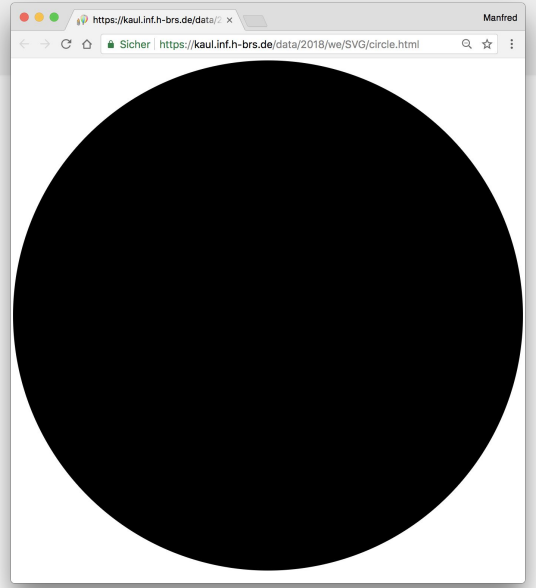
```
<svg viewBox="0 0 240 100" xmlns="http://www.w3.org/2000/svg">
  <style>
    .very_small { font: italic 6px sans-serif; }
        ...
  </style>

  ...
  <text x="120" y="55" class="Rrrrr">Grumpy!</text>
  <foreignObject x="20" y="70" width="200" height="160" class="very_small">
    <div xmlns="http://www.w3.org/1999/xhtml">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Sed mollis mollis mi ut ultricies. Nullam magna ipsum,
      porta vel dui convallis, rutrum imperdiet eros. Aliquam
      erat volutpat.
    </div>
  </foreignObject>
</svg>
```

Das **<foreignObject>**-Element ermöglicht die Einbindung eines fremden XML-Namespace, dessen graphischer Inhalt von einem anderen User-Agent dargestellt wird. Der eingebundene fremde graphische Inhalt ist Subjekt der SVG-Transformationen und des Compositing.

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/foreignObject

SVG Element <circle>

```
1 <svg viewBox="0 0 200 200" xmlns="http://www.w3.org/2000/svg">
2   <circle cx="100" cy="100" r="100"/>
3 </svg>
```

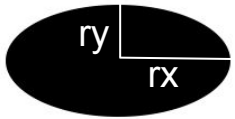


https://developer.mozilla.org/en-US/docs/Web/SVG/Element/circle

SVG Element <ellipse>

```
1  <svg width="120" height="120" viewBox="0 0 120 120"
2      xmlns="http://www.w3.org/2000/svg">
3
4    <ellipse cx="60" cy="60" rx="50" ry="25"/>
5  </svg>
```

Result



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/ellipse

https://de.wikipedia.org/wiki/Ellipse

SVG Element <rect>

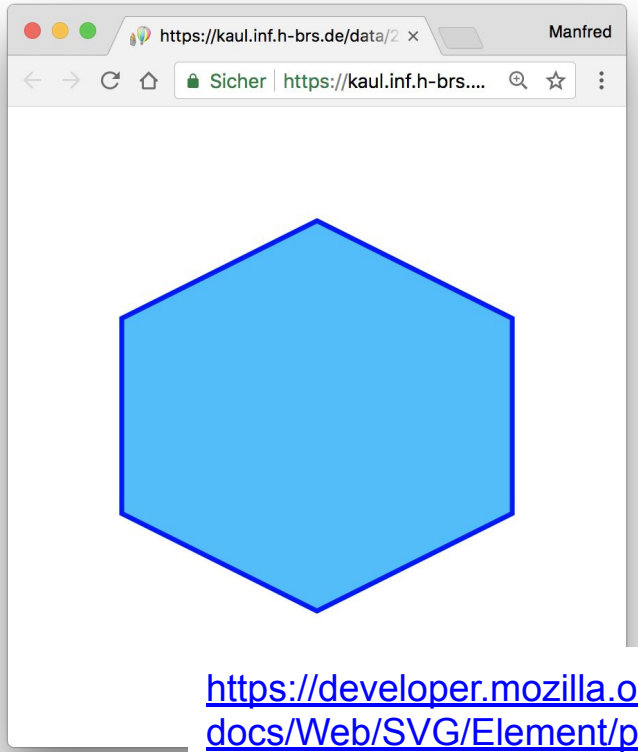
```svg
<svg width="120" height="120" viewBox="0 0 120 120"
     xmlns="http://www.w3.org/2000/svg">

  <rect x="10" y="10" width="100" height="100"/>
</svg>
```

Result

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/rect

SVG Element <polygon>

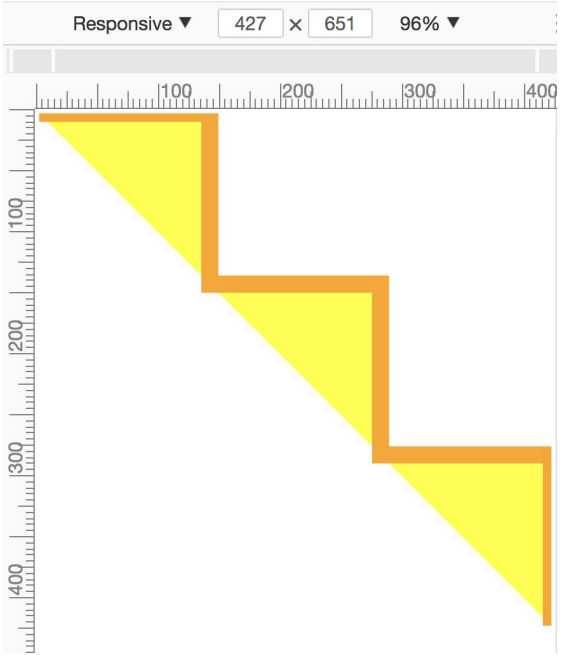
```
<svg viewBox="0 0 120 120">
  <polygon fill="DeepSkyBlue"
           points="60,20 100,40
                   100,80 60,100
                   20,80 20,40"
           stroke="Blue"
           stroke-width="1">
  </polygon>
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/polygon

SVG Element <polyline>

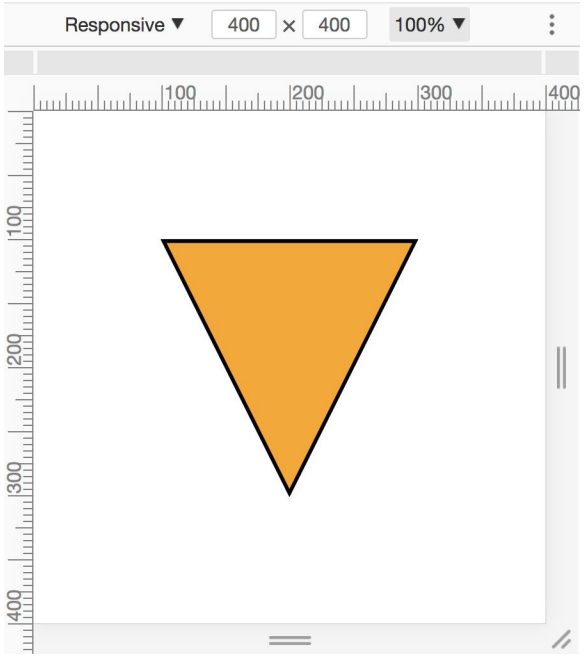
```
<svg viewBox="0 40 120 120">
  <polyline
        points="0,40 40,40
            40,80 80,80
            80,120 120,120
            120,160"
        fill="yellow"
        stroke="orange"
        stroke-width="4" />
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/polyline

SVG Element <path>

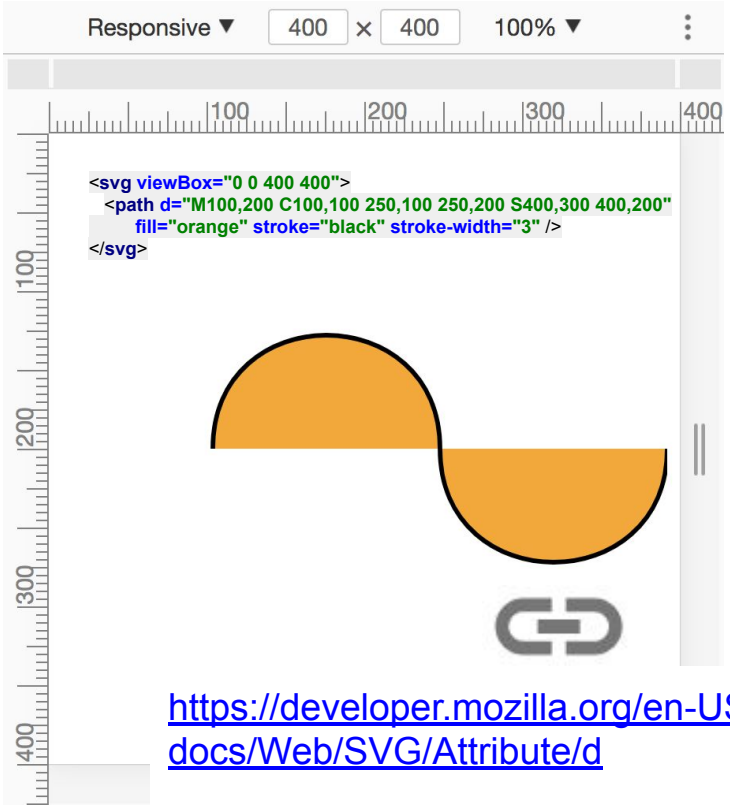
```
<svg viewBox="0 0 400 400">
  <path d="  M 100 100
             L 300 100
             L 200 300
             z"
      fill="orange"
      stroke="black"
      stroke-width="3" />
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/path

SVG Element <path d="M L Z">

Absolute	Relative	Command / Direction
M	m	moveTo M x,y
L	l	lineTo L x,y
C	c	curveTo (Cubic Bezier) C c1x,c1y c2x,c2y x,y
Q	q	curveTo (Quadratic Bezier) Q cx,cy x,y
S	s	curveTo (Smooth Cubic Bezier) S cx,cy x,y (für weichen Übergang)
T	t	curveTo (Smooth Quadratic Bezier) T cx,cy x,y
A	a	arcTo A rx,ry xAxisRotate LargeArcFlag,SweepFlag x,y
Z = z		closePath

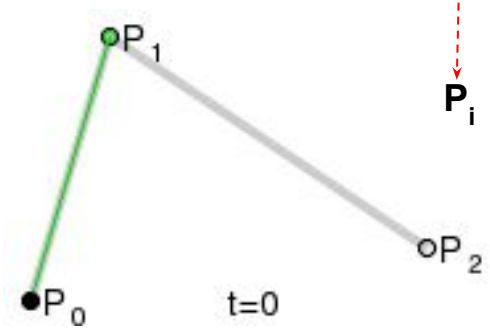


https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/d

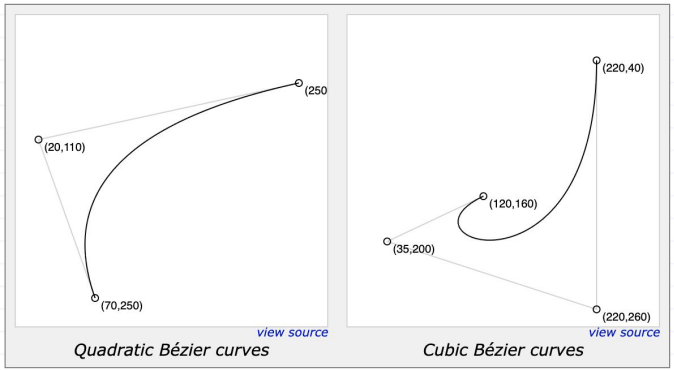
Bezierkurven

Eine quadratische Bézierkurve ist der Pfad, der durch die Funktion **b**(*t*) wie folgt beschrieben wird:

$$\mathbf{b}(t) = \sum_{i=0}^{2} \binom{2}{i} t^i (1-t)^{2-i} \mathbf{b}_i$$



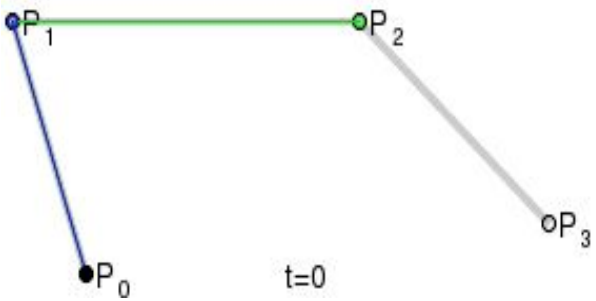
https://de.wikipedia.org/wiki/B%C3%A9zierkurve



https://pomax.github.io/bezierinfo/

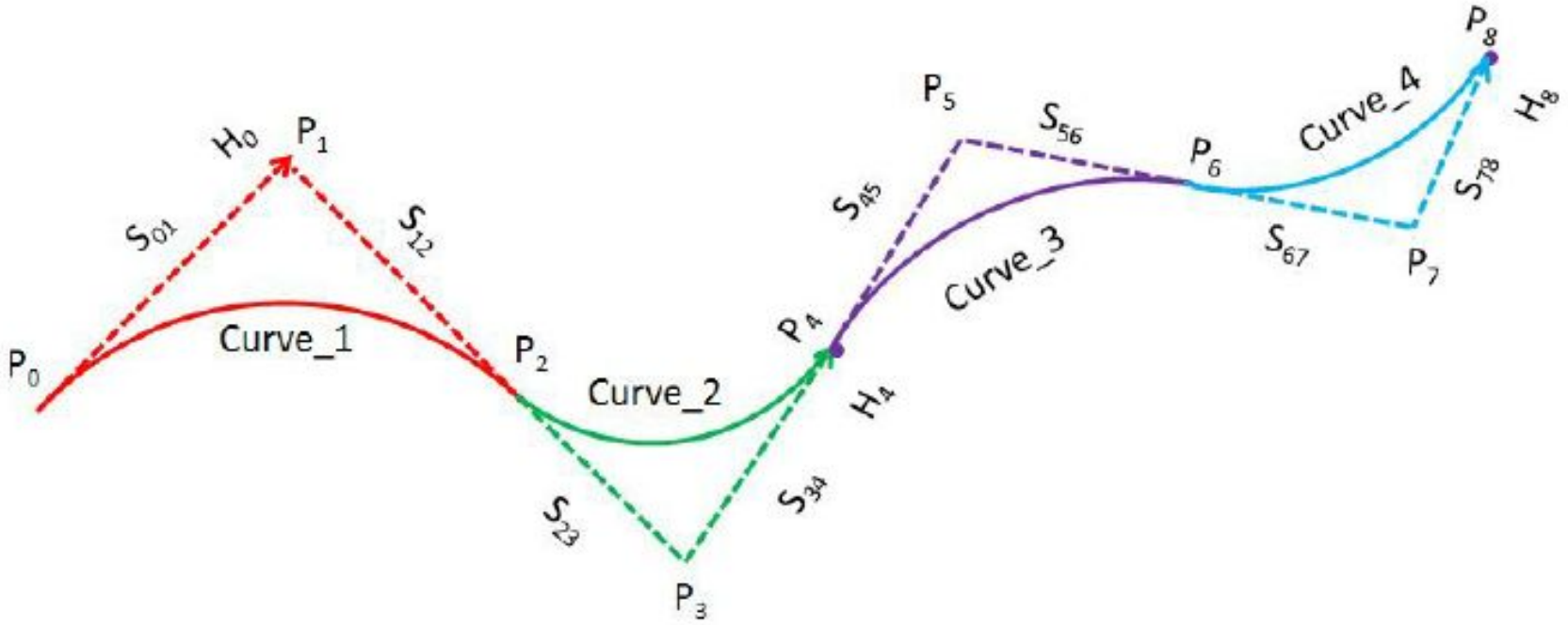
Eine kubische Bézierkurve ist der Pfad, der durch die Funktion **b**(*t*) wie folgt beschrieben wird:

$$\mathbf{b}(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} \mathbf{b}_i$$



https://de.wikipedia.org/wiki/B%C3%A9zierkurve

Weicher Übergang zwischen Bezierkurven



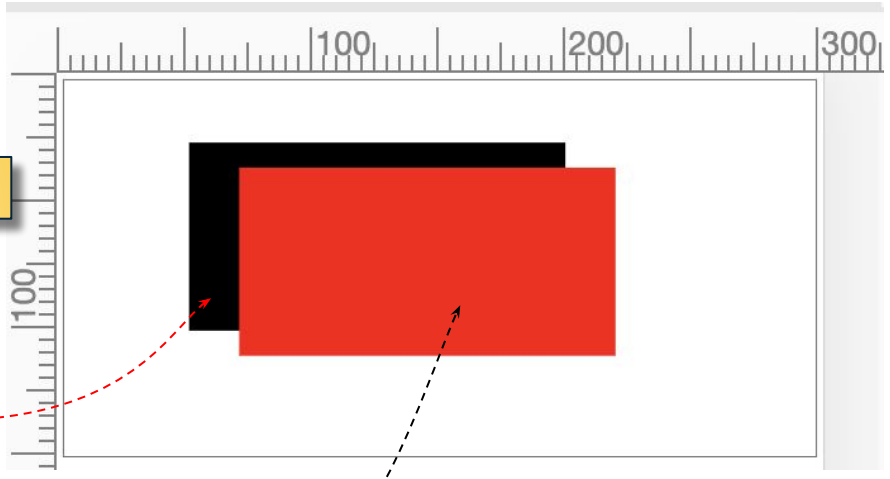
SVG Definitionen: <defs> und <use>

```
<style>
  svg {
    border: thin solid;
  }
</style>
<svg viewBox="0 0 300 150">

  <defs>
    <rect id="box" x="50" y="25" width="150" height="75"></rect>
  </defs>

  <use href="#box" />
  <use href="#box" x="20" y="10" fill="red"/>
</svg>
```

Was innerhalb von <defs> steht, wird nicht gezeichnet

<use href="#box"/> setzt nur voraus, dass es ein Element mit id="box" gibt. Dieses muss nicht innerhalb von <defs> stehen.

Translation um (20,10)

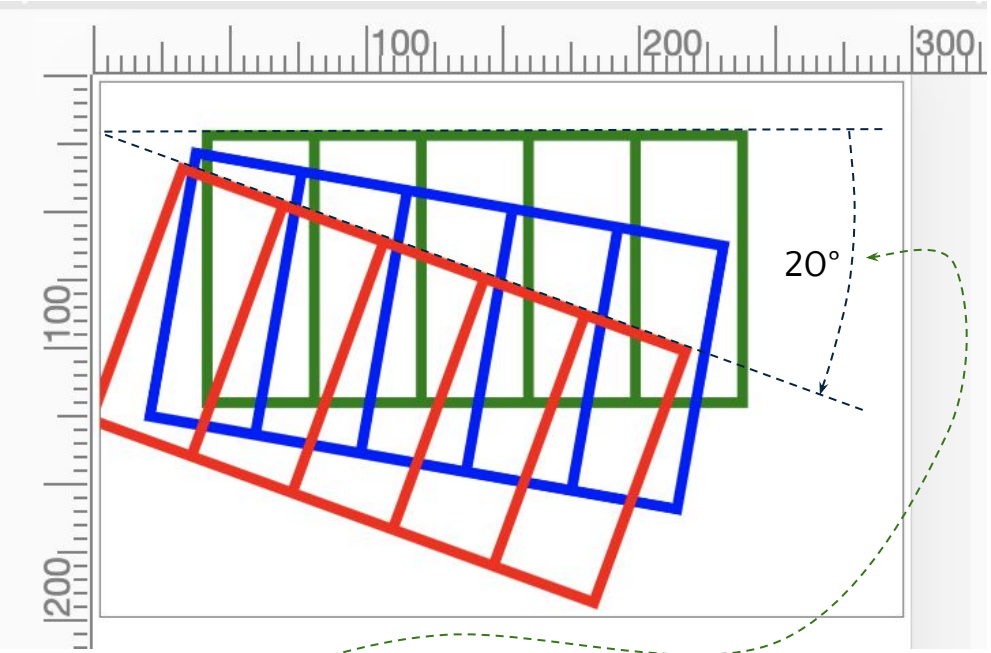


https://developer.mozilla.org/en-US/docs/Web/SVG/Element/defs

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/use

SVG Element Grouping <g>

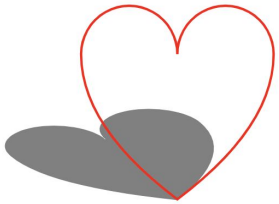
```
<style>
  svg {
    border: thin solid;
  }
</style>
<svg viewBox="0 0 150 100">
  <defs>
    <g id="ladder" stroke-width="2" fill-opacity="0.0">
      <rect x="20" y="10" width="100" height="50"/>
      <line x1="40" y1="10" x2="40" y2="60"/>
      <line x1="60" y1="10" x2="60" y2="60"/>
      <line x1="80" y1="10" x2="80" y2="60"/>
      <line x1="100" y1="10" x2="100" y2="60"/>
    </g>
  </defs>
  <use href="#ladder" stroke="green"/>
  <use href="#ladder" stroke="blue" transform="rotate(10)"/>
  <use href="#ladder" stroke="red" transform="rotate(20)"/>
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/g

SVG transform

```
<svg viewBox="-40 0 150 100">
  <g fill="grey"
    transform="rotate(-10 50 100)
          translate(-36 45.5)
          skewX(40)
          scale(1 0.5)">
     <path id="heart" d="M 10,30 A 20,20 0,0,1 50,30
                A 20,20 0,0,1 90,30 Q 90,60 50,90 Q 10,60 10,30 z" />
  </g>
  <use href="#heart" fill="none" stroke="red"/>
</svg>
```



$$\mathrm{CTM} = \begin{bmatrix} a_{\mathrm{ctm}} & c_{\mathrm{ctm}} & e_{\mathrm{ctm}} \\ b_{\mathrm{ctm}} & d_{\mathrm{ctm}} & f_{\mathrm{ctm}} \\ 0 & 0 & 1 \end{bmatrix}$$

- **translate (x (y))**
  Verschieben entlang der x/y-Achse. x und y werden entweder durch Leerzeichen oder durch Komma getrennt.
- **rotate (a (x (y)))**
  Rotieren um einen Winkel a. Die optionalen Parameter x und y versetzen den Punkt, um den das Element rotiert wird.
- **scale (x (y))**
  Verkleinern (mit Werte < 0) bzw. Vergrößern (mit Werten > 0) auf der xy-Achse. Werte (-1 1) bzw.
  (1 -1) spiegeln das Element an der x- bzw. y-Achse. Zwei Werte werden entweder durch Leerzeichen oder ein Komma getrennt.
- **skewX(x) skewY(y)**
  Verzerren oder Neigen entlang der xy-Achse

https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/transform

Alle Transformationen in CTM berücksichtigt

Fortunately, the `.getScreenCTM()` can be used on any SVG element and **the resulting matrix** considers **all transformations**. We can therefore create a generic `svgPoint` translation function:

```
// translate DOM to SVG coordinates
function svgTransform( domX, domY, svgRoot, svgElement ) {
 if ( ! svgElement ) svgElement = svgRoot;
 const svgPoint = svgRoot.createSVGPoint();
 svgPoint.x = domX;
 svgPoint.y = domY;
 return svgPoint.matrixTransform( svgElement.getScreenCTM().inverse() );
}
```

Transformation via Matrix-Multiplikation

CTM for mapping SVG units to screen coordinates including all transformations of svgElement

Matrix-Invertierung

https://www.sitepoint.com/how-to-translate-from-dom-to-svg-coordinates-and-back-again/

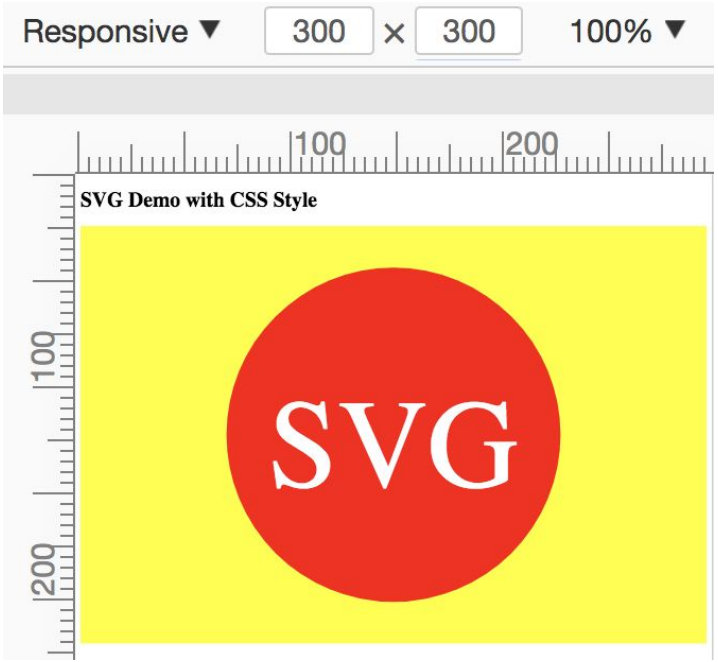
CSS styling via SVG Element <style>

```
<!doctype html>
<h1>SVG Demo with CSS Style</h1>
<svg viewBox="0 0 300 200">
  <style>
    rect {
      width: 100%;
      height: 100%;
      fill: yellow
    }
    circle {
      fill: red;
    }
    text {
      font-size: 60px;
      text-anchor: middle;
      fill: white;
    }
  </style>
```

SVG

```
  <rect />
  <circle cx="150" cy="100" r="80" />
  <text x="150" y="125">SVG</text>
</svg>
```

SVG



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/style

CSS styling via HTML Element <style>

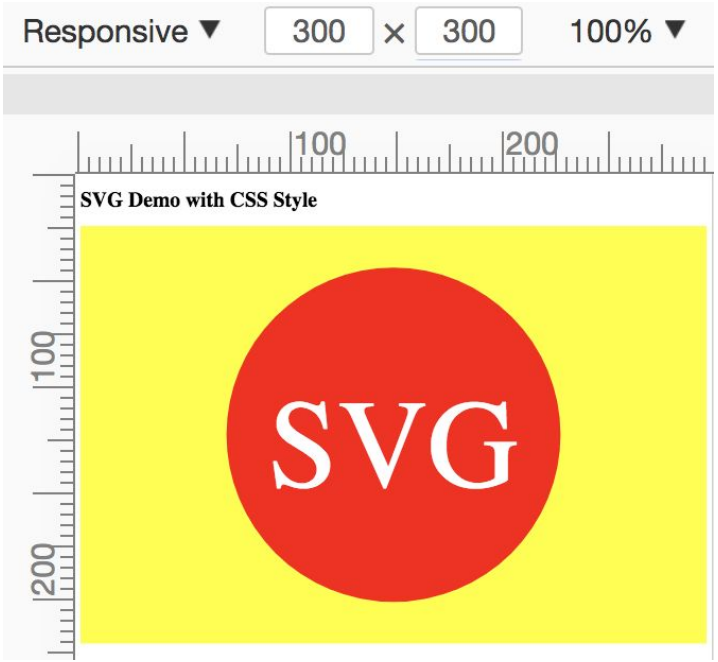
HTML

```
<!doctype html>
<style>
  rect {
    width: 100%;
    height: 100%;
    fill: yellow
  }
  circle {
    fill: red;
  }
  text {
    font-size: 60px;
    text-anchor: middle;
    fill: white;
  }
</style>
<h1>SVG Demo with CSS Style</h1>
```

SVG

```
<svg viewBox="0 0 300 200">
  <rect />
  <circle cx="150" cy="100" r="80" />
  <text x="150" y="125"> SVG </text>
</svg>
```

Trennung von Layout und Inhalt

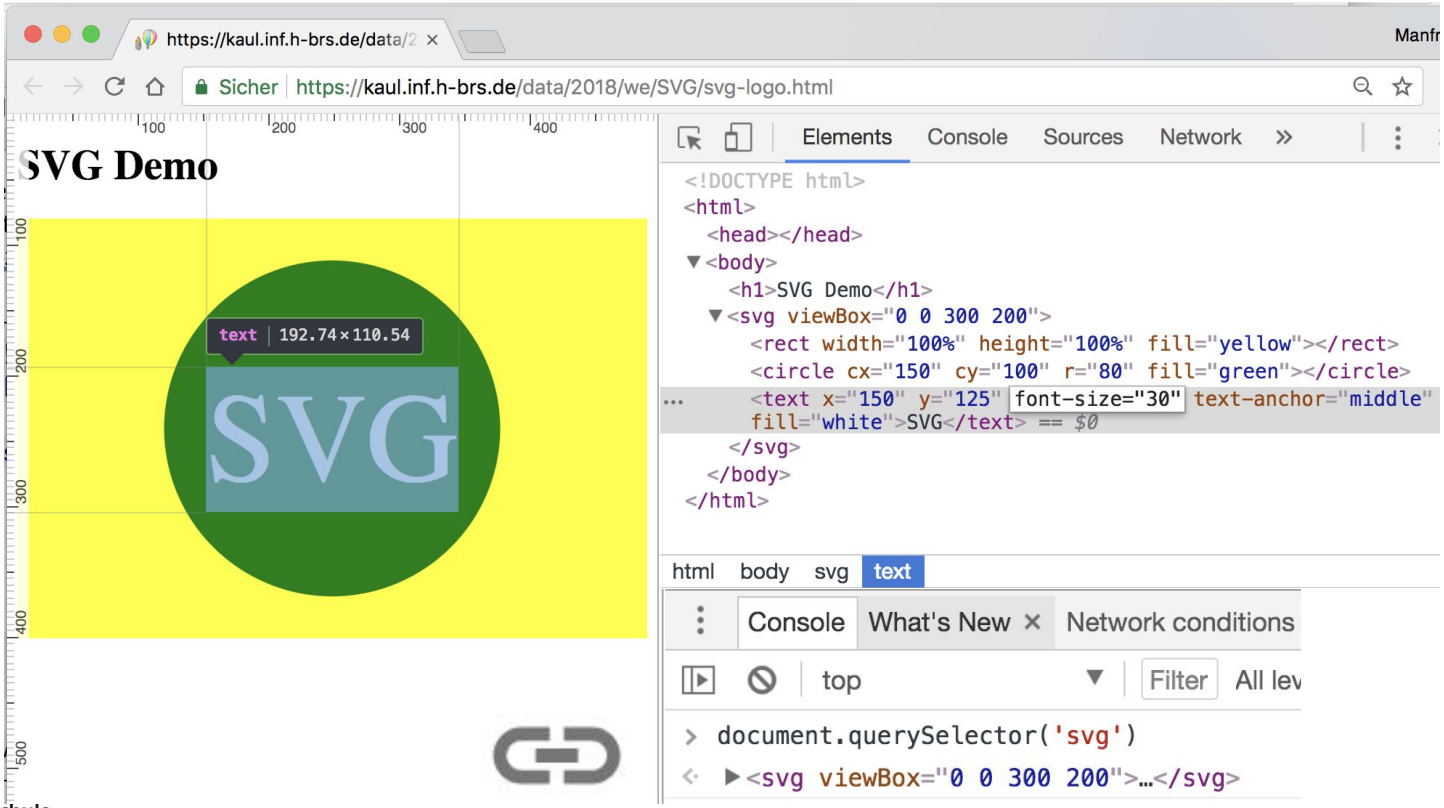


https://developer.mozilla.org/en-US/docs/Web/SVG/Element/style

CSS Hover-Effekte in SVG

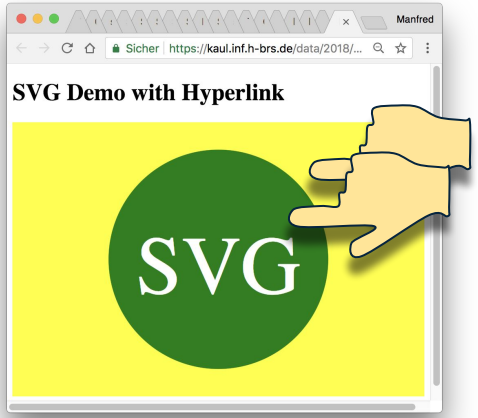
```
<!doctype html>
<style>
  circle:hover {
    fill: red;
  }
</style>
<h1>SVG Demo with CSS Hover</h1>
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle cx="150" cy="100" r="80" />
  <text x="150" y="125" font-size="60"
        text-anchor="middle" fill="white">SVG</text>
</svg>
```

SVG in Chrome Devtools



Hyperlinks in SVG mit <a href>

- exakte "Hotareas" in Grafiken
- keine Rechtecke wie in HTML
  - der **Kreis** ist die SVG-Hotarea




```
<!doctype html>
<h1>SVG Demo with Hyperlink</h1>
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <a href="https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/href" target="_blank">
     <circle cx="150" cy="100" r="80" fill="green" />
  </a>
  <text x="150" y="125" font-size="60" text-anchor="middle" fill="white">SVG</text>
</svg>
```

DOM Events in SVG

```
<!doctype html>
<h1>SVG Demo with Events</h1>
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle cx="150" cy="100" r="80"
          onmouseover="this.setAttribute('r',100)"
          onmouseout="this.setAttribute('r', 60)">
  </circle>
  <text x="150" y="125" font-size="60" text-anchor="middle"
fill="white">SVG</text>
</svg>
```

SVG Demo with Events

JavaScript Event Handler on SVG Elements

```
<!doctype html>
<h1>SVG Demo with JS Event Handler</h1>
<script>
  function increase(elem){
    elem.setAttribute('r',100);
  }
  function decrease(elem){
    elem.setAttribute('r',60);
  }
</script>
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle cx="150" cy="100" r="80"
          onmouseover="increase(this)"
          onmouseout="decrease(this)">
  </circle>
  <text x="150" y="125" font-size="60" text-anchor="middle"
fill="white">SVG</text>
</svg>
```

Unobstrusive JavaScript via <script>

Trennung von Programmierung und Inhalt

```
<!doctype html>
<h1>SVG Demo with JS Event Handler</h1>

<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle id="svg-circle" cx="150" cy="100" r="80" />
  <text x="150" y="125" font-size="60" text-anchor="middle"
fill="white">SVG</text>
</svg>
```

SVG

<script>-Tag in SVG **und** HTML erlaubt.

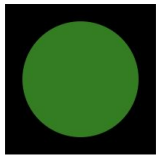
```
<script>
  const svg_circle = document.getElementById("svg-circle");
  svg_circle.addEventListener("mouseover", event => {
    event.target.setAttribute('r',100);
  });
  svg_circle.addEventListener("mouseout", event => {
    event.target.setAttribute('r',60);
  });
</script>
```

JavaScript

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/script

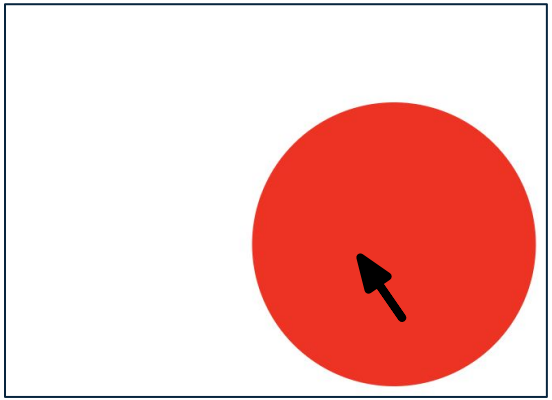
SVG-DOM-Manipulation via JS

```
<!doctype html>
<svg id="svg"></svg>
<script>
  const svg = document.getElementById("svg");
  svg.innerHTML = `
     <rect x="10" y="10" width="100" height="100" />
     <circle id="circle" cx="60" cy="60" r="40"
          stroke="black" stroke-width="3" fill="red" />
  `;
  let color = 'red';
  const circle = document.getElementById("circle");
  setInterval(()=>{
    color = color === 'red'? 'green' : 'red';
    circle.setAttribute( 'fill', color );
  }, 500);
</script>
```



Interaktion mit SVG-Elementen

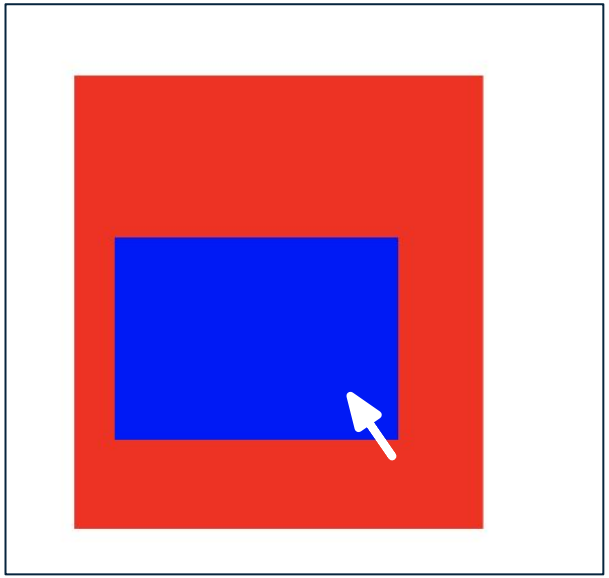
```
<!doctype html>
<svg>
  <circle id="circle" cx="100" cy="100" r="50" fill="red"></circle>
</svg>
<script>
  const circle = document.getElementById('circle');
  circle.addEventListener("mousemove", event => {
    circle.setAttribute('cx', event.x );
    circle.setAttribute('cy', event.y );
  });
</script>
```



Kreis folgt der Maus in Echtzeit.

Interaktivität in SVG

```
<!doctype html>
<svg>
  <rect class="resizable" x="10" y="10" width="100" height="50"
fill="red"></rect>
  <rect class="resizable" x="20" y="50" width="30" height="100"
fill="blue"></rect>
</svg>
<script>
  const resizables = [...document.getElementsByClassName('resizable')];
  resizables.forEach(rect => {
    rect.addEventListener("mousemove", event => {
      rect.setAttribute('width', event.x - 10);
      rect.setAttribute('height', event.y - 10);
    })
  })
</script>
```

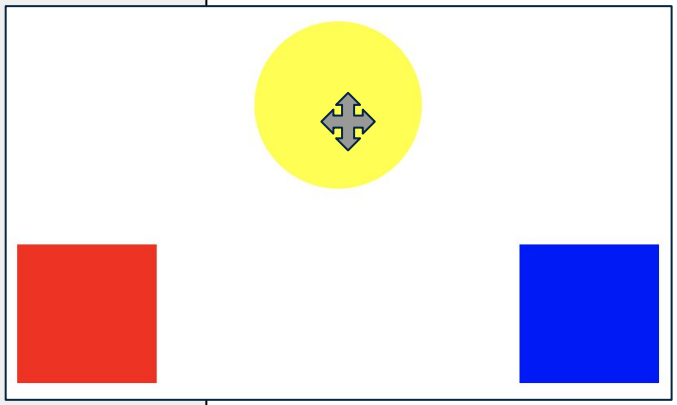


Größe der Rechtecke verändern

Drag & Drop in SVG -1-

```
<!doctype html>
<style>
  .static {
     cursor: not-allowed;
  }
  .draggable {
     cursor: move;
  }
</style>
<svg>
  <circle cx="135" cy="50" r="30" fill="yellow" class="draggable"></circle>
  <rect x="20" y="100" width="50" height="50" fill="red" class="static"></rect>
  <rect x="200" y="100" width="50" height="50" fill="blue" class="static"></rect>
</svg>
<script>
  const draggables = [...document.querySelectorAll('.draggable')];
  draggables.forEach( d => makeDraggable( d ) );

  const rects = [...document.querySelectorAll('rect')];
  rects.forEach(rect => rect.addEventListener('mouseover', e => rect.setAttribute('fill', 'yellow') ));
```

Alle verschiebbaren Elemente mit der Klasse "**draggable**" markieren



Drag & Drop in SVG -2-

```
function makeDraggable( elem ) {
 elem.addEventListener('mousedown', startDrag);
 elem.addEventListener('mousemove', drag);
 elem.addEventListener('mouseup', endDrag);
 elem.addEventListener('mouseleave', endDrag);
 let selectedElement, offset;

 function startDrag(evt) {
   if (evt.target.classList.contains('draggable')) {
     selectedElement = evt.target;
     offset = getMousePosition(evt);
     offset.x -= parseFloat(selectedElement.getAttribute("cx"));
     offset.y -= parseFloat(selectedElement.getAttribute("cy"));
   }
 }
```

```
 function drag(evt) {
   if (selectedElement) {
     evt.preventDefault();
     var coord = getMousePosition(evt);
     selectedElement.setAttribute("cx", coord.x - offset.x);
     selectedElement.setAttribute("cy", coord.y - offset.y);
   }
 }
 function endDrag(evt) {
   selectedElement = null;
 }
 function getMousePosition(evt) {
   var CTM = document.querySelector('svg').getScreenCTM();
   return {
     x: (evt.clientX - CTM.e) / CTM.a,
     y: (evt.clientY - CTM.f) / CTM.d
   };
 }
}
```

http://www.petercollingridge.co.uk/tutorials/svg/interactive/dragging/

Animation in SVG

3 Arten:

1. CSS **<style>**

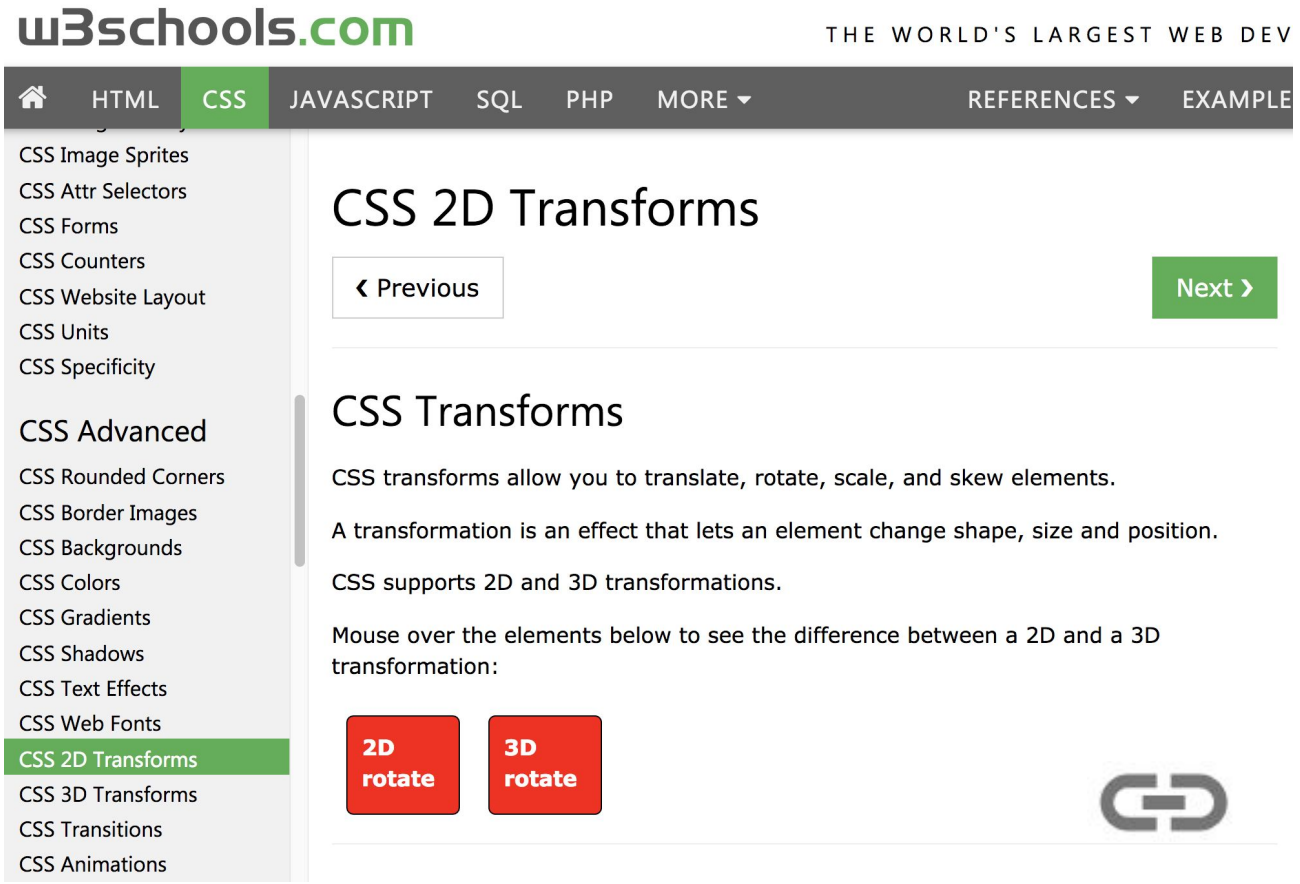
SMIL

2. SVG **<animate>**, **<animateMotion>**, **<animateTransform>**, **<set>**

3. JavaScript **<script>**

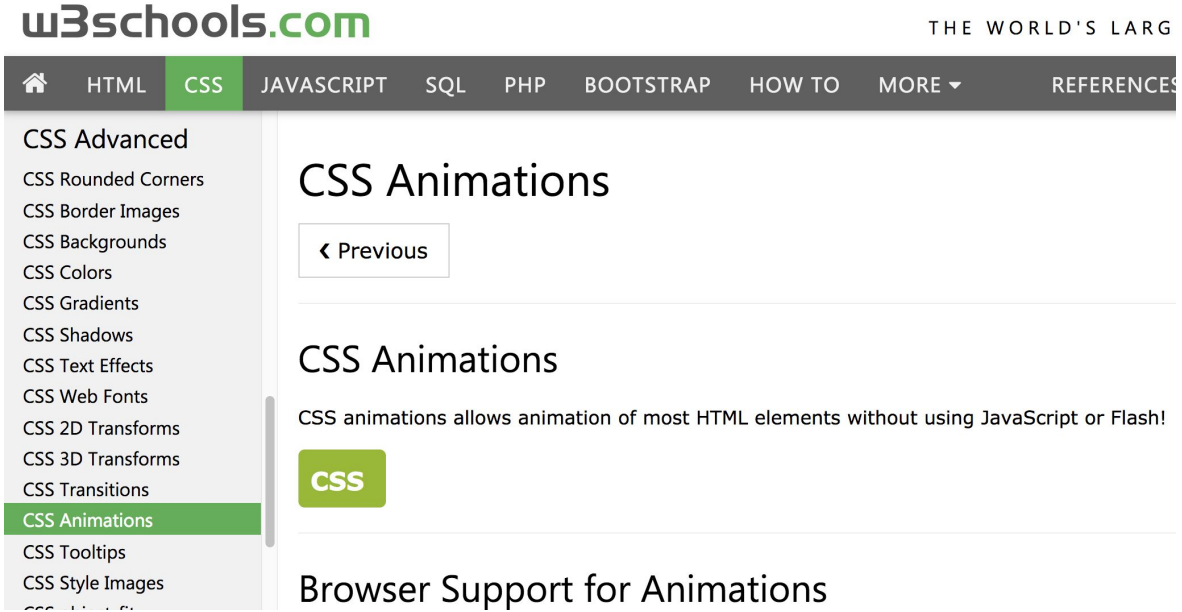
1. requestAnimationFrame
2. setInterval
3. setTimeout
4. Element.animate()

CSS 2D-Transformationen



CSS-Transitionen

CSS-Animationen



Test Yourself with Exercises!

Exercise 1 » Exercise 2 » Exercise 3 » Exercise 4 » Exercise 5 »

Exercise 6 »

Animation in SVG via CSS

```
<style>
  .starStyle {
     animation-name: starAnim;
     animation-duration: 2s;
     animation-iteration-count: 4;
     animation-direction: alternate;
     animation-timing-function: ease;
     animation-play-state: running;
  }
  @keyframes starAnim {
     0% {
        fill-opacity: 1.0;
        stroke-width: 0;
     }

     100% {
        fill-opacity: 0;
        stroke-width: 6;
     }
  }
</style>
```

```
<svg viewBox="0 0 200 200">
  <defs>
     <g id="starDef">
        <path d="M 38.042 -12.361 9.405 -12.944 -0.000 -40.000
     -9.405 -12.944 -38.042 -12.361 -15.217 4.944
     -23.511 32.361 0.000 16.000 23.511 32.361 15.217 4.944 Z"/>
     </g>
  </defs>

  <use id="star" class="starStyle" xlink:href="#starDef"
       transform="translate(100, 100)"
       style="fill: #008000; stroke: #008000"/>
</svg>
```

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

SVG mit SMIL-Animation

```
<rect x="10" y="10" width="200" height="20"
                    stroke="black" fill="none">
  <animate id="animation"
        attributeName="width"
        attributeType="XML"
        from="200"
        to="20"
        begin="0s"
        dur="5s"
        fill="freeze">
  </animate>
</rect>
```

- Was soll animiert werden? ⇒ width
  - XML-Attribut oder CSS oder auto (default)?
- Anfangswert beim Start der Animation
- Endwert beim Ende der Animation
- Startzeit
- Dauer
- "how to **fill** up the remaining time":
  - **freeze** = einfrieren beim Endwert

https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animate

Synchronized Multimedia Integration Language – SMIL


```
<rect x="10" y="10" width="200" height="20"
                    stroke="black" fill="none">
  <animate id="animation"
        attributeName="width"
        attributeType="XML"
        from="200" to="20"
        begin="0s" dur="5s"
        fill="freeze">
  </animate>
</rect>
```

SVG's animation elements were developed in collaboration with the W3C Synchronized Multimedia (SYMM) Working Group, developers of the Synchronized Multimedia Integration Language (SMIL) 3.0 Specification [SMIL].

The SYMM Working Group, in collaboration with the SVG Working Group, has authored the SMIL Animation specification [SMILANIM], which represents a general-purpose XML animation feature set. SVG incorporates the animation features defined in the SMIL Animation specification and provides some SVG-specific extensions.

For an introduction to the approach and features available in any language that supports SMIL Animation, see SMIL Animation overview and SMIL Animation animation model ([SMILANIM], sections 2 and 3). For the list of animation features which go beyond SMIL Animation, see SVG extensions to SMIL Animation.

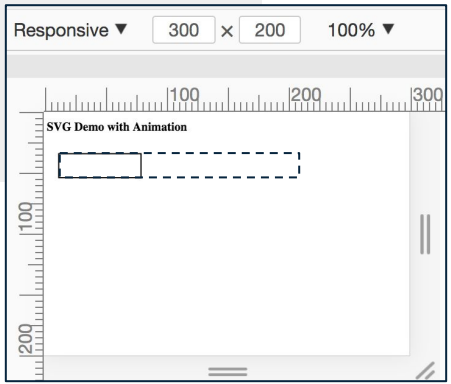
https://svgwg.org/specs/animations/#AnimationElementsIntro

SVG Animation mit SMIL

- **<animate>** – change values of numeric properties over time
- **<set>** – assign values to non numeric properties
- animateColor – modify color values over time (*deprecated*)
- **<animateTransform>** – modify transform values over time
- **<animateMotion>** – move an element along a path

Animation in SVG mit <animate>

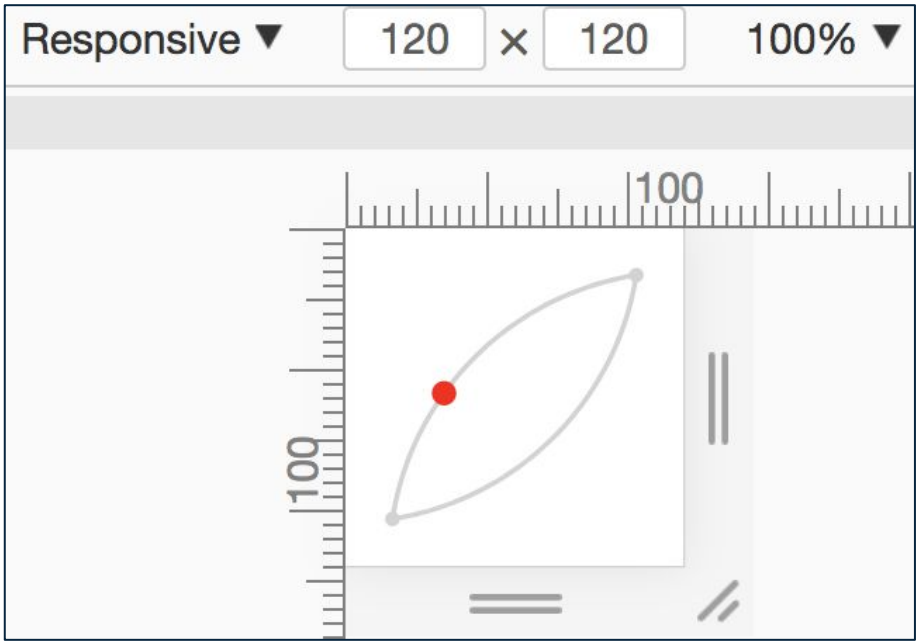
```
<!doctype html>
<h1>SVG Demo with Animation</h1>
<svg viewBox="0 0 300 200">
<rect x="10" y="10"
      width="200" height="20"
      stroke="black" fill="none">
  <animate
      attributeName="width"
      from="200" to="20"
      begin="0s" dur="5s"
      fill="freeze" />
</rect>
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animate

Animation in SVG mit <animateMotion>

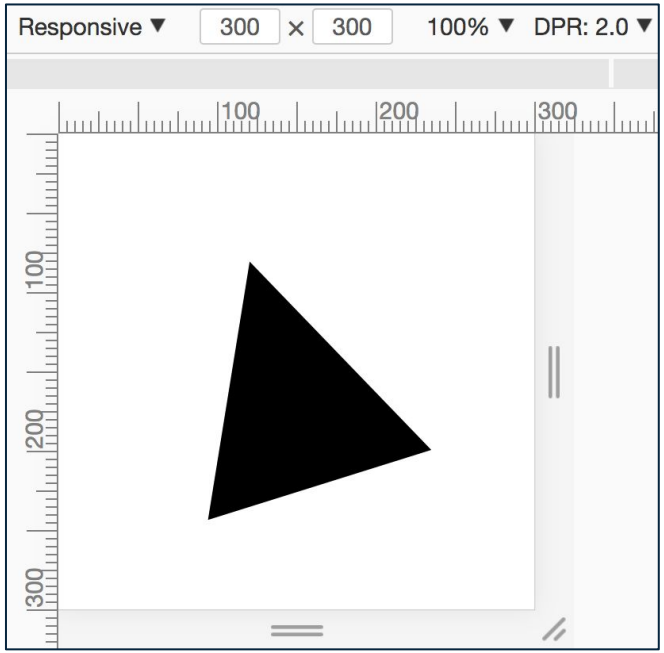
```
<svg viewBox="0 0 120 120">
  <!-- Draw the outline of the motion path in grey, along
       with 2 small circles at key points -->
  <path d="M10,110 A120,120 -45 0,1 110 10
           A120,120 -45 0,1 10,110"
        stroke="lightgrey" stroke-width="2"
        fill="none" id="theMotionPath"/>
  <circle cx="10" cy="110" r="3" fill="lightgrey" />
  <circle cx="110" cy="10" r="3" fill="lightgrey" />

  <!-- Red circle which will be moved along the motion path. -->
  <circle cx="" cy="" r="5" fill="red">

    <!-- Define the motion path animation -->
    <animateMotion dur="6s" repeatCount="indefinite">
      <mpath href="#theMotionPath"/>
    </animateMotion>
  </circle>
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateMotion

Animation in SVG mit <animateTransform>

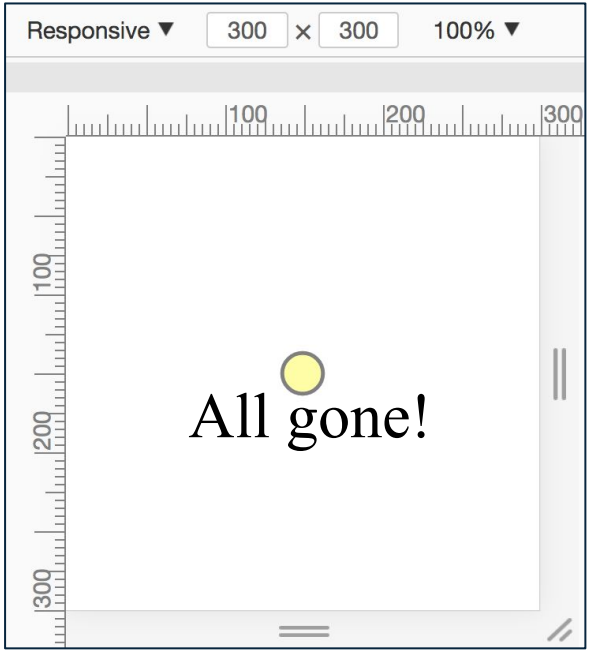
```
<svg viewBox="0 0 120 120">
  <polygon points="60,30 90,90 30,90">
    <animateTransform attributeName="transform"
        attributeType="XML"
        type="rotate"
        from="0 60 70"
        to="360 60 70"
        dur="10s"
        repeatCount="indefinite"/>
  </polygon>
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateTransform

Animation in SVG mit <set>

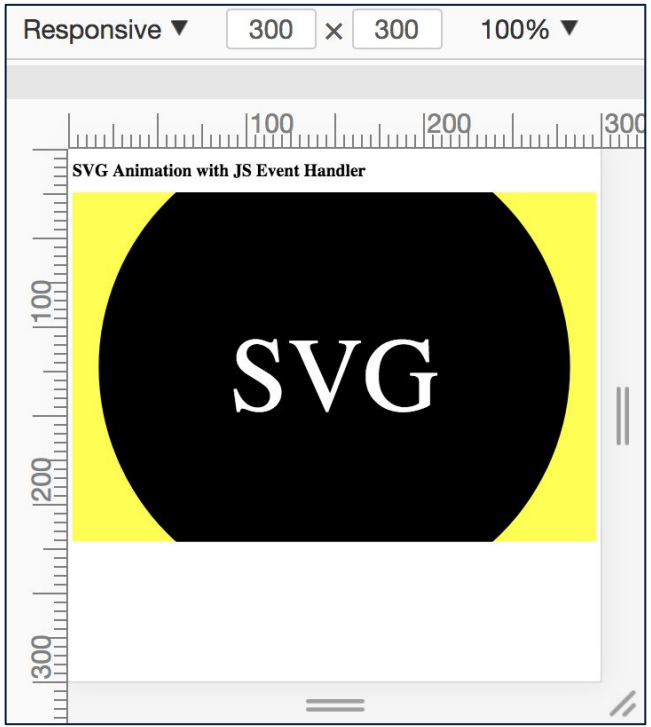
```
<svg viewBox="0 0 120 120">
  <circle cx="60" cy="60" r="30" style="fill: #ff9; stroke: gray;">
    <animate id="c1" attributeName="r" attributeType="XML"
             begin="0s" dur="4s" from="30" to="0" fill="freeze"/>
  </circle>

  <text text-anchor="middle" x="60" y="60"
        style="visibility: hidden;">
    <set attributeName="visibility" attributeType="CSS"
         to="visible" begin="4.5s" dur="1s" fill="freeze"/>
    All gone!
  </text>
</svg>
```



https://developer.mozilla.org/en-US/docs/Web/SVG/Element/set

Animation in SVG via JavaScript *setInterval*

```
<!doctype html>
<h1>SVG Animation with JS Event Handler</h1>
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle id="svg-circle" cx="150" cy="100" r="80" />
  <text x="150" y="125" font-size="60"
        text-anchor="middle" fill="white">SVG</text>
</svg>
<script>
 const svg_circle = document.getElementById("svg-circle");
 let inc = 60;
 let timer = setInterval( () => {
   inc += 1;
   //  if ( inc === 160 ) clearInterval(timer); // once only
   if ( inc === 160 ) inc = 60;                 // repeat
   svg_circle.setAttribute('r', inc );  // re-render every 100 msec
 }, 100 );
</script>
```



https://wiki.selfhtml.org/wiki/SVG/Tutorials/SVG_mit_JavaScript_animieren

Animation in SVG via JavaScript *requestAnimationFrame*

Verlagerung der Animation in die Rendering Loop

```
<!doctype html>
<h1>SVG Animation with requestAnimationFrame</h1>
<button onclick="stop()">Stop!</button>
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle id="svg-circle" cx="150" cy="100" r="80" />
  <text x="150" y="125" font-size="60" text-anchor="middle"
fill="white">SVG</text>
</svg>
<script>
 const svg_circle = document.getElementById("svg-circle");
 let inc = 60;
 let animation = requestAnimationFrame( step );
 function step( timestamp ){
   inc += 1;
   if ( inc === 160 ) inc = 60; // repeat
   svg_circle.setAttribute('r', inc );
   animation = requestAnimationFrame( step );
 }
```


```
function stop(){
  cancelAnimationFrame( animation );
}
</script>
```

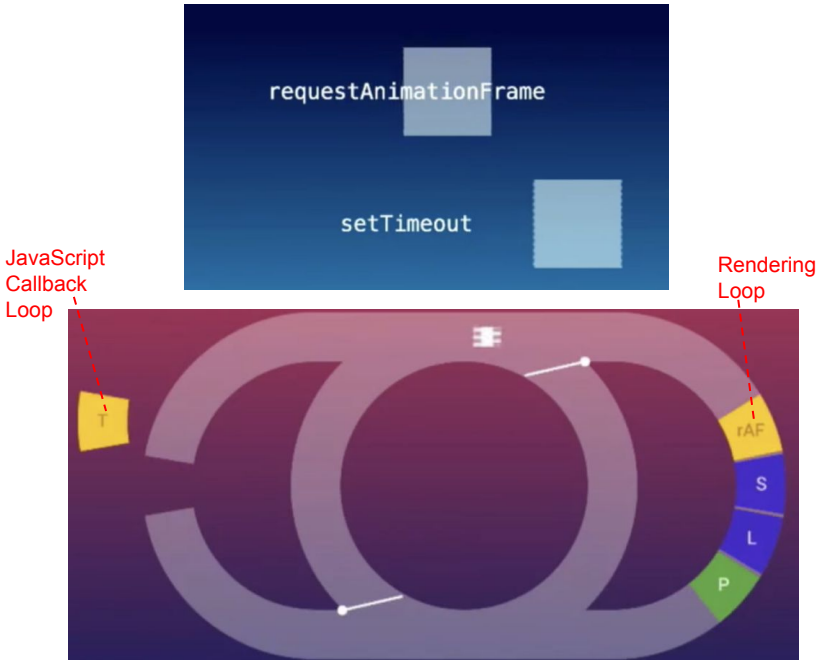
Bessere Animation mit *requestAnimationFrame()*

ist gegenüber *setInterval()* zu bevorzugen, um unnötige Aufrufe zu vermeiden

Ziel: Ruckelfreie Animation, ohne die CPU unnötig zu belasten.

```
<script>
 function myMove() {
   var elem = document.getElementById("animate");
   var pos = 0;
   frame();
   function frame() {
     if (pos > 249) {
       pos = 0;
     } else {
       pos += 5;
       elem.style.left = pos + 'px';
     }
     requestAnimationFrame( frame );
   }
 }
</script>
```

- Aufruf in der Rendering Loop
- besser: 60 Hz Update-Frequenz



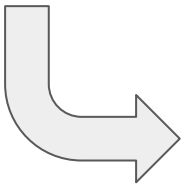
https://www.youtube.com/watch?v=0IsjjMRyIF8&t=389s

Web Animation API

Use The Platform

```css
#alice {
  animation: aliceTumbling infinite 3s linear;
}

@keyframes aliceTumbling {
  0% {
    color: #000;
    transform: rotate(0) translate3D(-50%, -50%, 0);
  }
  30% {
    color: #431236;
  }
  100% {
    color: #000;
    transform: rotate(360deg) translate3D(-50%, -50%, 0);
  }
}
```

- use Browser's standard animation engine
  - used by CSS animations
  - used by CSS transitions
- Web Animation API = API for JS
  - Use the standard animation engine from JavaScript



Element.animate()

JS

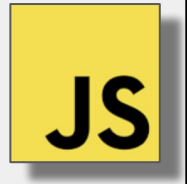
```js
document.getElementById("alice").animate(
  [
    { transform: 'rotate(0) translate3D(-50%, -50%, 0)', color: '#000' },
    { color: '#431236', offset: 0.3},
    { transform: 'rotate(360deg) translate3D(-50%, -50%, 0)', color: '#000' }
  ], 3000);
```

https://developer.mozilla.org/en-US/docs/Web/API/Web_Animations_API/Using_the_Web_Animations_API

SVGElement.animate()

```
<!doctype html>
<style>
  .myrect {
    animation: dance 2s infinite;
  }
  @keyframes dance {
    0% {
      transform: scale(1)
    }
    100% {
      transform: scale(0.1);
    }
  }
</style>
<svg width="250" height="50">
  <title>Shrinking Rectangle</title>
  <rect class="myrect" x="10" y="10" width="200"
        height="20" stroke="black" fill="none">
  </rect>
</svg>
```

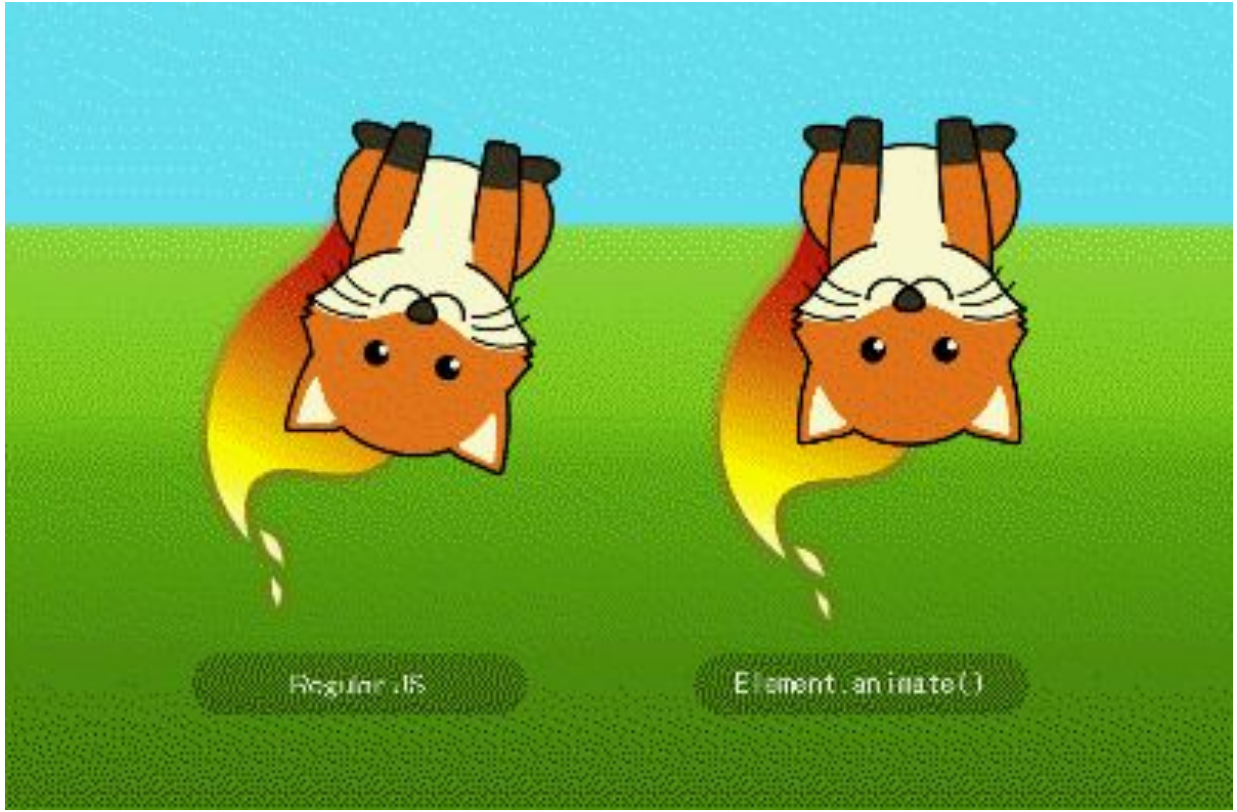
```
<!doctype html>
<svg width="250" height="50">
  <title>Shrinking Rectangle</title>
  <desc>Animating the width attribute of a rectangle</desc>

  <rect id="r" x="10" y="10" width="200" height="20"
                                stroke="black" fill="none">
  </rect>

</svg>
<script>
  const rect = document.getElementById('r');
  rect.animate(
    [
      { transform: 'scale(1)' },
      { transform: 'scale(0.1)' }
    ], {
      duration: 1000,
      iterations: Infinity
    }
  );
</script>
```



Element.animate()

Element.animate() API

https://hacks.mozilla.org/2016/08/animating-like-you-just-dont-care-with-element-animate/



Browser API
Element.animate()

Element.animate()-API

https://hacks.mozilla.org/2016/08/animating-like-you-just-dont-care-with-element-animate/

```
elem.animate(
    { transform: [ 'rotate(0deg)', 'rotate(360deg)' ] },
    { duration: 1000, iterations: Infinity }
);
```

Jede Sekunde eine Rotation animieren,
angefangen bei 0° bis 360°.

"However, if we have an animation that we know doesn't change layout and perhaps doesn't even need redrawing, it should be possible to let someone else take care of adjusting those layers on each frame. As it turns out, browsers already have a process designed precisely for that job — a separate thread or process known as the **compositor** that specializes in **arranging and combining layers**. All we need is a way to tell the compositor the whole story of the animation and let it get to work, leaving the main thread — that is, the part of the browser that's doing everything else to run your app — to forget about animations and get on with life."

Zusammenfassung: Vorteile von SVG

- **Browser-Standard** für Vektor-Grafik
  - exakte Darstellung bei jeder Vergrößerung
  - kleinere Dateien
  - keine Browser Plugins nötig
- HTML-artige Tags für Grafik
- DOM-Struktur für Grafik
- CSS Styling
- JavaScript Event Handler for SVG DOM Events
- SEO friendly
- Accessible

Use

The

Platform

Anwendung: SVG Maps

https://commons.wikimedia.org/wiki/Category:SVG_maps_of_Germany

Media in category "SVG maps of Germany"

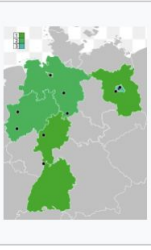
The following 57 files are in this category, out of 57 total.

ARD-Karte-GRUEN.svg
945 × 768; 140 KB

BayreuthMap.svg
1,715 × 1,216; 244 KB

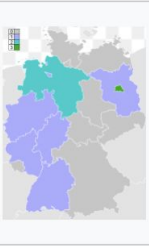
Bonding Standorte.svg
595 × 842; 185 KB

Bundeslaender Doppelnamen.svg
1,073 × 1,272; 705 KB



Bundesvision all cities.svg
592 × 801; 227 KB

Bundesvision Participation.svg
592 × 801; 219 KB

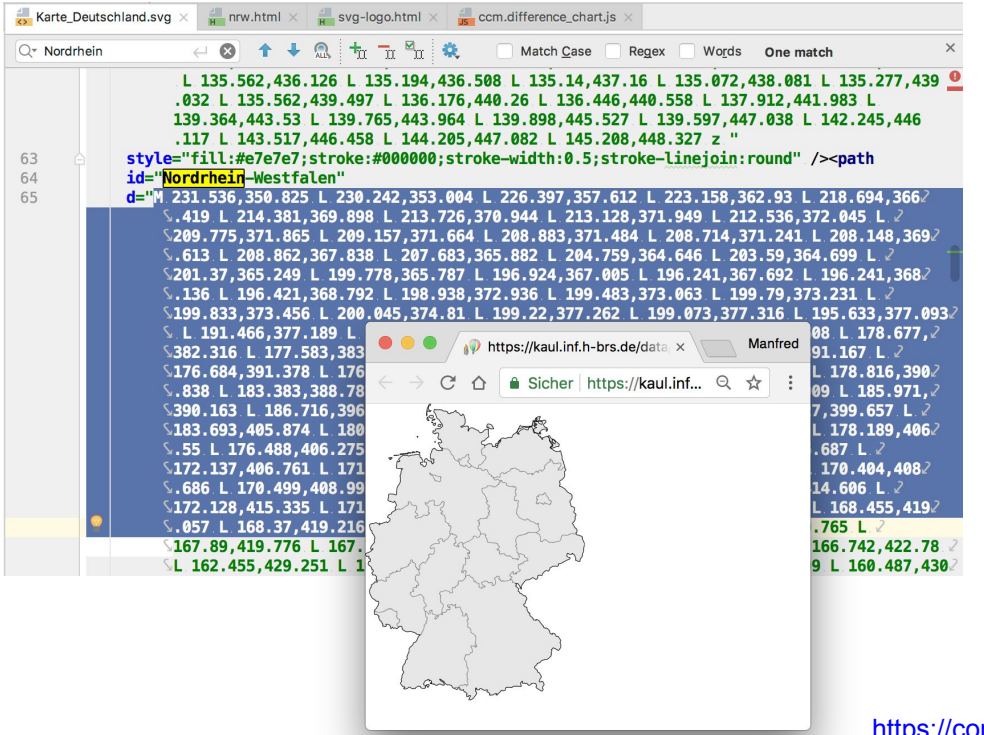


Bundesvision winners map.svg
592 × 801; 216 KB

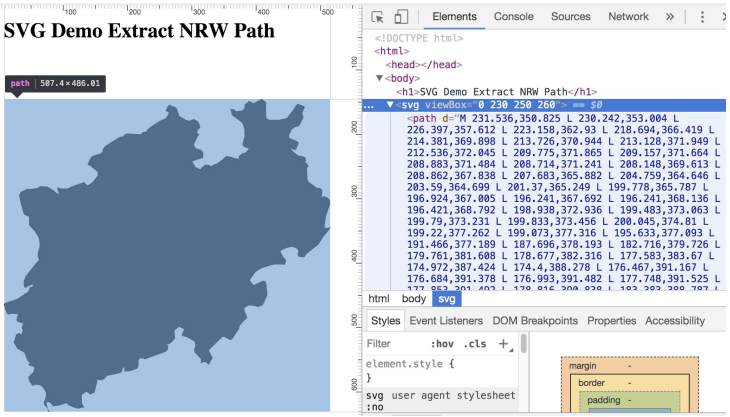
D-A-CH beschriftet.svg
1,257 × 1,514; 715 KB

Using sub-resources

Deutschlandkarte



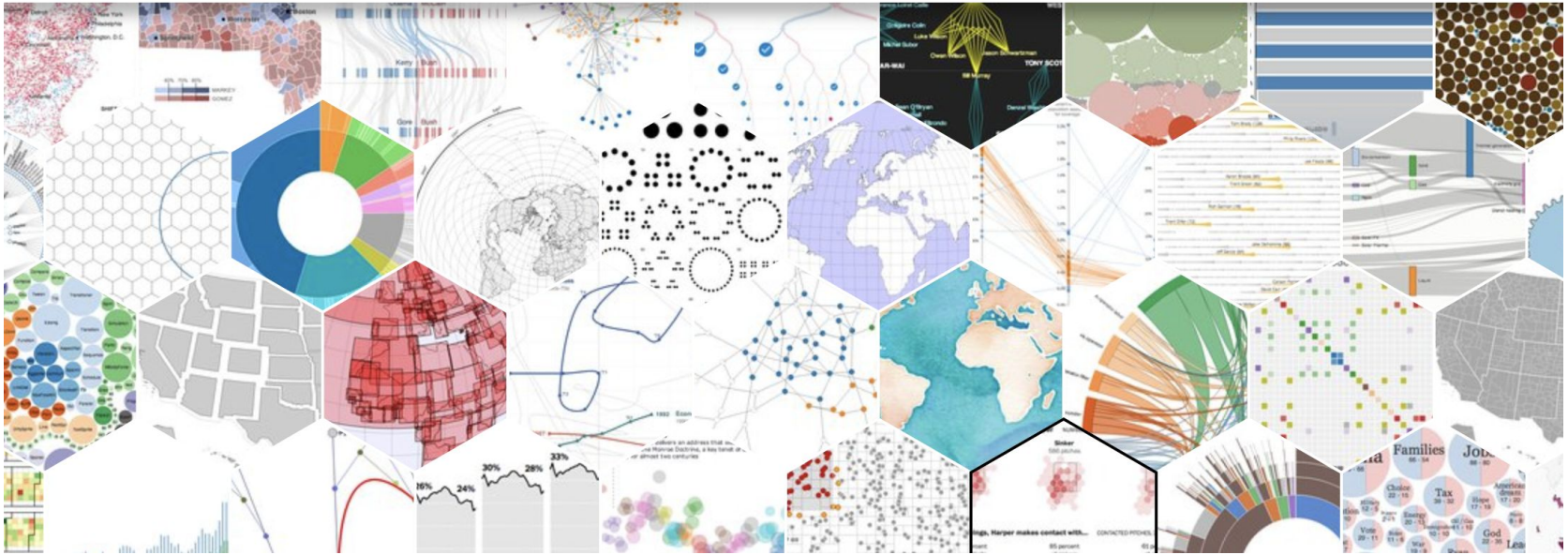
NRW-Karte



https://commons.wikimedia.org/wiki/Category:SVG_maps_of_Germany

d3.js

https://d3js.org/

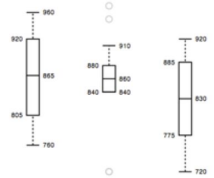


d3.js Gallery

https://github.com/d3/d3/wiki/Gallery

Visual Index

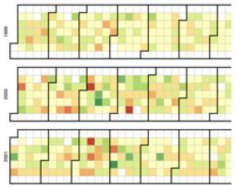
Box Plots



Bubble Chart

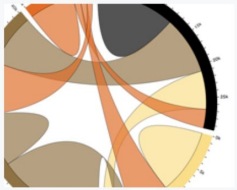
Bullet Charts

Calendar View

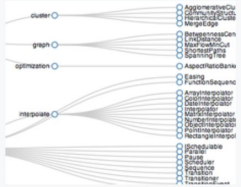


Non-contiguous Cartogram

Chord Diagram



Dendrogram



Force-Directed Graph

Circle Packing

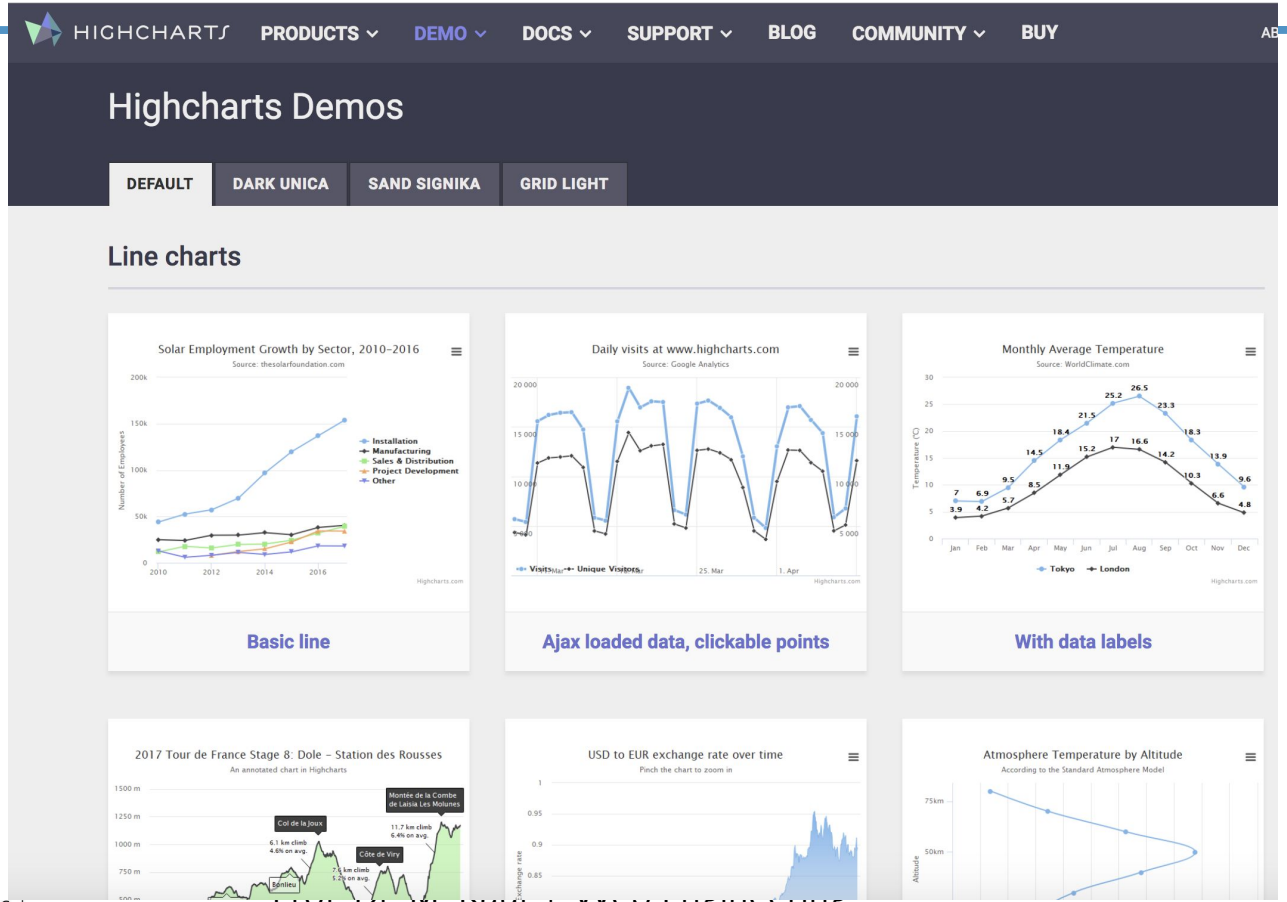
Population Pyramid

2000

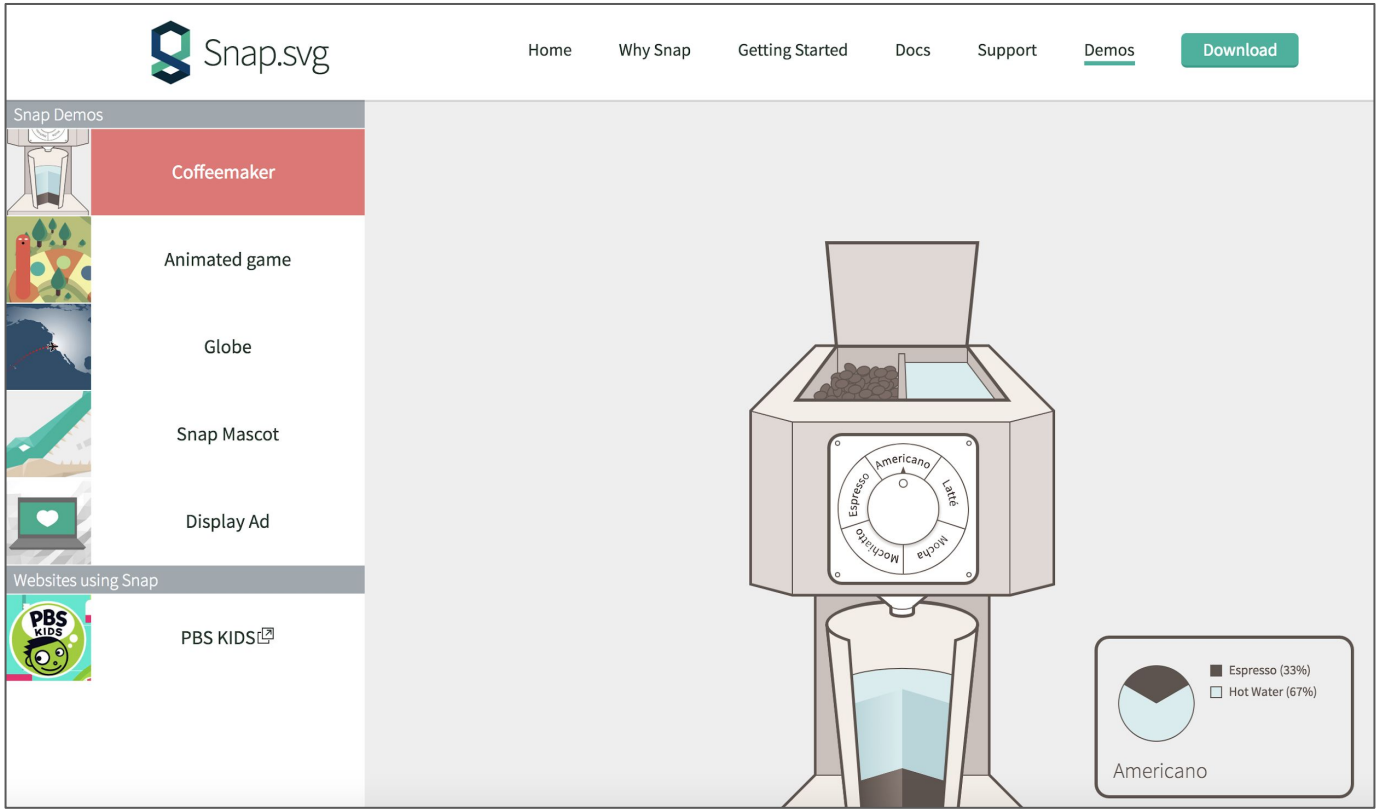
Stacked Bars

Streamgraph

Highcharts.com

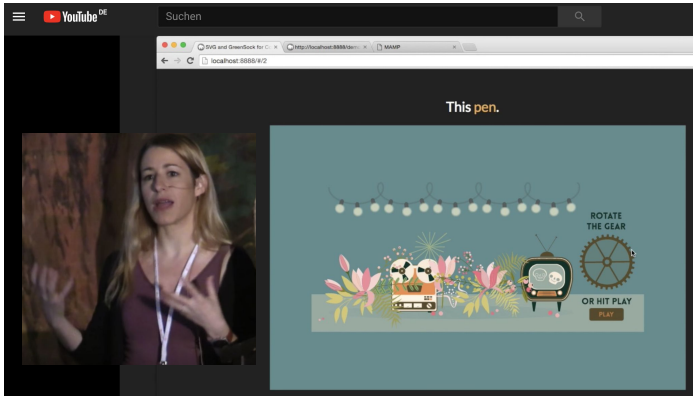


Snap SVG

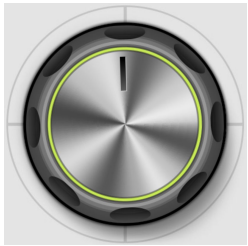


http://snapsvg.io/

Animation Libraries
GreenSock



https://www.youtube.com/watch?v=ZNukcHhpSXg



https://greensock.com/draggable

https://greensock.com/products/

Anwendung: Komplette Programmierumgebung in SVG

Smalltalk

Self

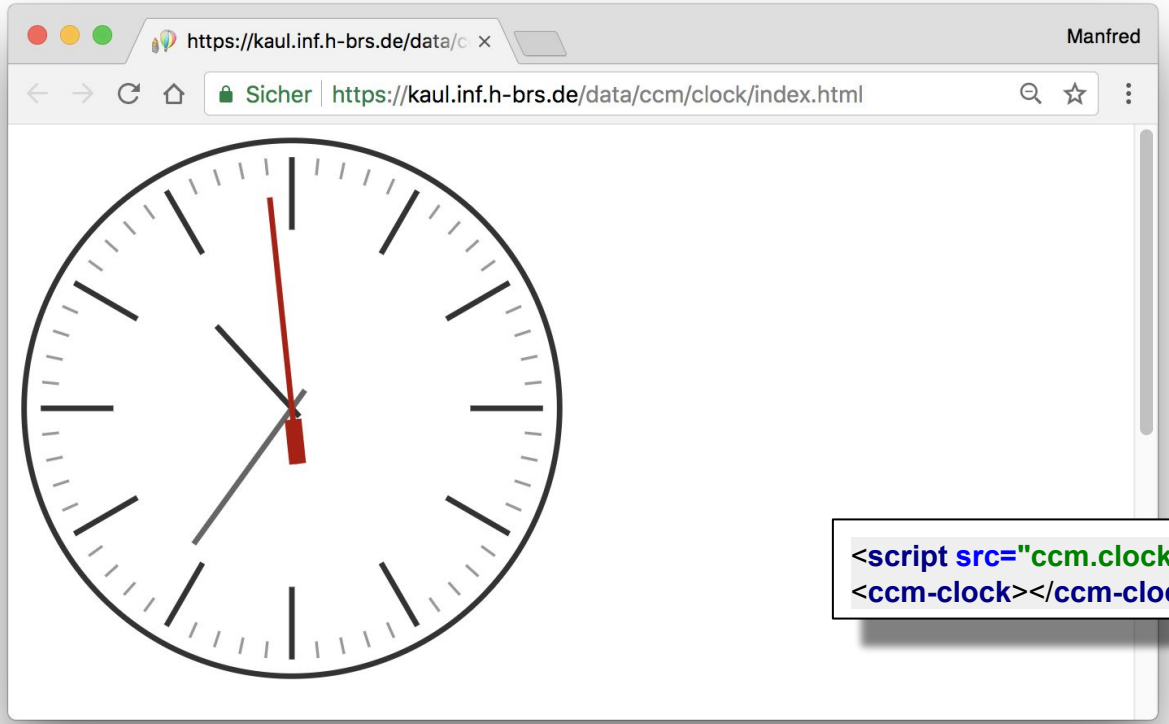
Squeak

Morphic

Lively

http://lively-kernel.org/

Analog-Uhr mit SVG

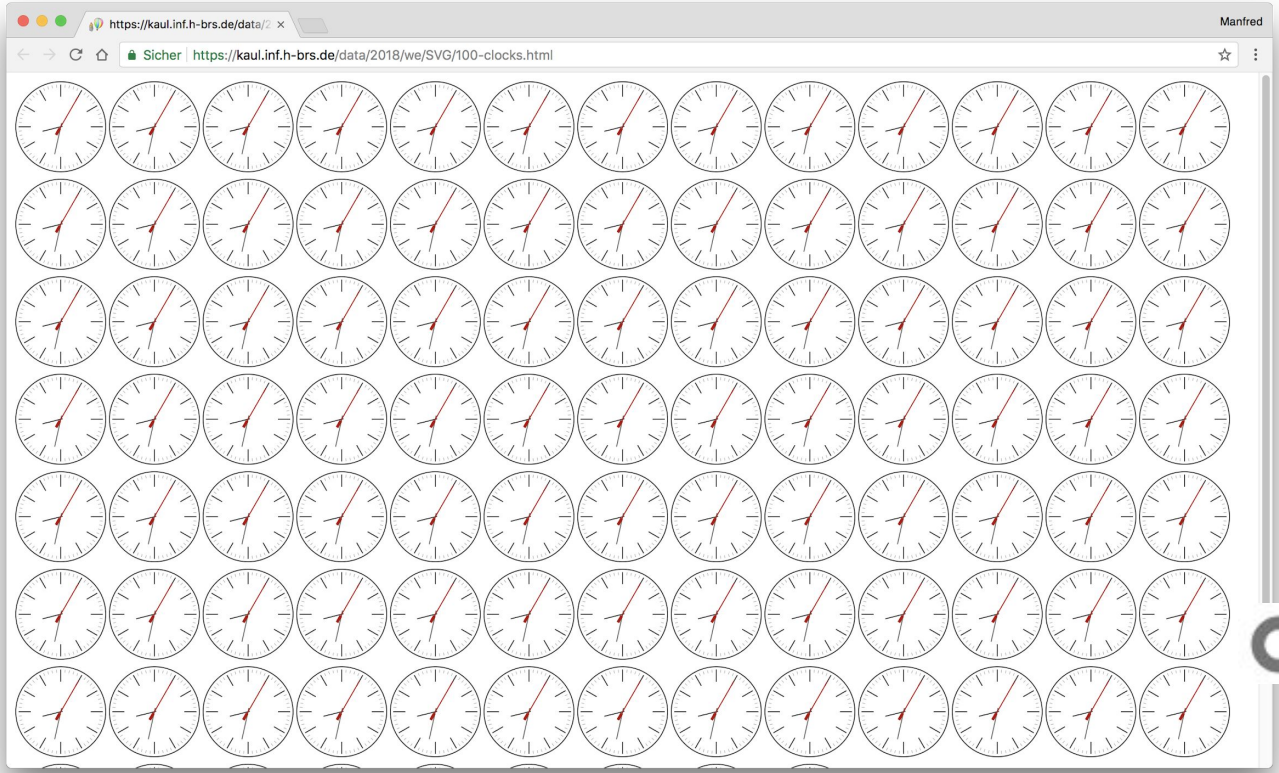



```
<script src="ccm.clock.js"></script>
<ccm-clock></ccm-clock>
```

Webseite mit 100 Uhren

https://kaul.inf.h-brs.de/data/2018/we/SVG/100-clocks.html

- 100 Uhren laufen noch synchron.
- Bei 1000 Uhren kommt es zu Sprüngen des Sekundenzeigers von ca. 5 sec.

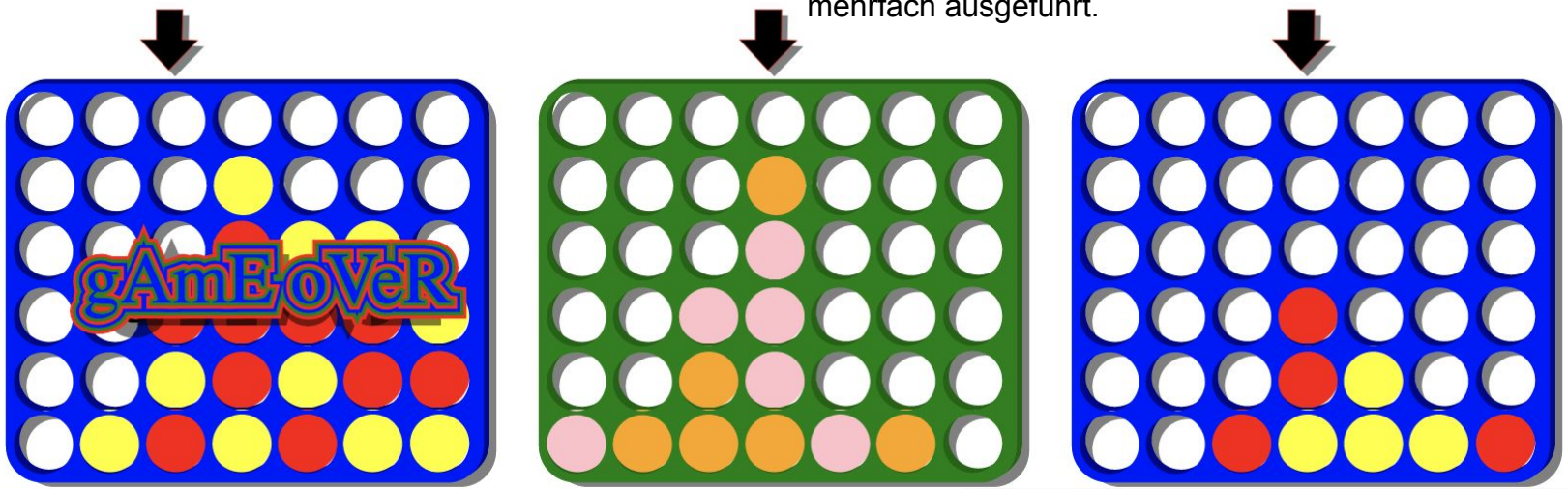


2D-Spiele mit SVG: Vier gewinnt!

Vier gewinnt! Vier gewinnt! Vier gewinnt!

Simultan-Spiel

Code wird nur einmal geladen,
aber mit unterschiedlichen Konfigurationen
mehrfach ausgeführt.

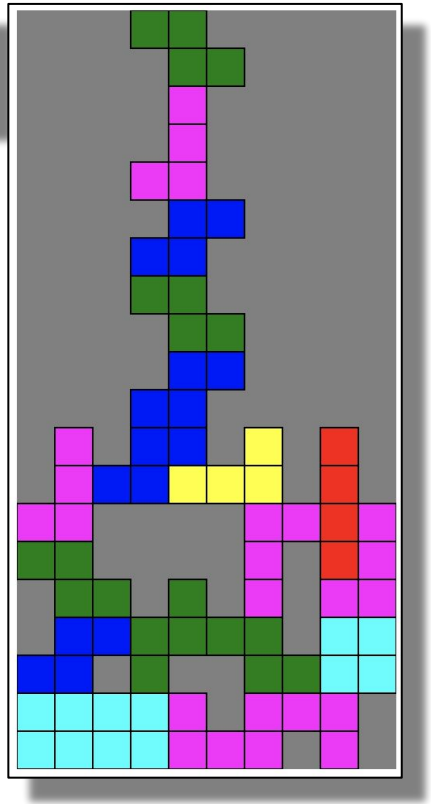


computer wins ! score:1/0

```
<script src="ccm.connect4.js"></script>
<ccm-connect4></ccm-connect4>
```

Tetris

```
<script src="ccm.tetris.js"></script>
<ccm-tetris></ccm-tetris>
```

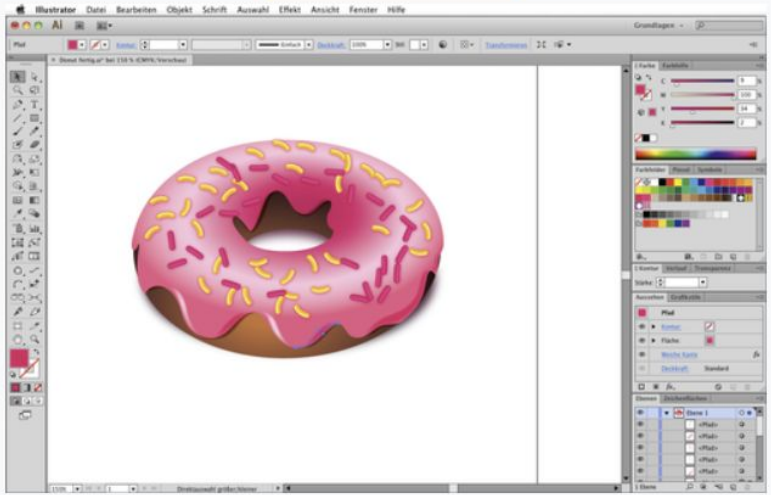


Historisches:

https://www.colinfahey.com/tetris/tetris.html

Adobe Illustrator

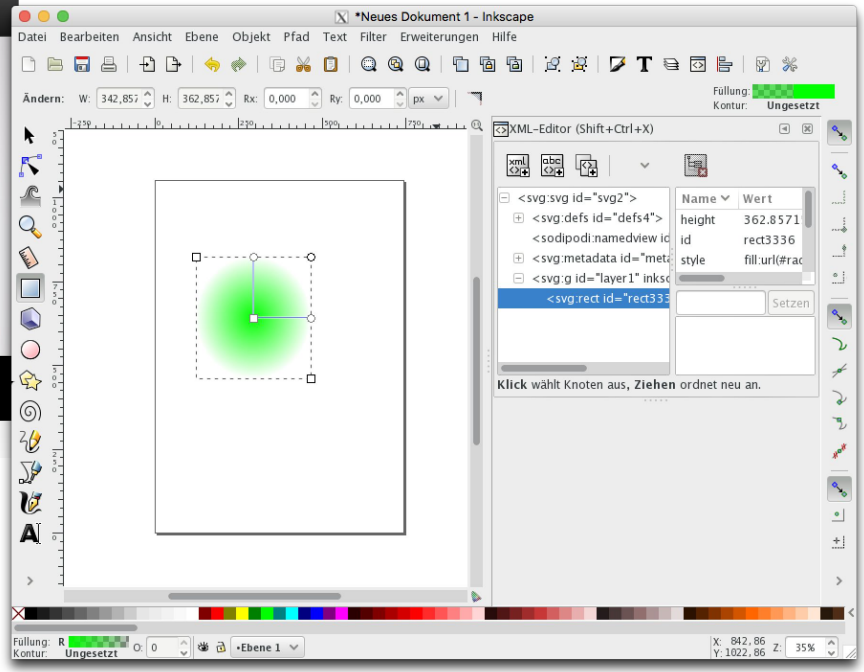
Adobe Illustrator



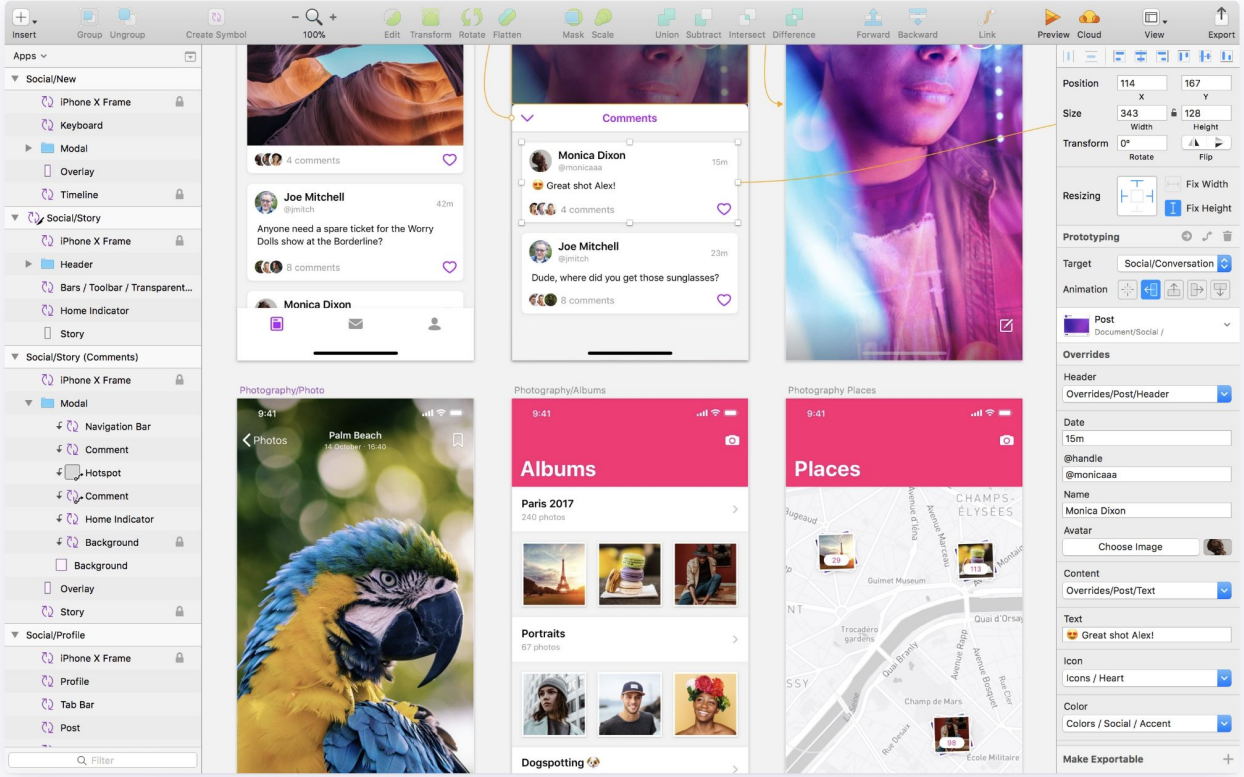
Adobe Illustrator CS6 (macOS, mit Plugins)

Inkscape

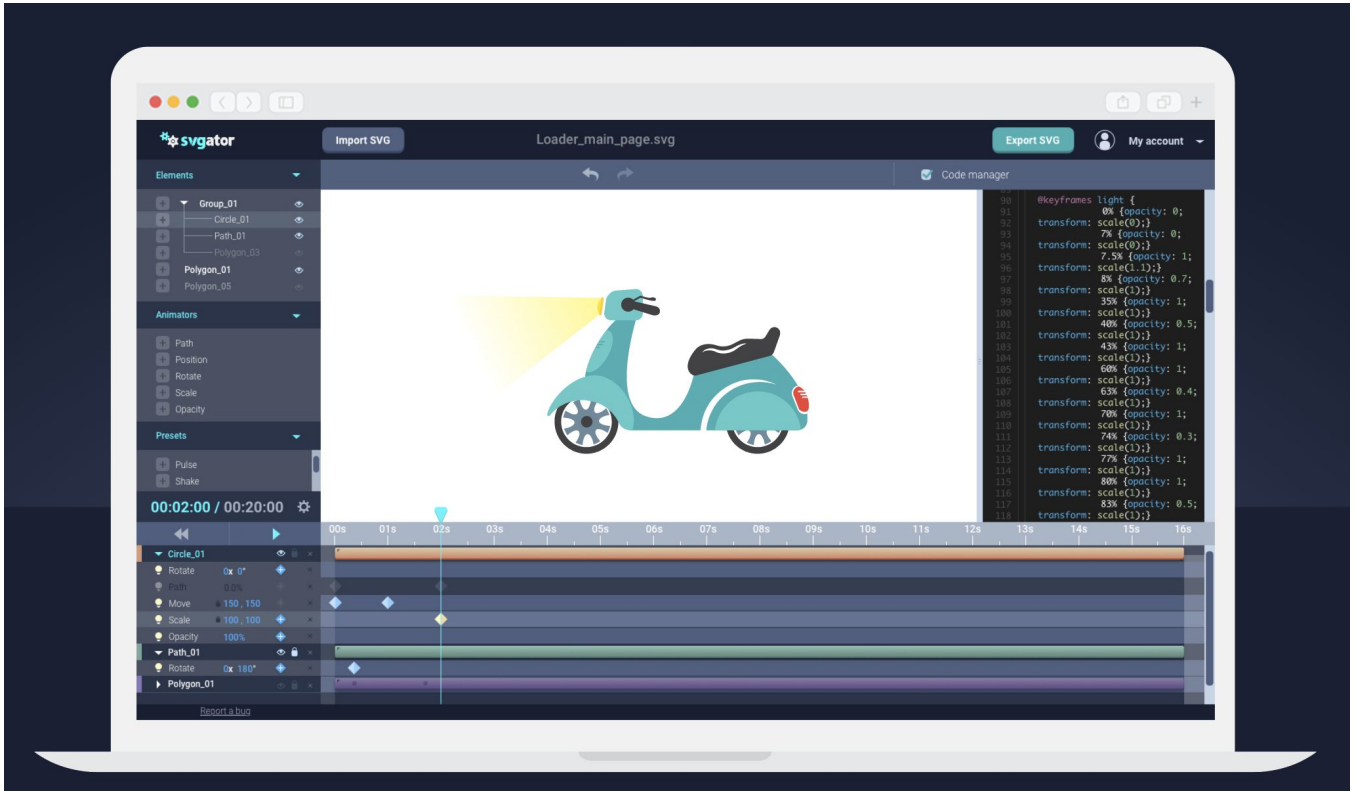
https://inkscape.org



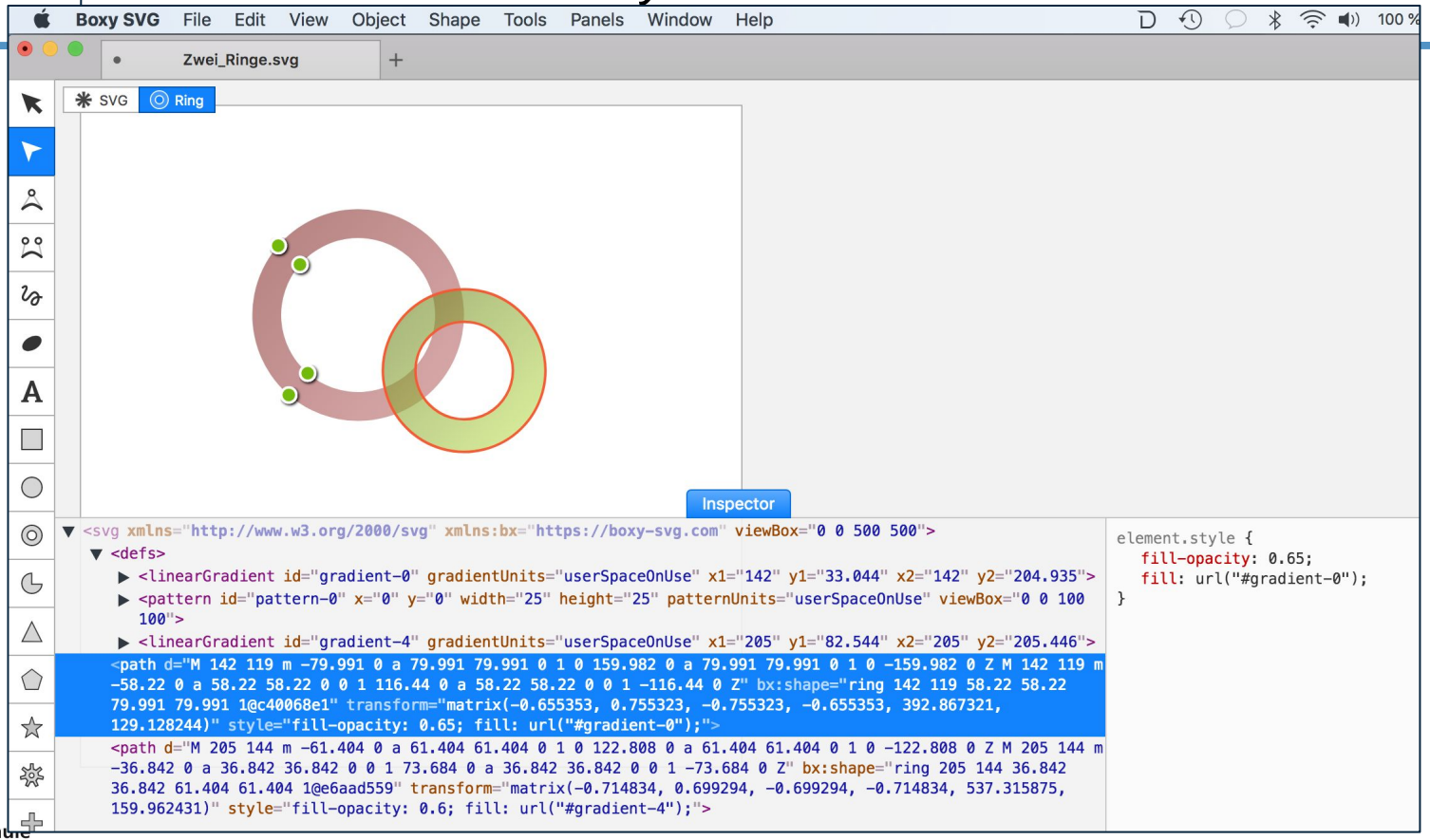
Sketch


Insert
Group
Ungroup
Create Symbol
100%
Edit
Transform
Rotate
Flatten
Mask
Scale
Union
Subtract
Intersect
Difference
Forward
Backward
Link
Preview
Cloud
View
Export
Apps
Social/New
iPhone X Frame
Keyboard
Modal
Overlay
Timeline
Social/Story
iPhone X Frame
Header
Bars / Toolbar / Transparent...
Home Indicator
Story
Social/Story (Comments)
iPhone X Frame
Modal
Navigation Bar
Comment
Hotspot
Comment
Home Indicator
Background
Background
Overlay
Story
Social/Profile
iPhone X Frame
Profile
Tab Bar
Post
Filter
4 comments
Joe Mitchell
@jmitch
42m
Anyone need a spare ticket for the Worry Dolls show at the Borderline?
8 comments
Monica Dixon
Comments
Monica Dixon
@monicaaa
15m
Great shot Alex!
4 comments
Joe Mitchell
@jmitch
23m
Dude, where did you get those sunglasses?
8 comments
Photography/Photo
9:41
Photos
Palm Beach
Photography/Albums
9:41
Albums
Paris 2017
240 photos
Portraits
67 photos
Dogspotting
Photography Places
9:41
Places
Position
114
X
167
Y
Size
343
Width
128
Height
Transform
0°
Rotate
Flip
Resizing
Fix Width
Fix Height
Prototyping
Target
Social/Conversation
Animation
Post
Document/Social /
Overrides
Header
Overrides/Post/Header
Date
15m
@handle
@monicaaa
Name
Monica Dixon
Avatar
Choose Image
Content
Overrides/Post/Text
Text
Great shot Alex!
Icon
Icons / Heart
Color
Colors / Social / Accent
Make Exportable

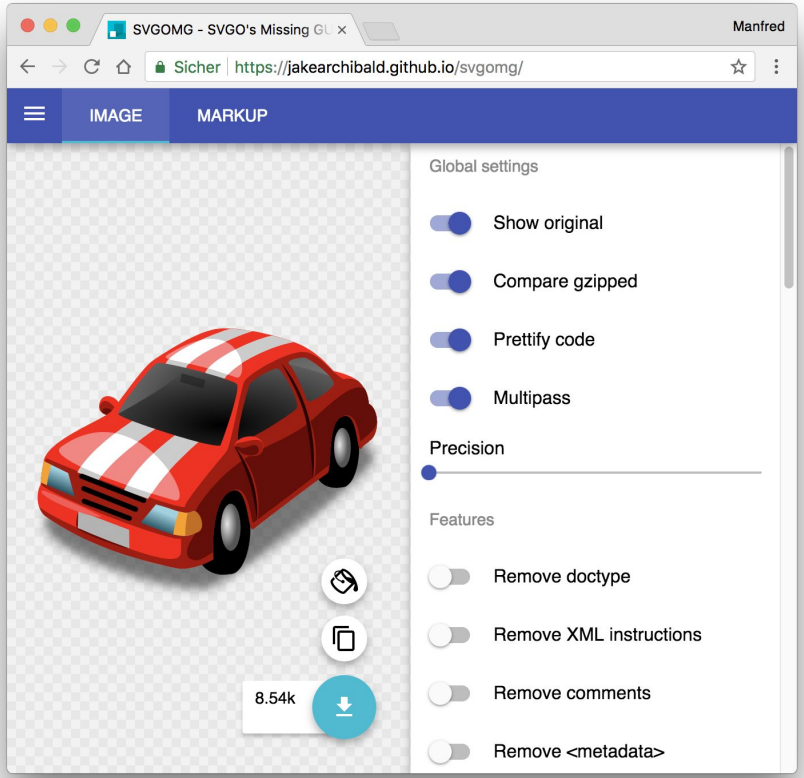
Generierung von Animationen



Boxy SVG

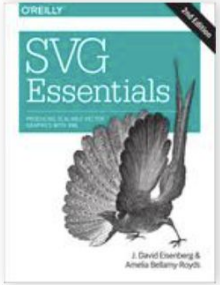

Boxy SVG File Edit View Object Shape Tools Panels Window Help
100 %
Zwei_Ringe.svg
SVG
Ring
Inspector
<svg xmlns="http://www.w3.org/2000/svg" xmlns:bx="https://boxy-svg.com" viewBox="0 0 500 500">
<defs>
<linearGradient id="gradient-0" gradientUnits="userSpaceOnUse" x1="142" y1="33.044" x2="142" y2="204.935">
<pattern id="pattern-0" x="0" y="0" width="25" height="25" patternUnits="userSpaceOnUse" viewBox="0 0 100 100">
<linearGradient id="gradient-4" gradientUnits="userSpaceOnUse" x1="205" y1="82.544" x2="205" y2="205.446">
<path d="M 142 119 m -79.991 0 a 79.991 79.991 0 1 0 159.982 0 a 79.991 79.991 0 1 0 -159.982 0 Z M 142 119 m -58.22 0 a 58.22 58.22 0 0 1 116.44 0 a 58.22 58.22 0 0 1 -116.44 0 Z" bx:shape="ring 142 119 58.22 58.22 79.991 79.991 1@c40068e1" transform="matrix(-0.655353, 0.755323, -0.755323, -0.655353, 392.867321, 129.128244)" style="fill-opacity: 0.65; fill: url("#gradient-0");">
<path d="M 205 144 m -61.404 0 a 61.404 61.404 0 1 0 122.808 0 a 61.404 61.404 0 1 0 -122.808 0 Z M 205 144 m -36.842 0 a 36.842 36.842 0 0 1 73.684 0 a 36.842 36.842 0 0 1 -73.684 0 Z" bx:shape="ring 205 144 36.842 36.842 61.404 61.404 1@e6aad559" transform="matrix(-0.714834, 0.699294, -0.699294, -0.714834, 537.315875, 159.962431)" style="fill-opacity: 0.6; fill: url("#gradient-4");">
element.style {
fill-opacity: 0.65;
fill: url("#gradient-0");
}

SVG Minimizer



https://jakearchibald.github.io/svgomg/

Literatur



SVG Essentials, 2nd Edition

★★★★★ 5 REVIEWS

by Amelia Bellamy-Royds, J. David Eisenberg

Publisher: O'Reilly Media, Inc.

Release Date: October 2014

ISBN: 9781449374358

Topic: Engineering

- https://www.safaribooksonline.com/library/view/svg-essentials-2nd/9781491945308/

Nachschlagewerke

- W3C
  - https://www.w3.org/TR/SVG2/
- WHATWG
  - https://html.spec.whatwg.org/multipage/embedded-content-other.html#svg-0
- Mozilla
  - https://developer.mozilla.org/en-US/docs/Web/SVG
- selfhtml
  - https://wiki.selfhtml.org/wiki/SVG
- Wikipedia
  - https://de.wikipedia.org/wiki/Scalable_Vector_Graphics
  - https://en.wikipedia.org/wiki/Scalable_Vector_Graphics

Tutorials

W3Schools

https://www.w3schools.com/graphics/svg_intro.asp

selfhtml

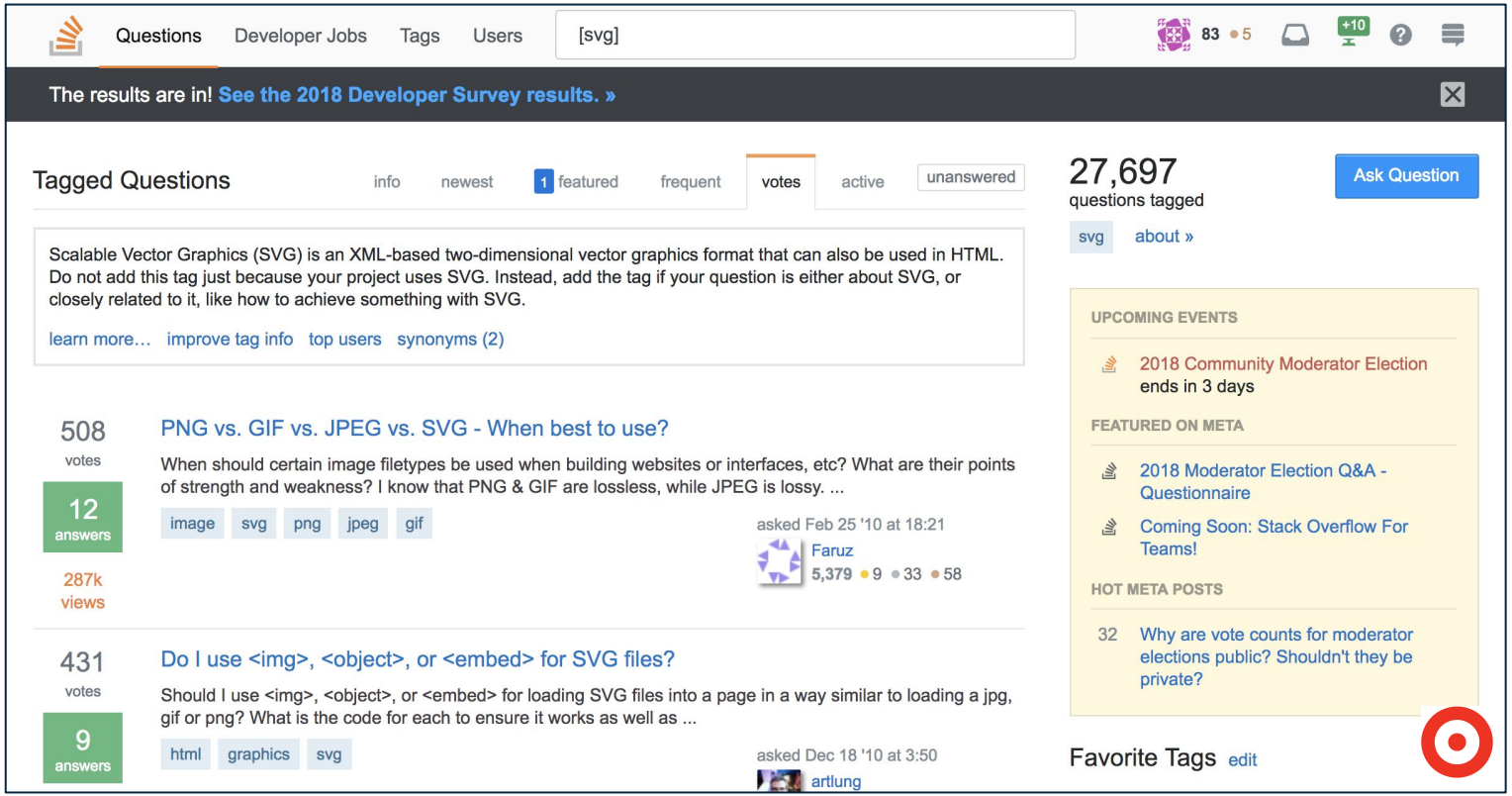
https://wiki.selfhtml.org/wiki/SVG

MediaEvent

https://www.mediaevent.de/tutorial/svg.html

Problemlösungsforum

https://stackoverflow.com/questions/tagged/svg



1. Erstellen Sie von Hand ohne Tools mit SVG folgende Grafik