

# 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."*

SVG by Mozilla Contributors is licensed under CC-BY-SA 2.5.



Use  
The  
Platform



# Inline SVG wie HTML nutzen



```
<!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>
```

## Inline 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`Tagged Template Literal
      <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}
    }
  }
}

```



```

    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

```
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}  
    }  
}
```



SVG-Koordinatensystem identisch mit dem DOM-Koordinatensystem

```
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

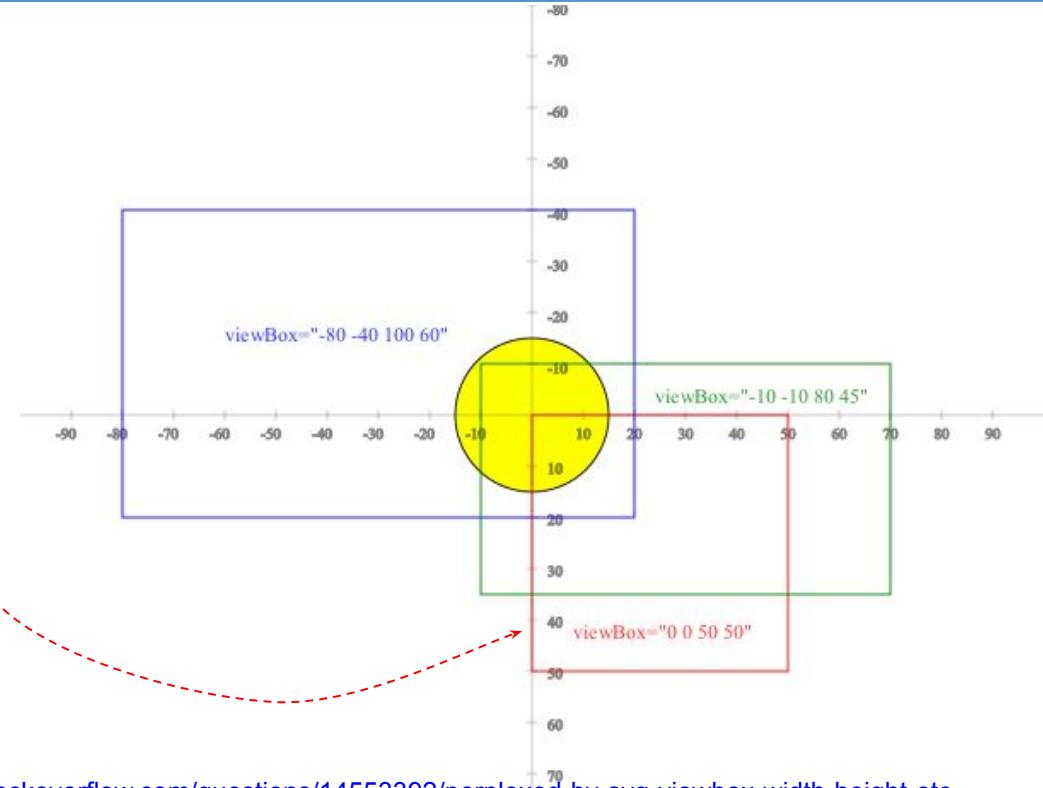
Bewege den Kreis mit der Maus

# 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](#)





- Das SVG-Wurzelement 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">`



# Transformation von DOM-Koordinaten ins SVG-Koordinaten

## Cumulative Transformation Matrix (CTM)

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

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

`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



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





	INLINE	IMAGE	OBJECT
STANDALONE FILE	✗	✓	✓
ANIMATIONS	✓	✓	✓
USER INTERACTIONS	✓	✗	✓
JAVASCRIPT	✓	✗	✓
 <b>CONTROLLABLE FROM OUTSIDE</b>	✓	✗	✗

SVG von JS aus steuern

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



# Die wichtigsten 7 SVG-Elemente

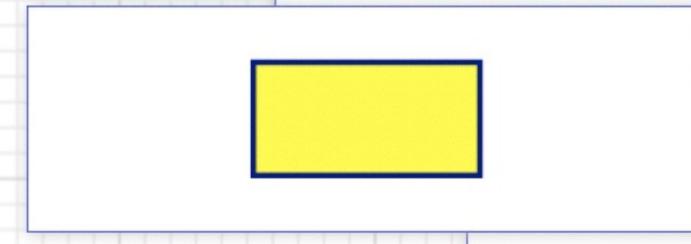
**<text>**

Hello, out there

**<ellipse>**



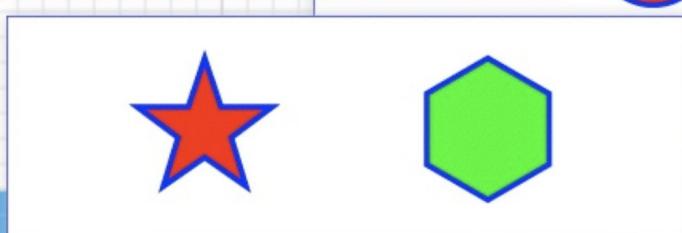
**<rect>**



**<circle>**



**<line>**



**<polygon>**

**<path>**



<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

F

G

<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>

<font-foreignObject>

<foreignObject>

<feFuncA>

<feFuncB>

<feFuncG>

<feFuncR>

<feGaussianBlur>

<glyph>

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V — Z

<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>

<font-foreignObject>

<foreignObject>

<feFuncA>

<feFuncB>

<feFuncG>

<feFuncR>

<feGaussianBlur>

<glyphRef>

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V — Z

<hatch>

<hatchpath>

<hkern>

<iframe>

<image>

<line>

<linearGradient>

<marker>

<mask>

<mesh>

<meshgradient>

<meshpatch>

<meshrow>

<metadata>

<missing-glyph>

<mpath>

<path>

<pattern>

<polygon>

<polyline>

<script>

<set>

<solidcolor>

<stop>

<svg>

<switch>

<symbol>

<text>

T

<textPath>

<title>

<tref>

<tspan>

<use>

<unknown>

<video>

<view>

<vkern>

14

[Home](#)[HTML](#)[CSS](#)[JAVASCRIPT](#)[SQL](#)[PHP](#)[BOOTSTRAP](#)[HOW TO](#)[MORE ▾](#)[REFERENCES ▾](#)[EXAM](#)[HTML Graphics](#)[Graphics HOME](#)[Google Maps](#)[Maps Intro](#)[Maps Basic](#)[Maps Overlays](#)[Maps Events](#)[Maps Controls](#)[Maps Types](#)[Maps Reference](#)[SVG Tutorial](#)[SVG Intro](#)

## SVG Tutorial

[◀ Previous](#)

SVG stands for Scalable Vector Graphics.

SVG defines vector-based graphics in XML format.

### Examples in Each Chapter

With our "Try it Yourself" editor, you can edit the SVG, and click on a button to view th

[SELFHTML](#)[Wiki](#)[Forum](#)[Blog](#)[Lesen](#) [Fragen](#)

selfhtml

Die Energie des Verstehens

[ÜBERSICHT](#)[Wie fange ich an?](#)[HTML](#)[CSS](#)[JavaScript](#)[Referenz](#)[OFFLINE-WIKI](#)[Installieren](#)

## SVG

Das auf [XML](#) basierende **SVG** (Scalable Vecto  
zweidimensionaler Vektorgrafiken).

### Einstieg in SVG

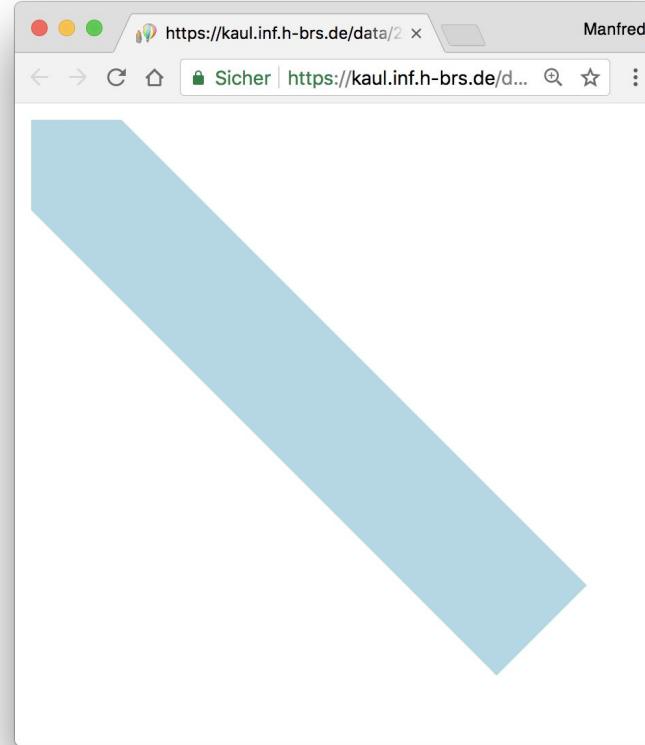
- Warum SVG nutzen?
- SVG in Webseiten einbinden
- Grundformen
- SVG mit CSS stylen
- SVG in responsiven Webseiten
- Grundgerüst eines SVG-Dokuments



# 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>

```
1 <svg xmlns="http://www.w3.org/2000/svg"  
2   width="500" height="40" viewBox="0 0 500 40">  
3  
4   <text x="0" y="35" font-family="Verdana" font-size="35">  
5     Hello, out there  
6   </text>  
7 </svg>
```

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

## Result

*No word wrap*

Hello, out there





# 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 imperdierit eros. Aliquam
    erat volutpat.
  </div>
</foreignObject>
</svg>
```

kein Zeilenumbruch

My **cat**  
is

*Grumpy*

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed mollis mollis mi ut ultricies. Nullam magna ipsum, porta vel dui convallis, rutrum imperdierit eros. Aliquam erat volutpat.*

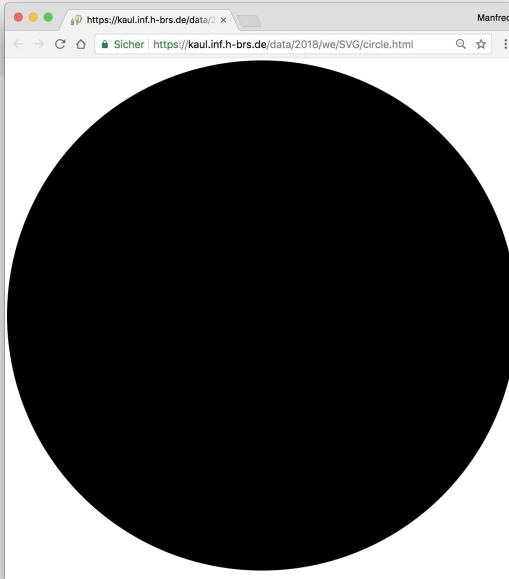
HTML-Zeilenumbruch

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.



# 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>
```



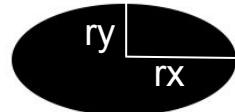
<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/circle>



# 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

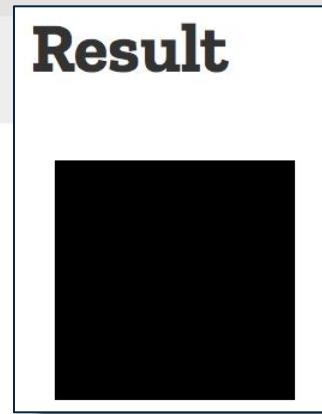


<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/ellipse>

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

# SVG Element <rect>

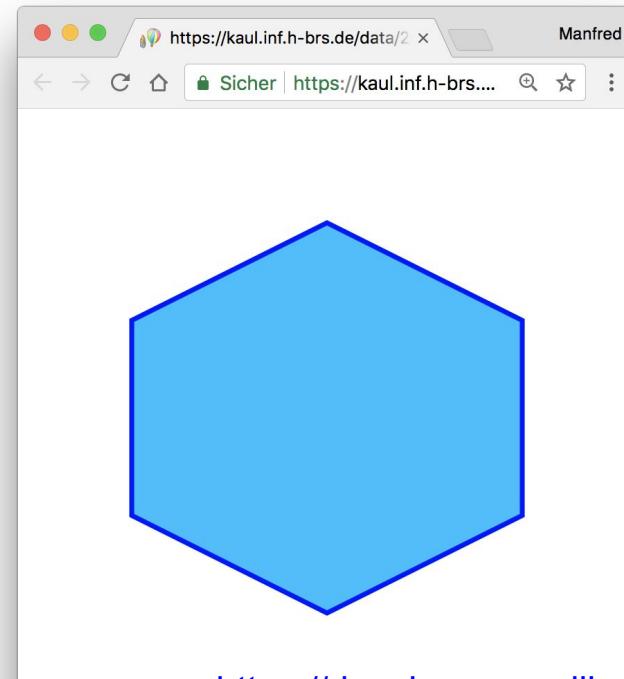
```
1  <svg width="120" height="120" viewBox="0 0 120 120"  
2    xmlns="http://www.w3.org/2000/svg">  
3  
4    <rect x="10" y="10" width="100" height="100"/>  
5  </svg>
```



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

# SVG Element <polygon>

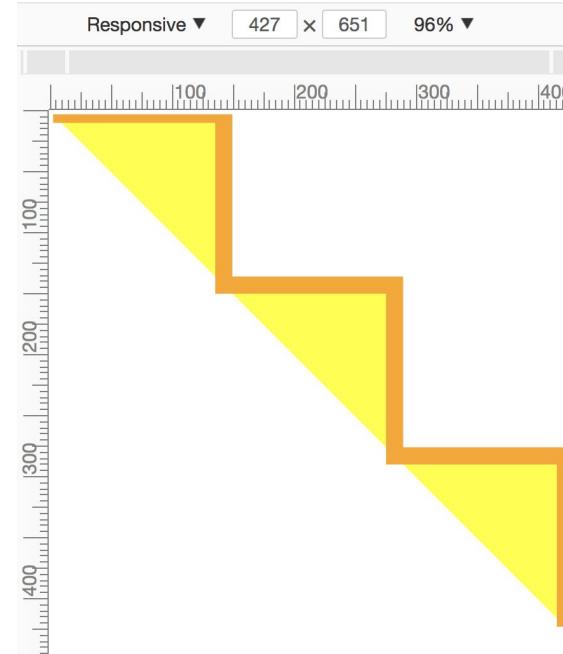
```
<svg viewBox="0 0 120 120">
  <polygon fill="DeepSkyBlue"
    points="60,20 100,40
      100,80 60,100
      20,80 20,40"
    stroke="Blue"
    stroke-width="1">
  </polygon>
</svg>
```



<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>
```

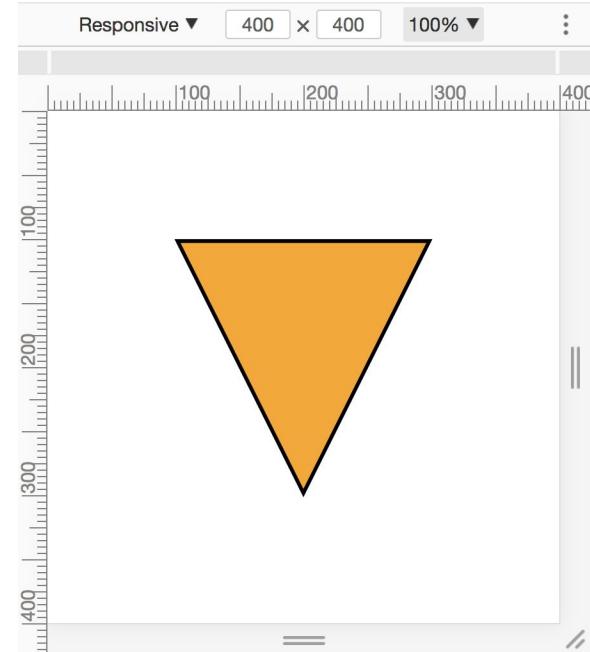


<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/polyline>



# 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>
```

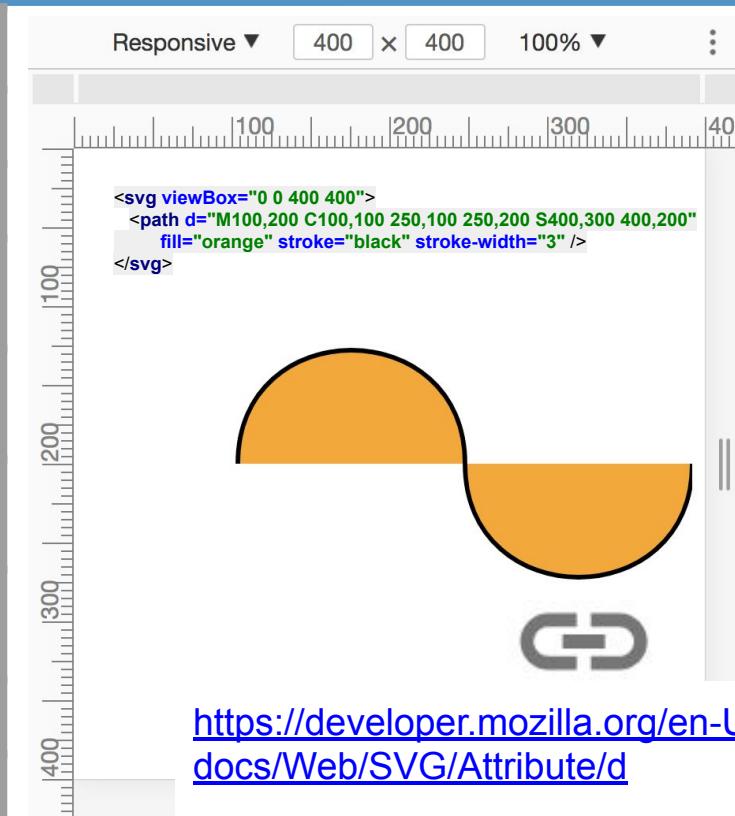


<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/path>



# 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

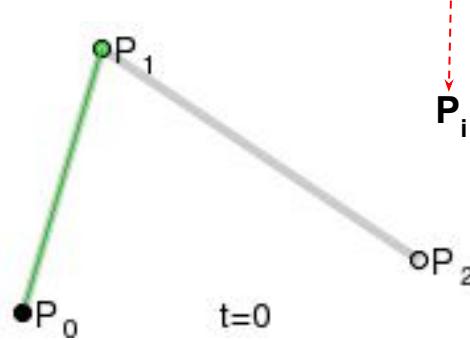


<https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/d>

# Bezierkurven

Eine quadratische Bézierkurve ist der Pfad, der durch die Funktion  $\mathbf{b}(t)$  wie folgt beschrieben wird:

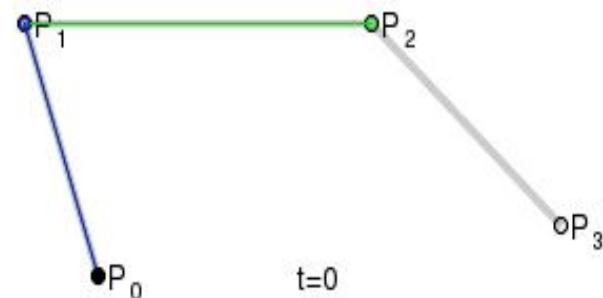
$$\mathbf{b}(t) = \sum_{i=0}^2 \binom{2}{i} t^i (1-t)^{2-i} \mathbf{b}_i$$



<https://de.wikipedia.org/wiki/B%C3%A4zierkurve>

Eine kubische Bézierkurve ist der Pfad, der durch die Funktion  $\mathbf{b}(t)$  wie folgt beschrieben wird:

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

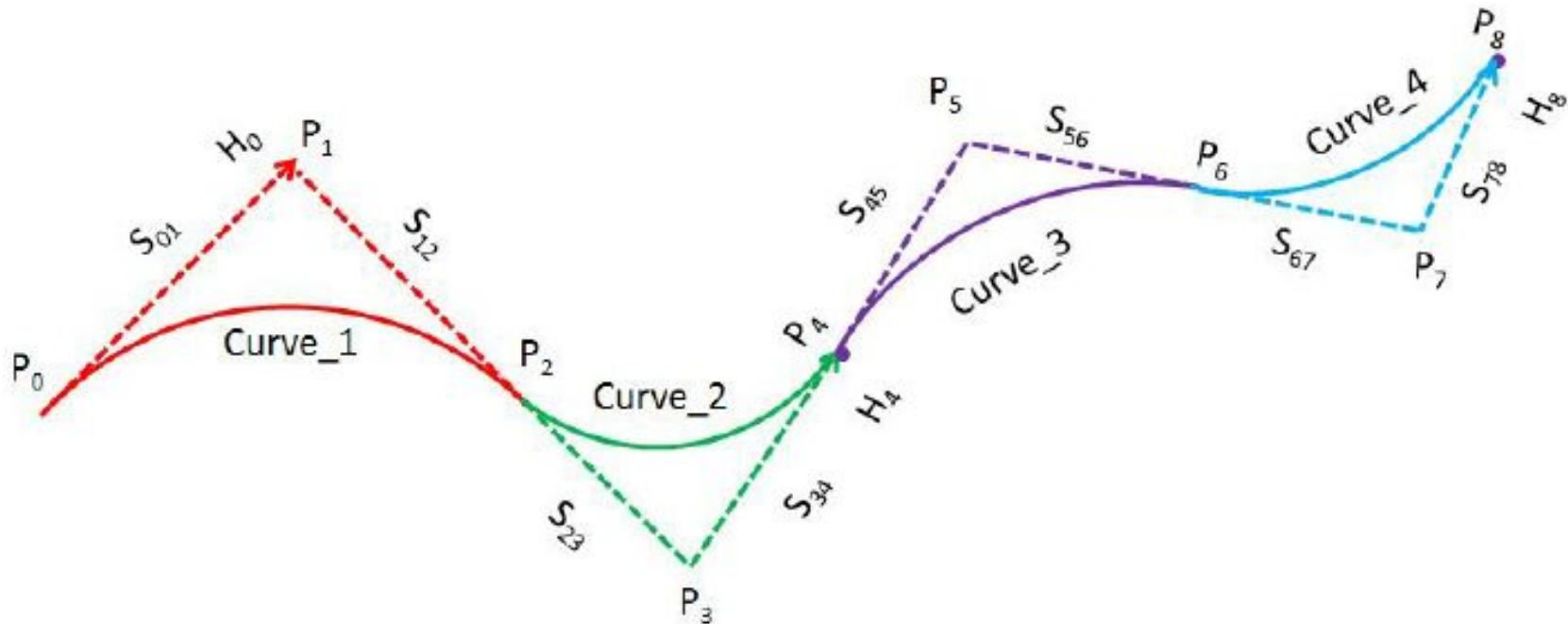


<https://de.wikipedia.org/wiki/B%C3%A4zierkurve>

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



# Weicher Übergang zwischen Bezierkurven



# SVG Definitionen: <defs> und <use>

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

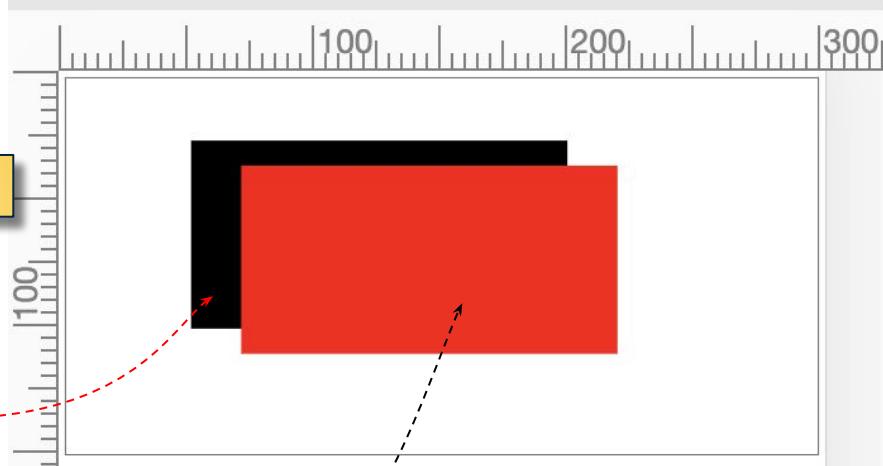
  <defs>
    <rect id="box" x="50" y="25" width="150" height="75"></rect>
  </defs>

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

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

Was innerhalb von <defs> steht, wird nicht gezeichnet

Translation um (20,10)

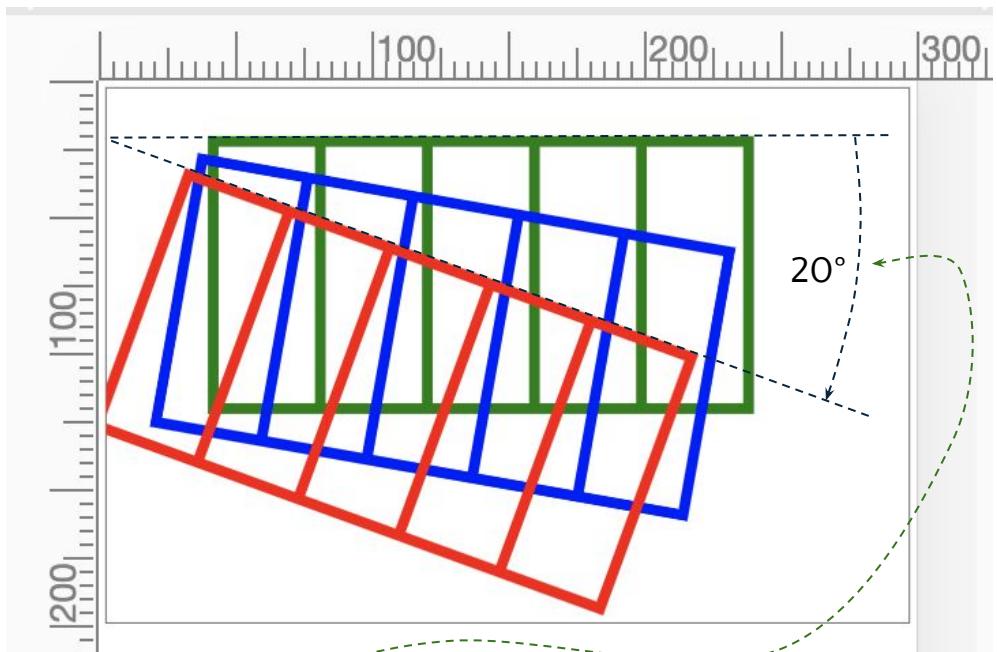


<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/defs>

<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>
```



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

- **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 Werten < 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/>



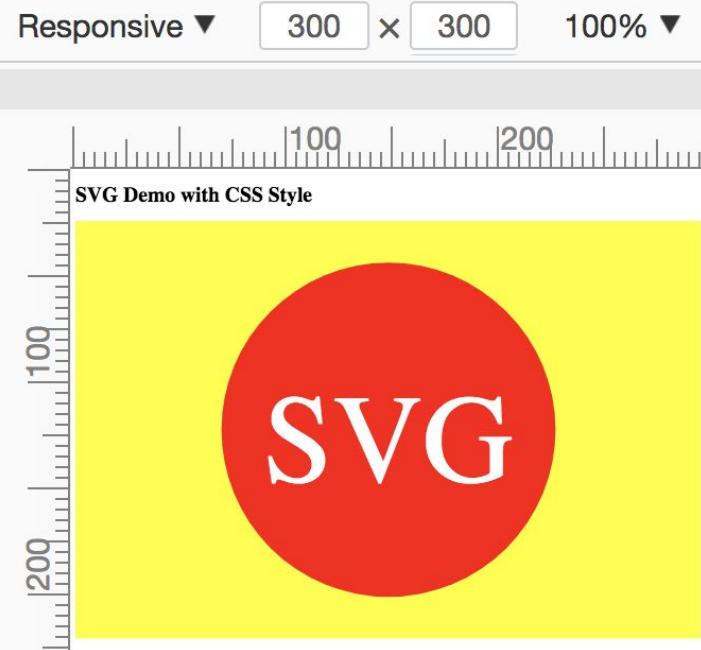
# CSS styling via SVG Element <style>

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

SVG



SVG



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



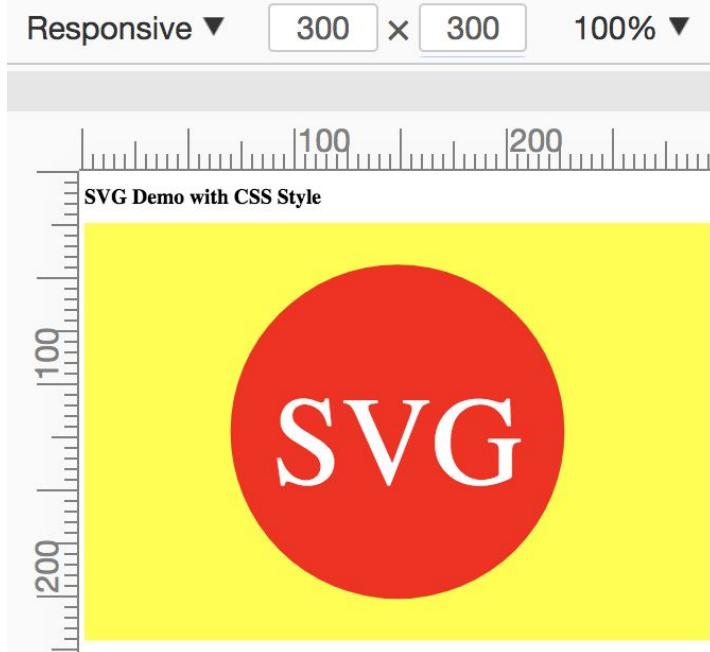
# CSS styling via HTML Element `<style>`

```
<!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>
```

HTML

```
<svg viewBox="0 0 300 200"> SVG
  <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="60px"
        text-anchor="middle" fill="white">SVG</text>
</svg>
```





# SVG in Chrome Devtools

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The main view displays an 'SVG Demo' page featuring a yellow background with a green circle containing the text 'SVG'. A callout box highlights the text node with the dimensions '192.74×110.54'. The DevTools sidebar shows the following HTML code:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>SVG Demo</h1>
    <svg viewBox="0 0 300 200">
      <rect width="100%" height="100%" fill="yellow"></rect>
      <circle cx="150" cy="100" r="80" fill="green"></circle>
      ...
      <text x="150" y="125" font-size="30" text-anchor="middle" fill="white">SVG</text>
    </svg>
  </body>
</html>
```

The 'text' tab is selected in the bottom navigation bar. The console panel shows the following log entries:

```
> document.querySelector('svg')
<  ><svg viewBox="0 0 300 200">...</svg>
```



# 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>
```





```
<!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>
```

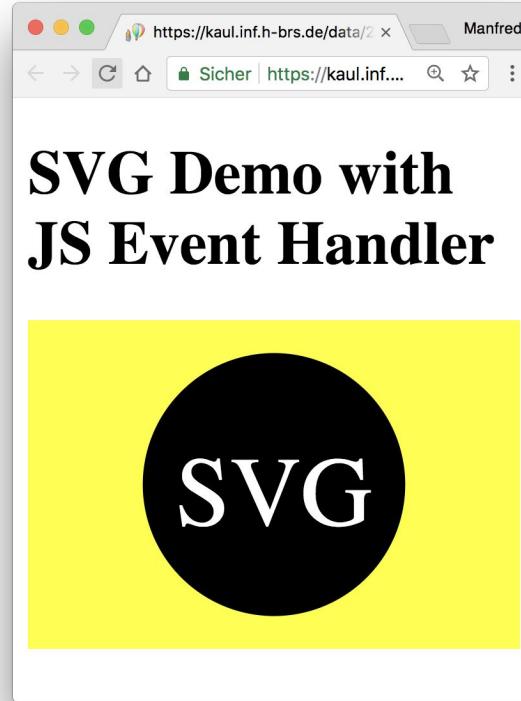
```
<svg viewBox="0 0 300 200">
  <rect width="100%" height="100%" fill="yellow" />
  <circle id="svg-circle" cx="150" cy="100" r="80" />
  <text x="150" y="125" font-size="60" text-anchor="middle"
    fill="white">SVG</text>
</svg>
```

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

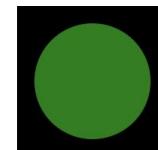
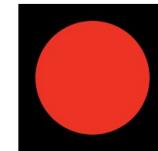


<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/script>



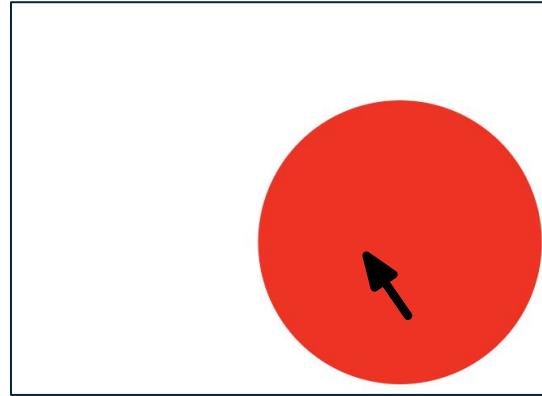
# SVG-DOM-Manipulation via JS

```
<!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

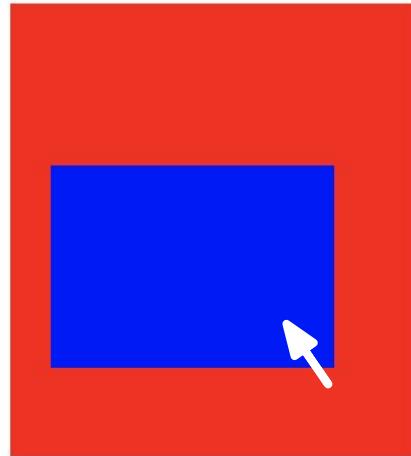
```
<!doctype html>
<svg>
  <circle id="circle" cx="100" cy="100" r="50" fill="red"></circle>
</svg>
<script>
  const circle = document.getElementById('circle');
  circle.addEventListener("mousemove", event => {
    circle.setAttribute('cx', event.x);
    circle.setAttribute('cy', event.y);
  });
</script>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



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



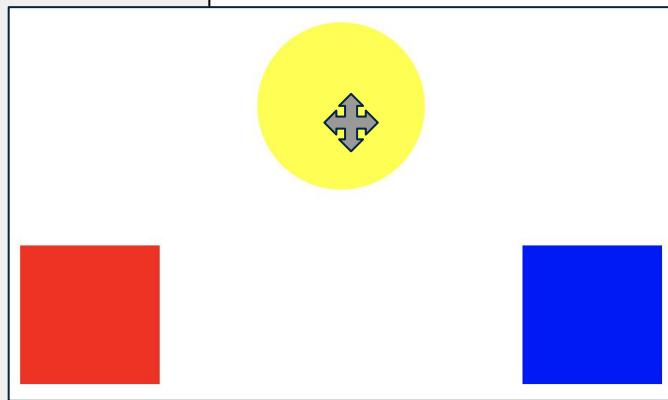
# Drag & Drop in SVG -1-

```
<!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
  };
}
```





## 3 Arten:

SMIL



1. CSS `<style>`
2. SVG `<animate>,  
<animateMotion>,  
<animateTransform>,  
<set>`
3. JavaScript `<script>`
  1. `requestAnimationFrame`
  2. `setInterval`
  3. `setTimeout`
  4. `Element.animate()`





# CSS 2D-Transformationen

w3schools.com

THE WORLD'S LARGEST WEB DEV



HTML

CSS

JAVASCRIPT

SQL

PHP

MORE ▾

REFERENCES ▾

EXAMPLE

CSS Image Sprites

CSS Attr Selectors

CSS Forms

CSS Counters

CSS Website Layout

CSS Units

CSS Specificity

CSS Advanced

CSS Rounded Corners

CSS Border Images

CSS Backgrounds

CSS Colors

CSS Gradients

CSS Shadows

CSS Text Effects

CSS Web Fonts

CSS 2D Transforms

CSS 3D Transforms

CSS Transitions

CSS Animations

## CSS 2D Transforms

◀ Previous

Next ▶

## CSS Transforms

CSS transforms allow you to translate, rotate, scale, and skew elements.

A transformation is an effect that lets an element change shape, size and position.

CSS supports 2D and 3D transformations.

Mouse over the elements below to see the difference between a 2D and a 3D transformation:

2D  
rotate

3D  
rotate





- CSS Forms
- CSS Counters
- CSS Website Layout
- CSS Units
- CSS Specificity

### CSS Advanced

- CSS Rounded Corners
- CSS Border Images
- CSS Backgrounds
- CSS Colors
- CSS Gradients
- CSS Shadows
- CSS Text Effects
- CSS Web Fonts
- CSS 2D Transforms
- CSS 3D Transforms

[CSS Transitions](#)

[CSS Animations](#)

## CSS Transitions

[◀ Previous](#)

[Next ▶](#)

## CSS Transitions

CSS transitions allows you to change property values smoothly (from one value to another), over a given duration.

**Example:** Mouse over the element below to see a CSS transition effect:

CSS





# CSS-Animationen

w3schools.com

THE WORLD'S LARG

HOME HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO MORE ▾ REFERENCES

CSS Advanced

CSS Rounded Corners

CSS Border Images

CSS Backgrounds

CSS Colors

CSS Gradients

CSS Shadows

CSS Text Effects

CSS Web Fonts

CSS 2D Transforms

CSS 3D Transforms

CSS Transitions

CSS Animations

CSS Tooltips

CSS Style Images

CSS Filters

## CSS Animations

◀ Previous

### CSS Animations

CSS animations allows animation of most HTML elements without using JavaScript or Flash!

CSS

### Browser Support for Animations

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](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>



[next](#) [contents](#)

## Synchronized Multimedia Integration Language (SMIL 3.0)

W3C Recommendation 01 December 2008

```
<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>





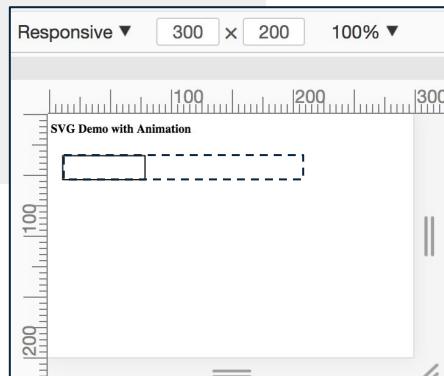
- **<animate>** - change values of numeric properties over time
- **<set>** - assign values to non numeric properties
- animateColor - modify color values over time (*deprecated*)
- **<animateTransform>** - modify transform values over time
- **<animateMotion>** - move an element along a path





# 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>
```



W3C Recommendation

[next](#) [contents](#)

Synchronized Multimedia Integration Language (SMIL 3.0)

W3C Recommendation 01 December 2008

This version:

<http://www.w3.org/TR/2008/REC-SMIL3-20081201/>

Latest SMIL 3 version:

<http://www.w3.org/TR/SMIL3/>

Latest SMIL Recommendation:

<http://www.w3.org/TR/SMIL/>

Previous version:

<http://www.w3.org/TR/2008/PR-SMIL3-20081006/>

Editors:

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



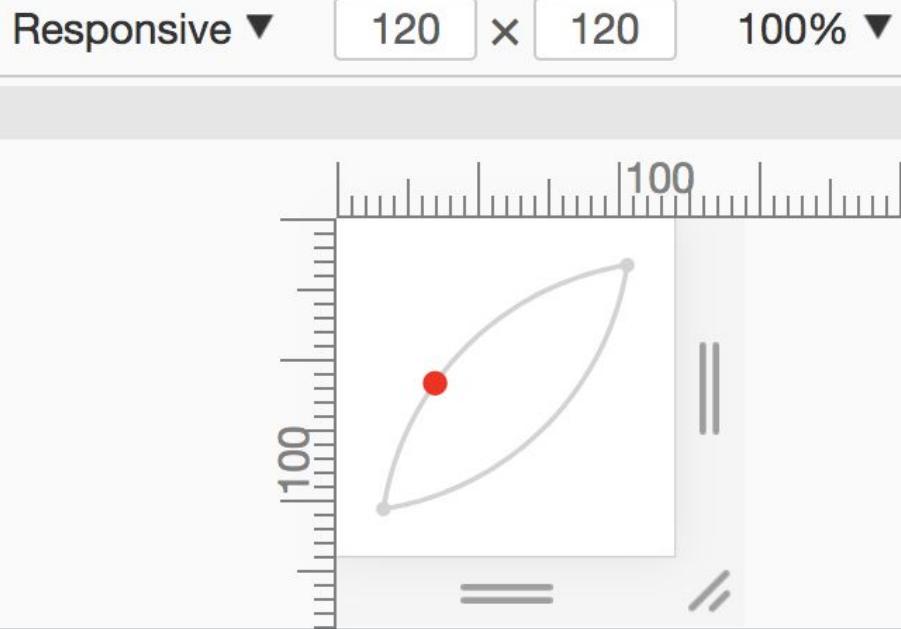
# Animation in SVG mit <animateMotion>



```
<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>
```



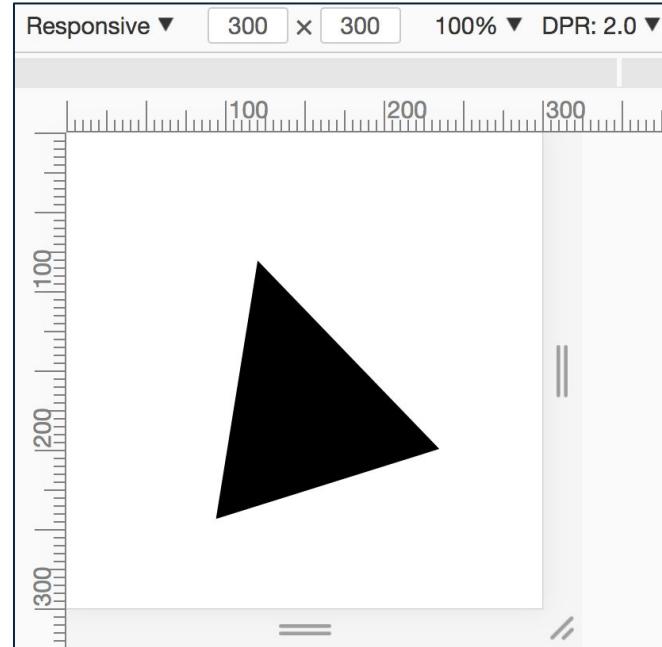
<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/animateMotion>



# 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>

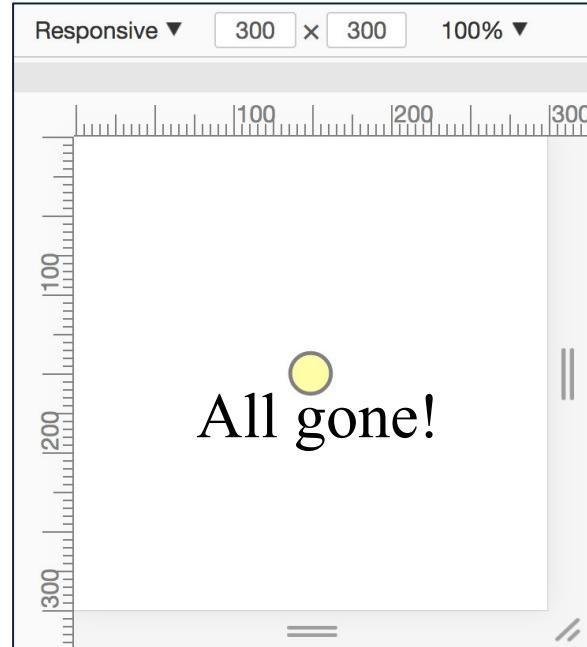


# Animation in SVG mit <set>



```
<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>
```



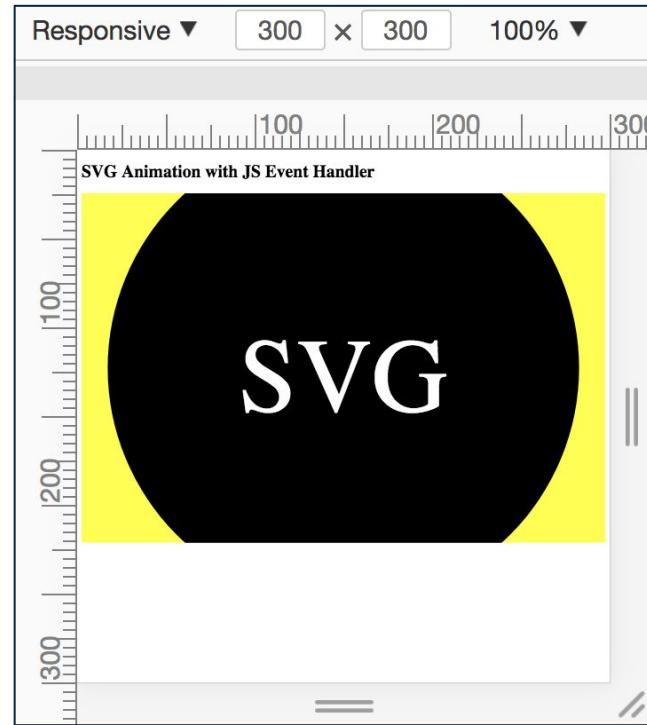
<https://developer.mozilla.org/en-US/docs/Web/SVG/Element/set>





# 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](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 );
}

```

## SVG Animation with requestAnimationFrame

Stop!



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



# Bessere Animation mit *requestAnimationFrame()*

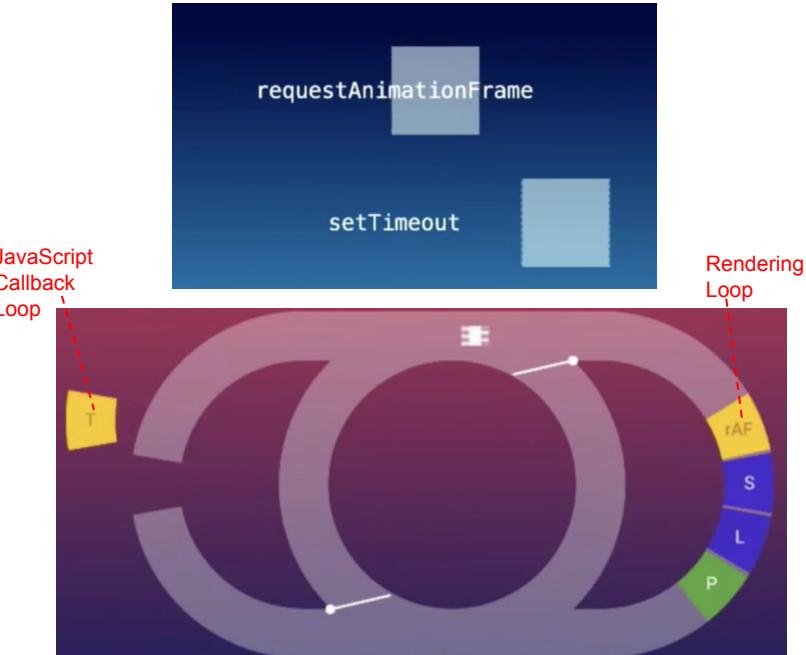
ist gegenüber *setInterval()* zu bevorzugen, um unnötige Aufrufe zu vermeiden

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

```
<script>
function myMove() {
  var elem = document.getElementById("animate");
  var pos = 0;
  frame();
}

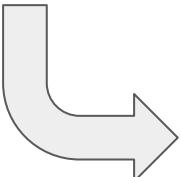
function frame() {
  if (pos > 249) {
    pos = 0;
  } else {
    pos += 5;
    elem.style.left = pos + 'px';
  }
  requestAnimationFrame( frame );
}
</script>
```

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



# Web Animation API

```
1 #alice {  
2   animation: aliceTumbling infinite 3s linear;  
3 }  
  
4 @keyframes aliceTumbling {  
5   0% {  
6     color: #000;  
7     transform: rotate(0) translate3D(-50%, -50%, 0);  
8   }  
9   30% {  
10    color: #431236;  
11  }  
12  100% {  
13    color: #000;  
14    transform: rotate(360deg) translate3D(-50%, -50%, 0);  
15  }  
16 }  
17 }
```



- 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

```
1 document.getElementById("alice").animate(  
2   [  
3     { transform: 'rotate(0) translate3D(-50%, -50%, 0)', color: '#000' },  
4     { color: '#431236', offset: 0.3},  
5     { transform: 'rotate(360deg) translate3D(-50%, -50%, 0)', color: '#000' }  
6   ], 3000);
```

# 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>
```

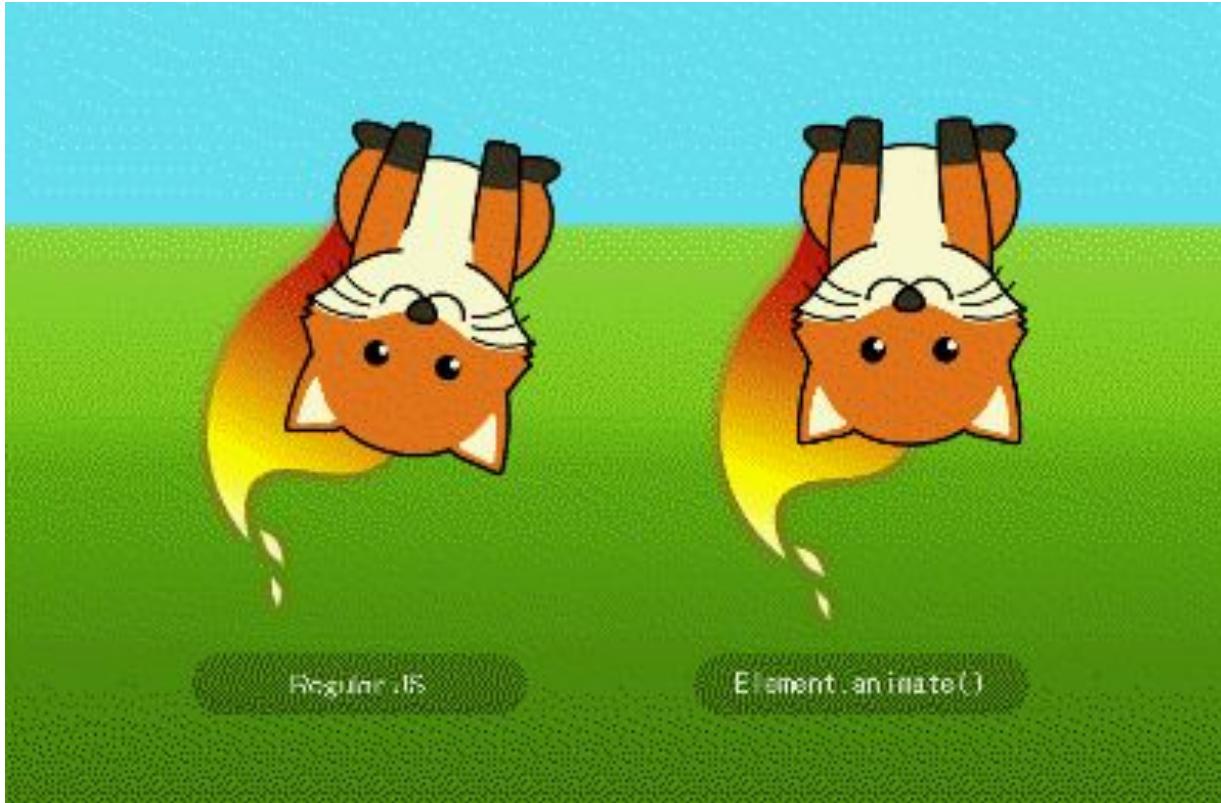


Element.animate()



# 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](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

Karte\_Deutschland.svg nrw.html svg-logo.html ccm.difference\_chart.js

Nordrhein

Match Case Regex Words One match

```

L 135.562,436.126 L 135.194,436.508 L 135.14,437.16 L 135.072,438.081 L 135.277,439
.032 L 135.562,439.497 L 136.176,440.26 L 136.446,440.558 L 137.912,441.983 L
139.364,443.53 L 139.765,443.964 L 139.898,445.527 L 139.597,447.038 L 142.245,446
.117 L 143.517,446.458 L 144.205,447.082 L 145.208,448.327 z "
style="fill:#e7e7e7;stroke:#000000;stroke-width:0.5;stroke-linejoin:round" /><path
id="Nordrhein-Westfalen"
d="M 231.536,350.825 L 230.242,353.004 L 226.397,357.612 L 223.158,362.93 L 218.694,366
.419 L 214.381,369.898 L 213.726,370.944 L 213.128,371.949 L 212.536,372.045 L
209.775,371.865 L 209.157,371.664 L 208.883,371.484 L 208.714,371.241 L 208.148,369
.613 L 208.862,367.838 L 207.683,365.882 L 204.759,364.646 L 203.59,364.699 L
201.37,365.249 L 199.778,365.787 L 196.924,367.005 L 196.241,367.692 L 196.241,368
.136 L 196.421,368.792 L 198.938,372.936 L 199.483,373.063 L 199.79,373.231 L
199.833,373.456 L 200.045,374.81 L 199.22,377.262 L 199.073,377.316 L 195.633,377.093
L 191.466,377.189 L 182.316 L 177.583,383 L 176.684,391.378 L 176
.838 L 183.383,388.78 L 390.163 L 186.716,396 L 183.693,405.874 L 180
.55 L 176.488,406.275 L 172.137,406.761 L 171
.686 L 170.499,408.99 L 172.128,415.335 L 171
.057 L 168.37,419.216 L 167.89,419.776 L 167
.162.455,429.251 L 1

```

Manfred

Sicher | https://kaul.inf.h-brs.de/data/

NRW-Karte

SVG Demo Extract NRW Path

path | 587.4 × 406.01

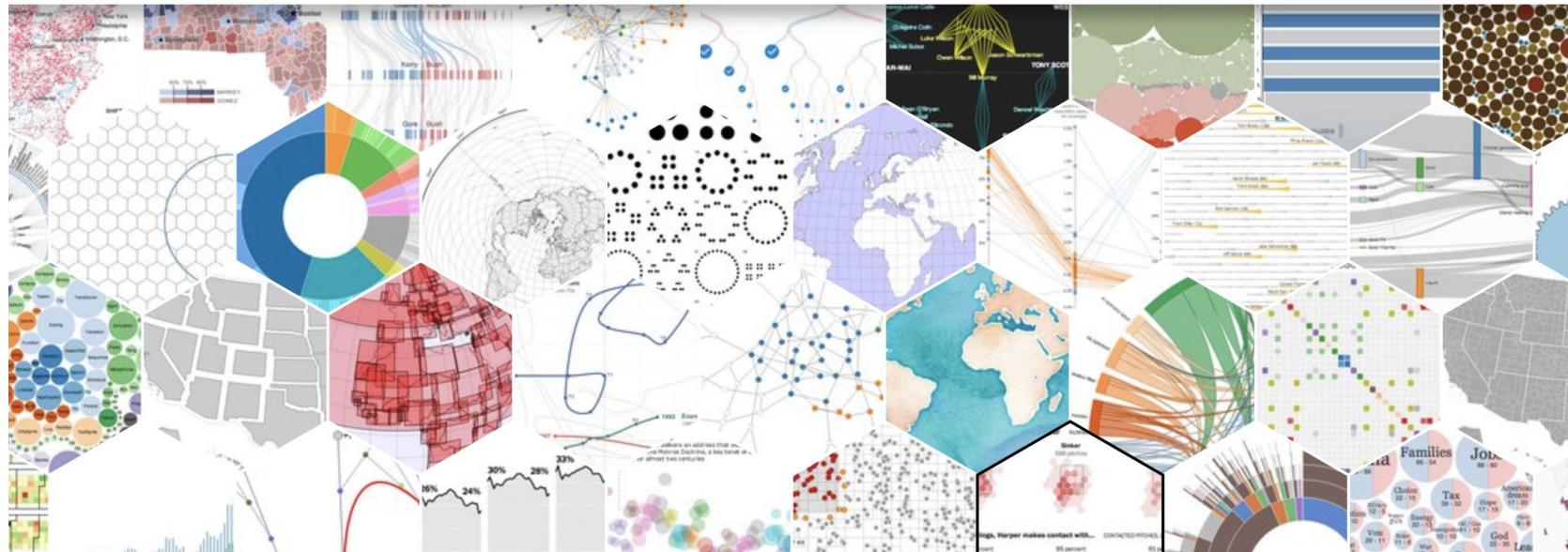
path d="M 231.536,350.825 L 230.242,353.004 L 226.397,357.612 L 223.158,362.93 L 218.694,366.419 L 214.381,369.898 L 213.726,370.944 L 213.128,371.949 L 212.536,372.045 L 209.775,371.865 L 209.157,371.664 L 208.883,371.484 L 208.714,371.241 L 208.148,369.613 L 208.862,367.838 L 207.683,365.882 L 204.759,364.646 L 203.59,364.699 L 201.37,365.249 L 199.778,365.787 L 196.924,367.005 L 196.241,367.692 L 196.241,368.136 L 196.421,368.792 L 198.938,372.936 L 199.483,373.063 L 199.79,373.231 L 199.833,373.456 L 200.045,374.81 L 199.22,377.262 L 199.073,377.316 L 195.633,377.093 L 191.466,377.189 L 182.316 L 177.583,383 L 176.684,391.378 L 176.838 L 183.383,388.78 L 390.163 L 186.716,396 L 183.693,405.874 L 180.55 L 176.488,406.275 L 172.137,406.761 L 171.686 L 170.499,408.99 L 172.128,415.335 L 171.057 L 168.37,419.216 L 167.89,419.776 L 167.162.455,429.251 L 1

[https://commons.wikimedia.org/wiki/Category:SVG\\_maps\\_of\\_Germany](https://commons.wikimedia.org/wiki/Category:SVG_maps_of_Germany)



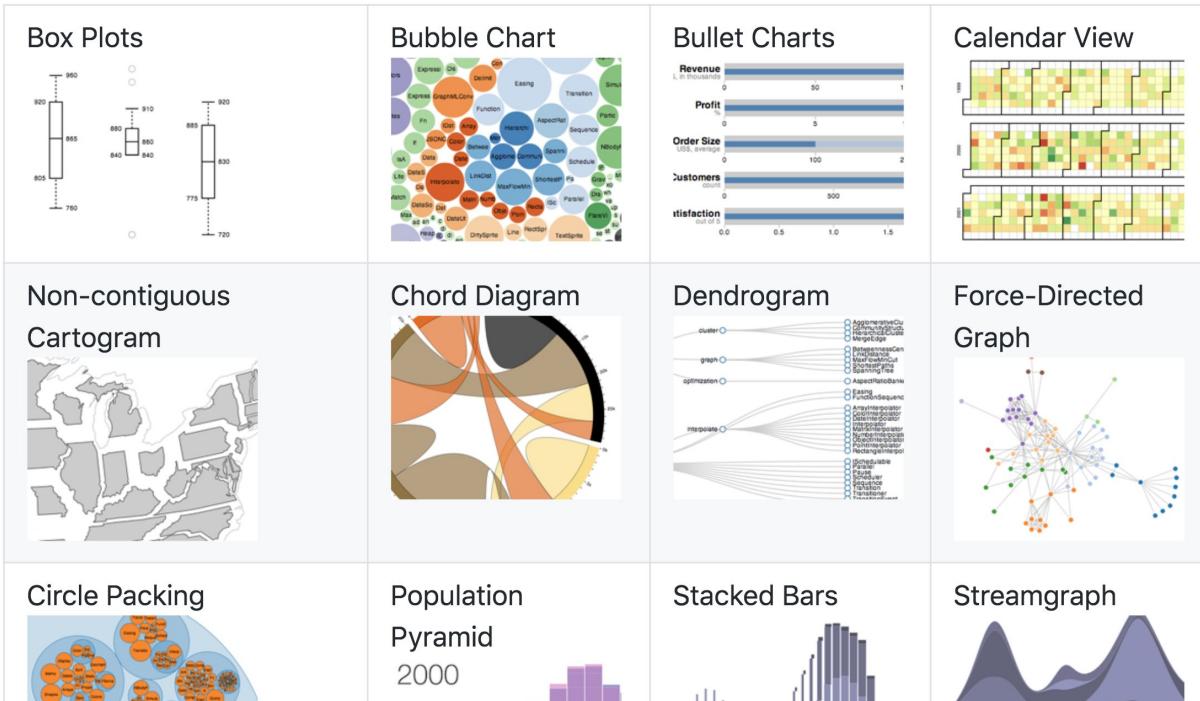


# d3 Data-Driven Documents



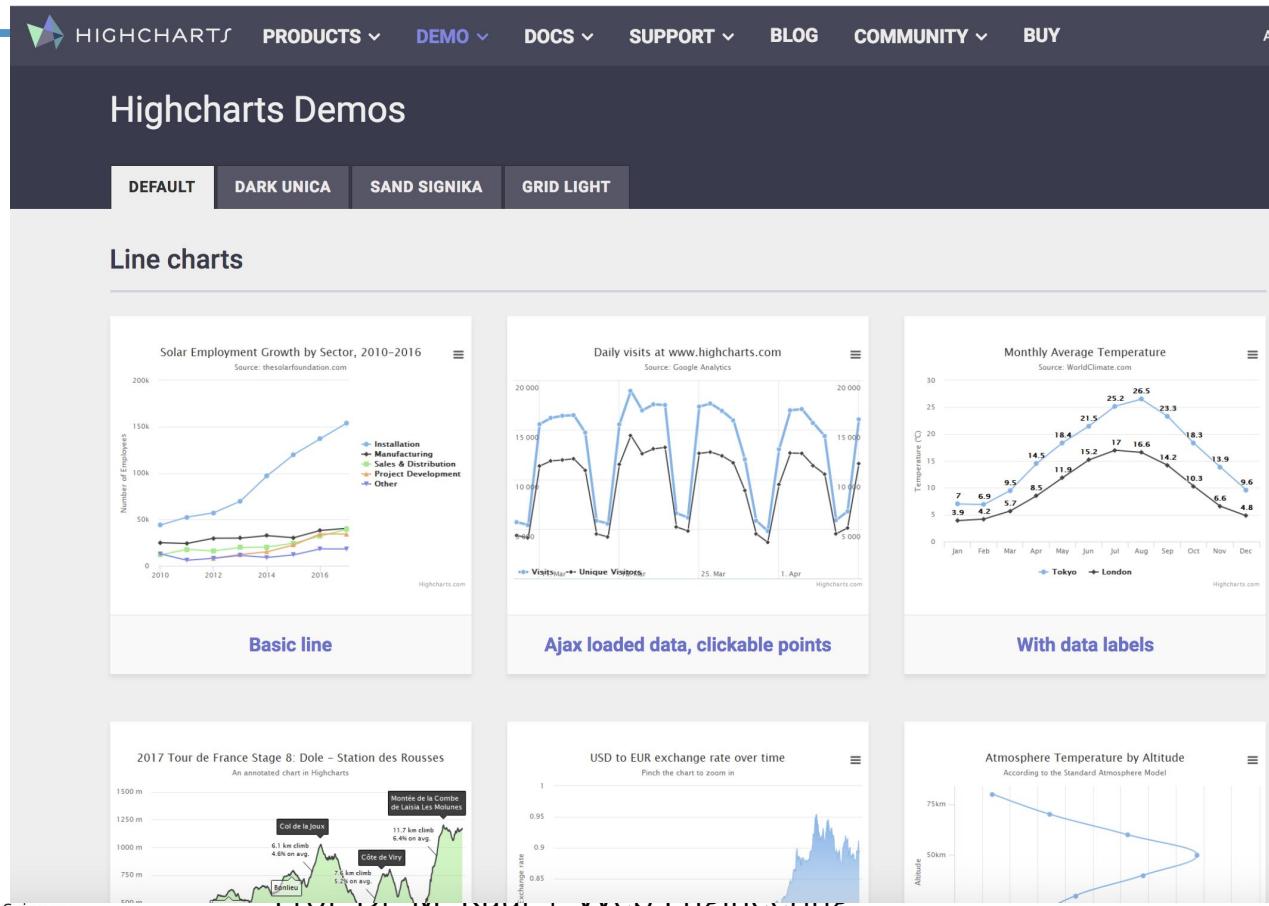


## Visual Index





# Highcharts.com



# Snap SVG

Snap.svg

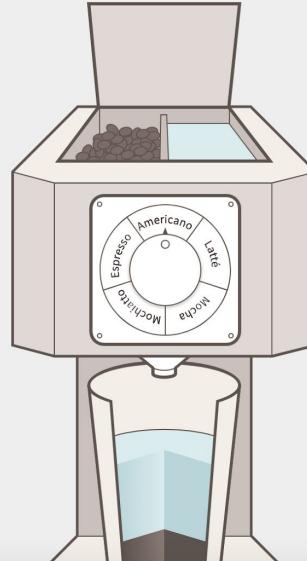
Home Why Snap Getting Started Docs Support [Demos](#) [Download](#)

Snap Demos

-  Coffeemaker
-  Animated game
-  Globe
-  Snap Mascot
-  Display Ad

Websites using Snap

-  PBS KIDS [↗](#)



A detailed illustration of a vintage-style coffee machine. The machine has a light brown body with a circular control panel on top. The control panel features a dial with labels: Espresso, Mochaccino, Cappuccino, Mocha, Latte, and Americano. Below the control panel is a large cylindrical water reservoir. A spout at the bottom dispenses coffee into a clear glass cup. To the right of the machine is a small inset showing a pie chart labeled "Americano". The pie chart indicates a ratio of Espresso (33%) and Hot Water (67%).

<http://snapsvg.io/>



# Animation Libraries

## GreenSock

This pen.

ROTATE THE GEAR  
OR HIT PLAY

PLAY

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



<https://greensock.com/draggable>



GreenSock Engaging the Internet

Products Examples Docs Forums Club GreenSock

Our Products

Browse through our amazing tools.

Language: HTML5

Search Products... SEARCH

Sort By: Popular New

**GSAP**

GSAP is a suite of tools for scripted, high-performance HTML5 animations that work in all major browsers. No other library delivers such advanced sequencing, API efficiency, and tight control. Stop wrestling with

**TweenMax**

TweenMax extends TweenLite, adding many useful (but non-essential) features like repeat(), repeatDelay(), yoyo(), staggered tweens, and more. It also includes many extra tools and plugins by default, making it

**TweenLite**

TweenLite is the fast, lightweight, and flexible core of GSAP (GreenSock Animation Platform), a JavaScript library for HTML5 animation.

<https://greensock.com/products/>

# Anwendung: Komplette Programmierumgebung in SVG

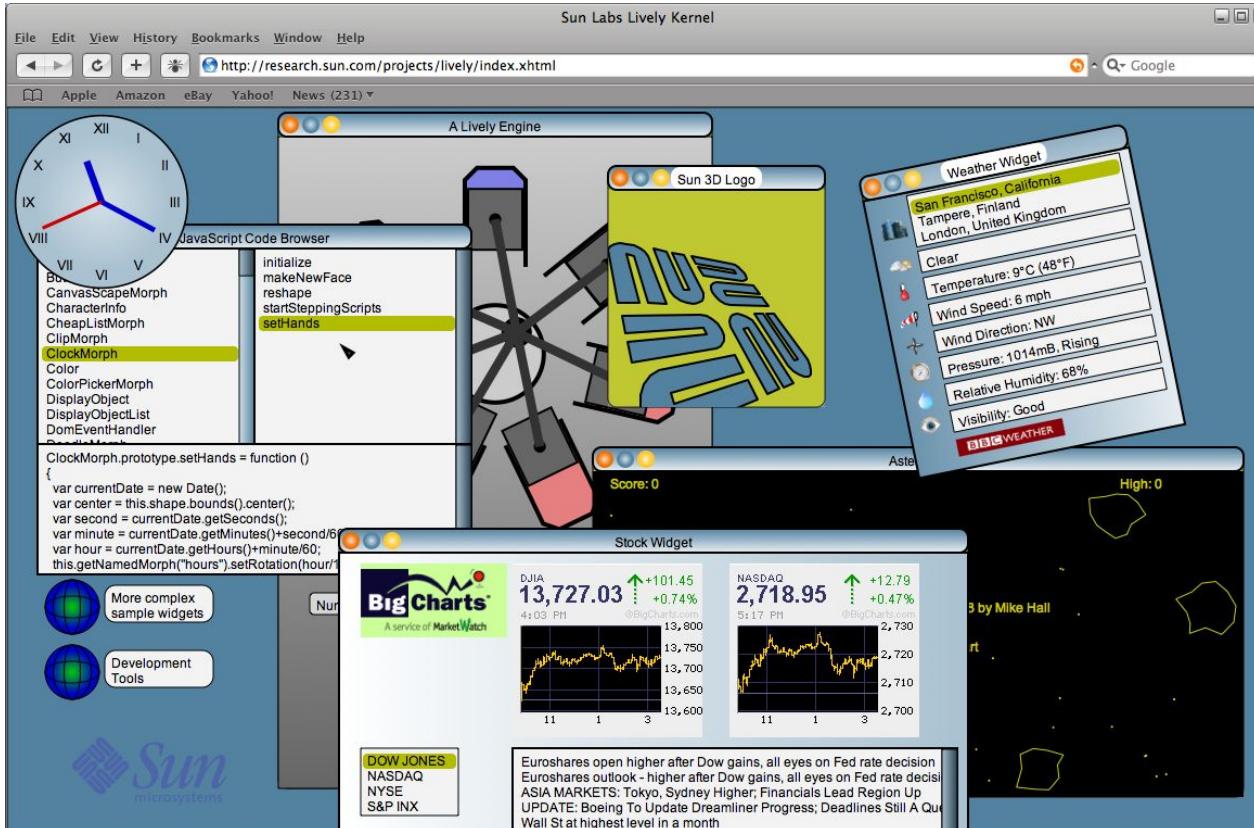
Smalltalk

Self

Squeak

Morphic

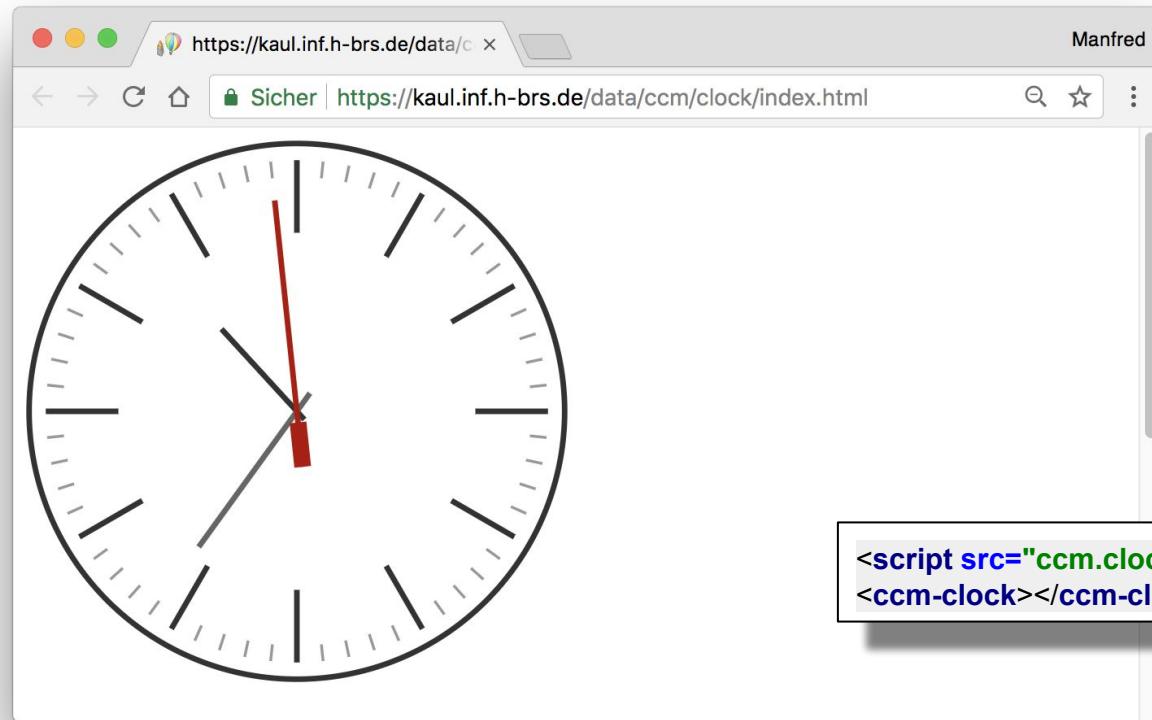
Lively



<http://lively-kernel.org/>



# Analog-Uhr mit SVG



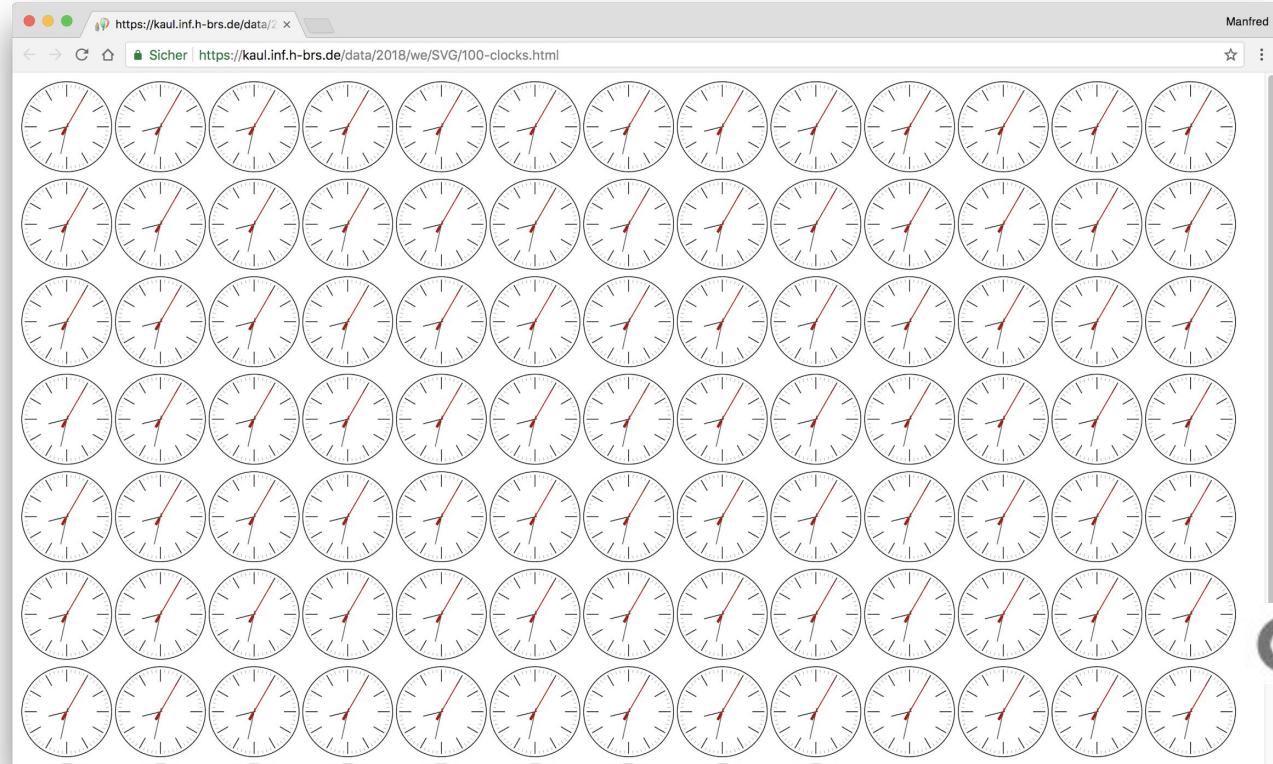
```
<script src="ccm.clock.js"></script>
<ccm-clock></ccm-clock>
```



# Webseite mit 100 Uhren

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

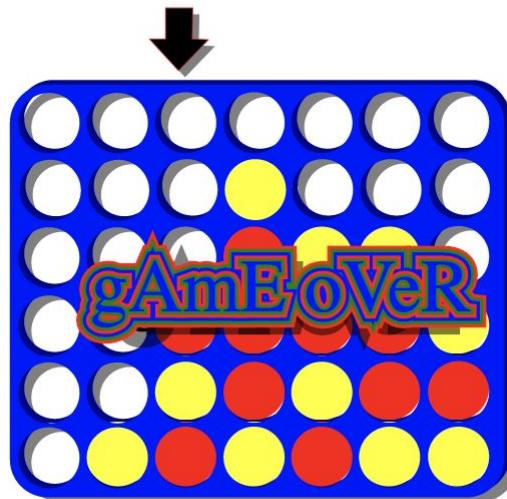
- 100 Uhren laufen noch synchron.
- Bei 1000 Uhren kommt es zu Sprüngen des Sekundenzeigers von ca. 5 sec.



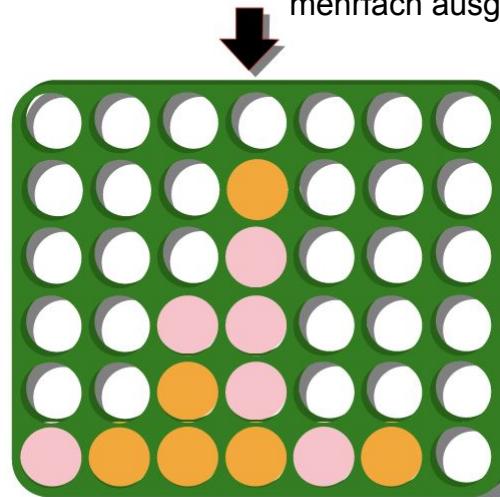


# Vier gewinnt! Vier gewinnt! Vier gewinnt!

*Simultan-Spiel*

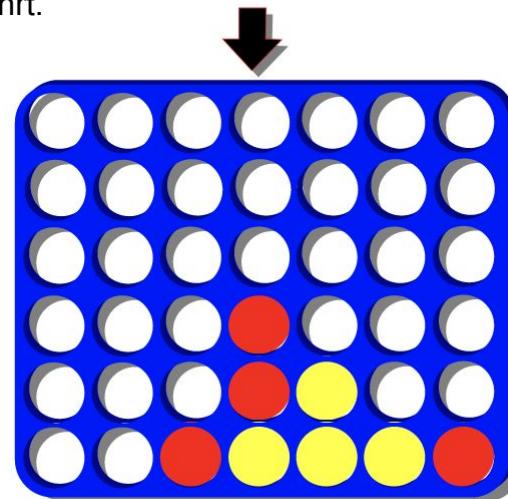


computer wins ! score:1/0



Prof. Dr. M. Kaul | Web Engineering

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



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

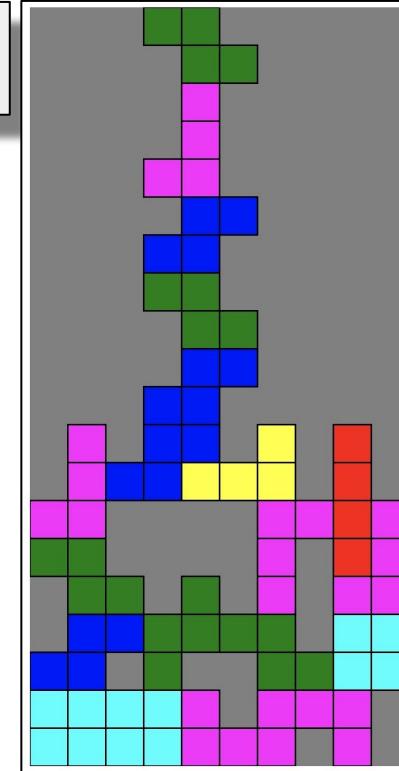




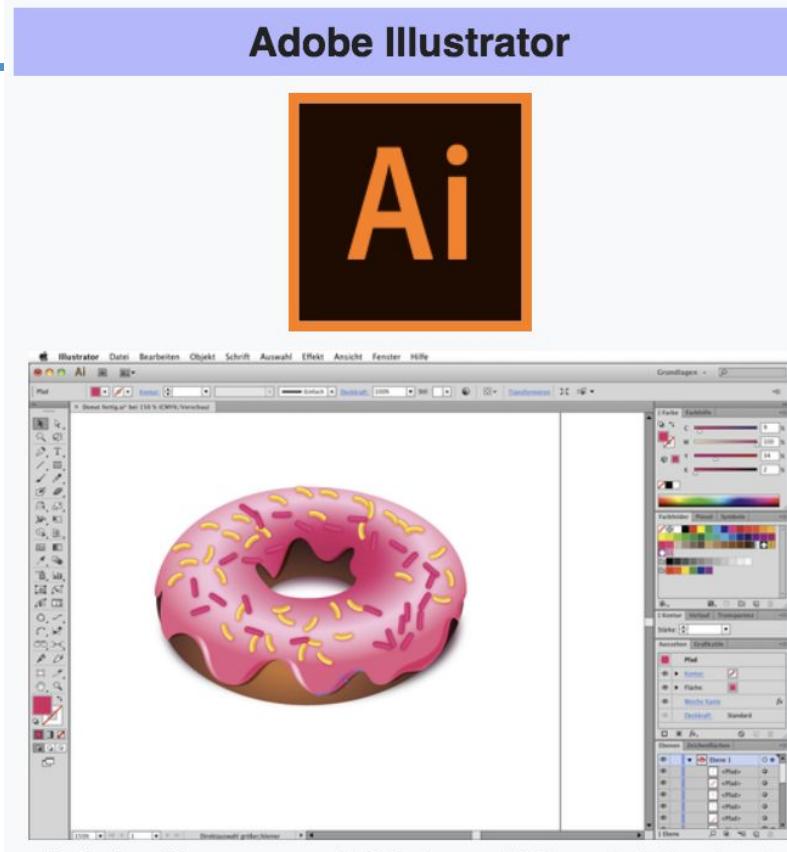
```
<script src="ccm.tetris.js"></script>
<ccm-tetris></ccm-tetris>
```

Historisches:

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



# Adobe Illustrator



Adobe Illustrator CS6 (macOS, mit Plugins)

Prof. Dr. M. Kaul | Web Engineering





ABOUT DOWNLOAD NEWS COMMUNITY LEARN CONTRIBUTE DEVELOP SUPPORT US

Inkscape is a professional vector graphics editor for Windows, Mac OS X and Linux. It's free and open source.

[Download](#)

Current stable version: 0.92.3

Inkdrop Diffusion in Water by artelnjeru01

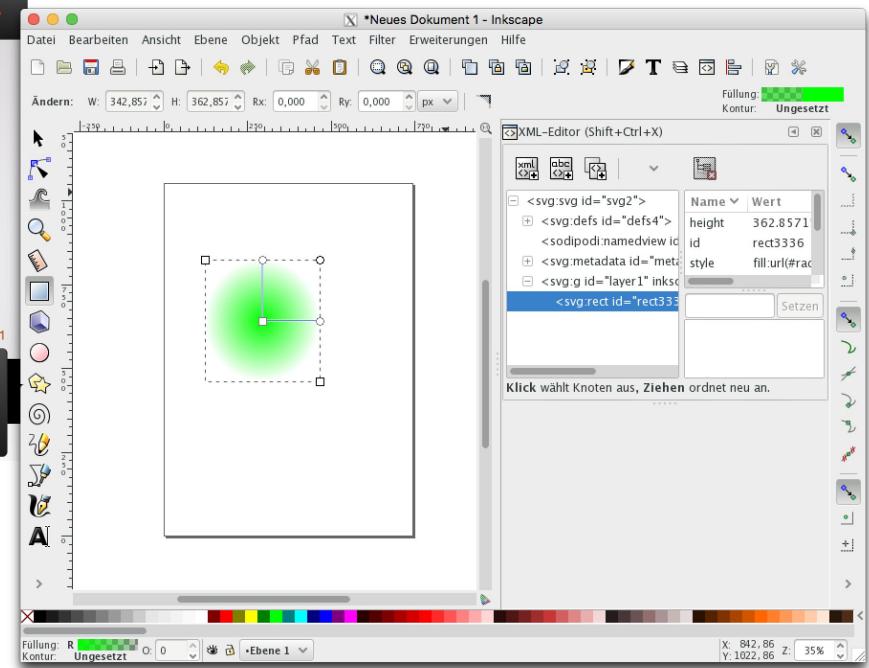
**INKSCAPE 0.92**  
Draw *Freely*

**Overview**  
What is Inkscape and how can I get it?

**Features**  
Find out what Inkscape is capable of

**Gallery**  
Showcase of creations from the community

**Learning Resources**  
Resources to help you get the most out of Inkscape



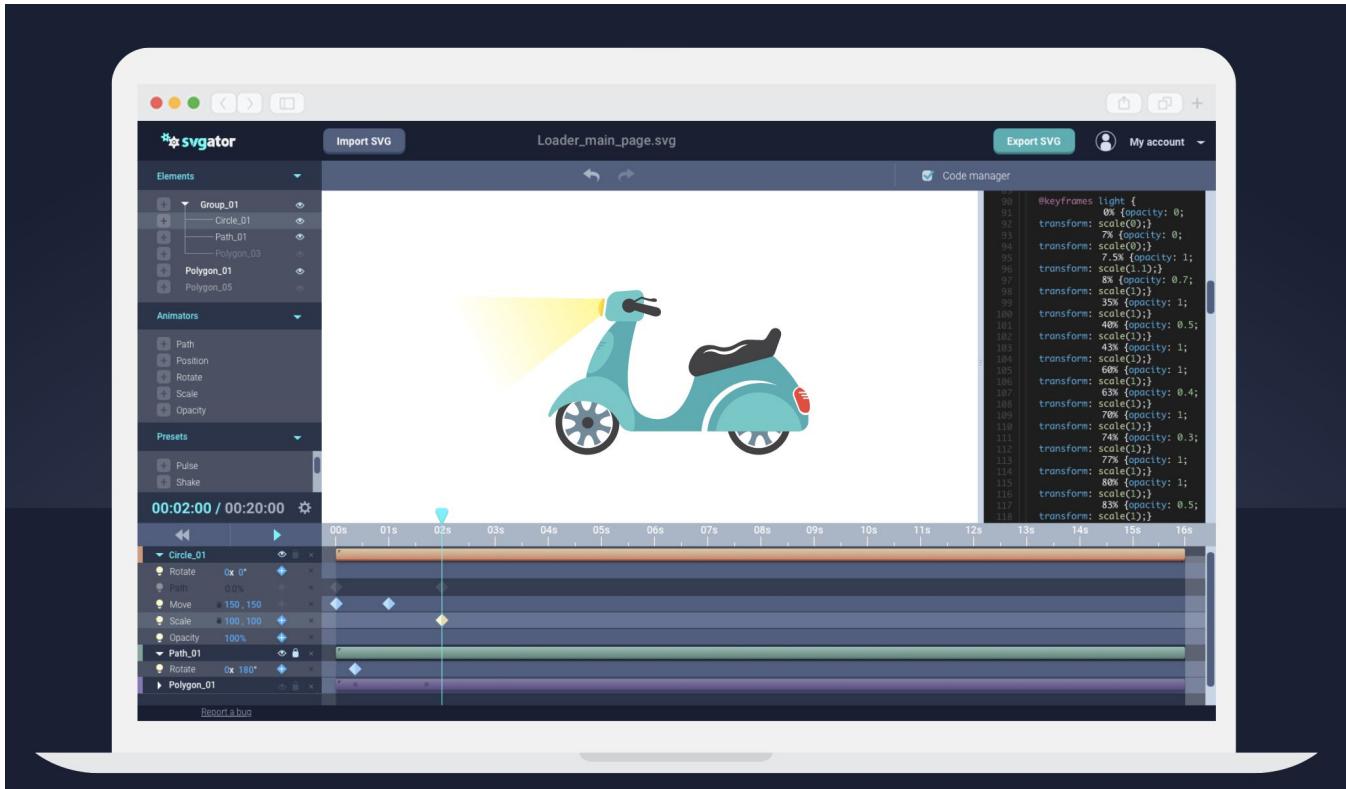
# Sketch



# Sketch

The screenshot shows the Sketch application interface. On the left is the library panel with categories like Social/New, Social/Story, Social/Story (Comments), Social/Profile, and Photography. The main canvas displays a wireframe of a mobile application screen. The right side features the Inspector panel with various settings for selected elements, such as Position, Size, Transform, Resizing, Prototyping, Target, Animation, Overrides, Header, Date, @Handle, Name, Avatar, Content, Text, Icon, Color, and Make Exportable.

# Generierung von Animationen





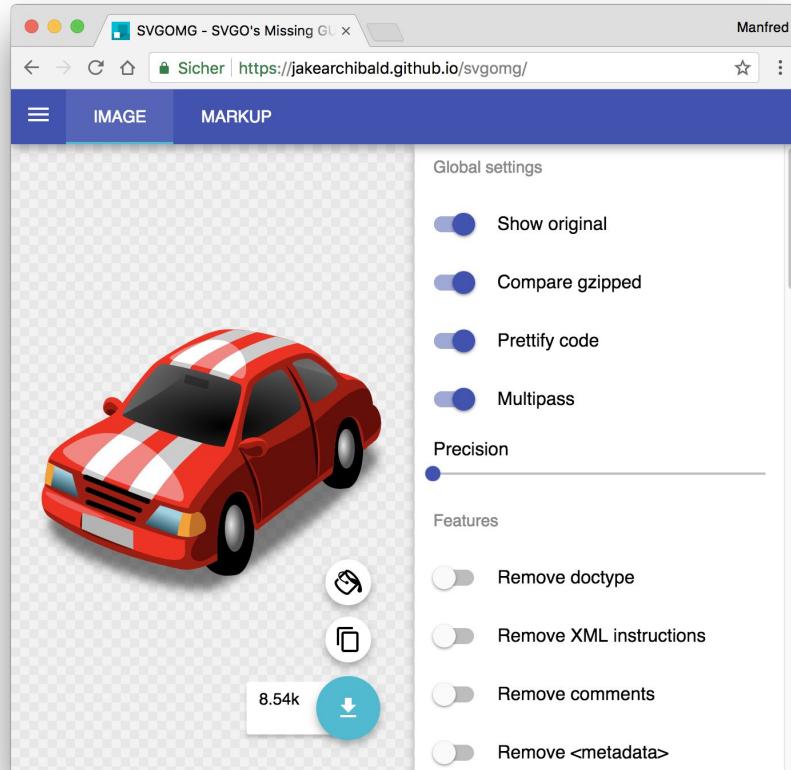
# Boxy SVG

The screenshot shows the Boxy SVG application window. The title bar reads "Boxy SVG" and the file name is "Zwei\_Ringe.svg". The main canvas displays two overlapping rings: a red one on the left and a green one on the right, each with a small green circular handle. The green ring has a red outline. On the left side, there is a vertical toolbar with various icons for selection, transformation, and shape creation. At the bottom, an "Inspector" panel is open, showing the XML code for the SVG elements and their corresponding CSS styles.

```
<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">
      <stop offset="0%>
        <stop offset="100%" stop-color="#f08080"/>
    </linearGradient>
    <pattern id="pattern-0" x="0" y="0" width="25" height="25" patternUnits="userSpaceOnUse" viewBox="0 0 100 100">
      <image alt="A small green circle with a red outline, representing a handle for the green ring." />
    </pattern>
    <linearGradient id="gradient-4" gradientUnits="userSpaceOnUse" x1="205" y1="82.544" x2="205" y2="205.446">
      <stop offset="0%>
        <stop offset="100%" stop-color="#ffcc00"/>
    </linearGradient>
  </defs>
  <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 58.22 0 0 1 116.44 0 a 58.22 58.22 0 0 1 -116.44 0 Z" bx:shape="ring" 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" transform="matrix(-0.714834, 0.699294, -0.699294, -0.714834, 537.315875, 159.962431)" style="fill-opacity: 0.6; fill: url("#gradient-4");">
```

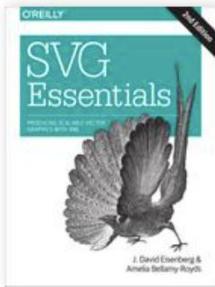
```
element.style {
  fill-opacity: 0.65;
  fill: url("#gradient-0");
}
```

# SVG Minimizer



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





## 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/>



# Nachschatlagewerke

- 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://de.wikipedia.org/wiki/Scalable_Vector_Graphics)
  - [https://en.wikipedia.org/wiki/Scalable\\_Vector\\_Graphics](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics)



## W3Schools

[https://www.w3schools.com/graphics/svg\\_intro.asp](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>

Questions Developer Jobs Tags Users [svg]

The results are in! See the 2018 Developer Survey results. »

Tagged Questions info newest 1 featured frequent votes active unanswered

27,697 questions tagged Ask Question

Scalable Vector Graphics (SVG) is an XML-based two-dimensional vector graphics format that can also be used in HTML. Do not add this tag just because your project uses SVG. Instead, add the tag if your question is either about SVG, or closely related to it, like how to achieve something with SVG.

learn more... improve tag info top users synonyms (2)

508 votes 12 answers 287K views

**PNG vs. GIF vs. JPEG vs. SVG - When best to use?**

When should certain image filetypes be used when building websites or interfaces, etc? What are their points of strength and weakness? I know that PNG & GIF are lossless, while JPEG is lossy. ...

image svg png jpeg gif

asked Feb 25 '10 at 18:21 Faruz 5,379 ● 9 ● 33 ● 58

431 votes 9 answers

**Do I use <img>, <object>, or <embed> for SVG files?**

Should I use <img>, <object>, or <embed> for loading SVG files into a page in a way similar to loading a jpg, gif or png? What is the code for each to ensure it works as well as ...

html graphics svg

asked Dec 18 '10 at 3:50 artlung

UPCOMING EVENTS

2018 Community Moderator Election ends in 3 days

FEATURED ON META

2018 Moderator Election Q&A - Questionnaire

Coming Soon: Stack Overflow For Teams!

HOT META POSTS

32 Why are vote counts for moderator elections public? Shouldn't they be private?

Favorite Tags [edit](#)





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

