



SVG

- Skalierbare Vektorgrafiken. Ein Überblick.

Dr. Thomas Meinike

Hochschule Merseburg (FH)
Fachbereich Informatik und Kommunikationssysteme

thomas.meinike@hs-merseburg.de
<http://www.et.fh-merseburg.de/person/meinike/>

Merseburg, 2005-10-25



Inhaltliche Schwerpunkte

⇒ Grundlagen:

- Allgemeines über das Thema SVG
- Aufbau von SVG-Dokumenten
- Grundformen und weitere Zeichentechniken
- Gradienten, Filter, Transformationen, Animationen

⇒ Fortgeschrittene Techniken:

- Aktionsprogrammierung mit JavaScript
- SVG-Generierung mittels XSLT und PHP

⇒ Nicht oder nur ansatzweise behandelt werden:

- Mobile SVG-Standards (→ tekcom-Tagung im Nov. 2005)
- SVG 1.2 (→ Material unter svglbc.datenverdrahten.de)



Was ist SVG?

- ⇒ SVG wurde ursprünglich von Adobe als Gegenentwurf zu anderen vektorbasierten Web-Technologien wie Flash oder VML entwickelt und später als W3C-Empfehlung verabschiedet.

Qualität von Vektorgrafiken im Web wird verbessert; wichtigstes Anwendungsgebiet eCommerce

Adobe Systems stellt SVG als neuen Web-Grafikstandard vor

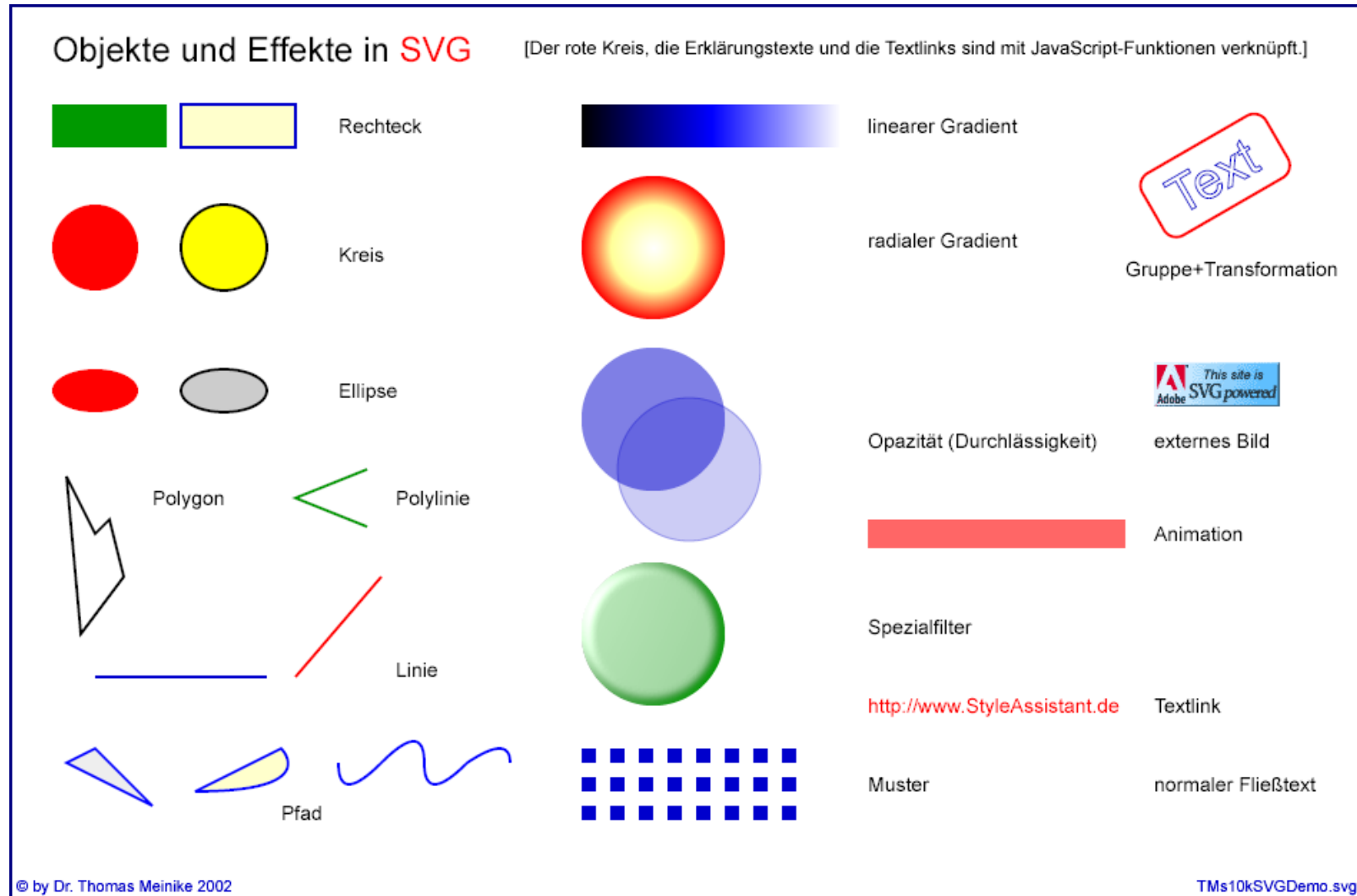
Unterschleißheim, 18. Februar 1999 – Adobe Systems hat dem World Wide Web Consortium (W3C) einen neuen offenen Standard für Web-Grafiken vorgeschlagen: Mit den hochauflösenden Scalable Vector Graphics (SVG) läßt sich die grafische Qualität von Web-Angeboten entscheidend verbessern. Ein wichtiger Anwendungsbereich sind eCommerce-Websites, denn mit SVG kann die von gedruckten Katalogen, Magazinen und Anzeigen gewohnte Qualität jetzt auch im Web realisiert werden. SVG verleiht Web-Grafiken darüber hinaus dynamische und interaktive Dimensionen. Die Grafiken sind deutlich schneller herunterzuladen, entlasten damit die Bandbreite und optimieren die Browser-Performance. SVG wird im Laufe des Jahres für Endkunden, Web-Publisher und Entwickler verfügbar sein. Mit der Spezifikation der Precision Graphics Markup Language (PGML) hatte Adobe dem W3C bereits einen Web-Standard vorgeschlagen und damit seine zentrale Rolle bei der technologischen Entwicklung im Web-Bereich unterstrichen.

„Wir streben im Web-Publishing eine ähnlich führende Position an, wie wir sie seit langem im Print-Bereich einnehmen“, sagt John Warnock, Chief Executive Officer (CEO) von Adobe Systems. „Die Entwicklung von SVG ist ein wichtiger Schritt in diese Richtung. SVG wird besonders den derzeit am schnellsten wachsenden Sektor des Internet, das eBusiness, voranbringen und so die Online-Erfahrung von morgen mitdefinieren.“



Was ist SVG?

⇒ XML-Vokabular zur Beschreibung von 2D-Vektorgrafiken



Was ist SVG?

- ⇒ SVG 1.0 ist eine 2001 verabschiedete W3C-Spezifikation zur Beschreibung von 2D-Vektorgrafiken in XML-Syntax.
- ⇒ Enthalten sind geometrische Primitive wie Rechteck, Kreis, Ellipse, Linie, Polylinie, Polygon sowie weitere Elemente zur Darstellung von Pfaden, Text, Hyperlinks usw.
- ⇒ SVG-Inhalte lassen sich mit CSS formatieren und durch den Einsatz von JavaScript-Routinen dynamisch verändern sowie mit Sprachen wie PHP, Perl usw. leicht auf dem Webserver erzeugen.
- ⇒ Techniken wie Animationen, Gradienten, Filter und Transformationen ermöglichen auch komplexe Darstellungen.

⇒ Vorteile:

offen, plattformunabhängig, durchsuchbar, Einsatz für Web und Print

⇒ Einsatzgebiete:

Kartografie, technische Illustration, Datenpräsentation, XML-Workflows



⇒ Zeitraster:

W3C-SVG-Roadmap

09/2001: SVG 1.0

01/2003: SVG 1.1 (Modularisierung in mobile Profile Tiny bzw. Basic)

07/2006: SVG 1.2 (Erweiterung des Sprachumfangs)

Document	FWD	<i>Next WD</i>	LC	CR	PR	REC
SVG 1.0	11 Feb 1999	–	03 Mar 2000	02 Aug 2000	19 July 2001	5 Sep 2001
SVG 1.1	30 Oct 2001	–	15 Feb 2002	30 Apr 2002	15 Nov 2002	14 Jan 2003
SVG Mobile Profiles	30 Oct 2001	–	15 Feb 2002	30 Apr 2002	15 Nov 2002	14 Jan 2003
SVG Mobile1.2	9 Dec 2003	–	13 April 2005	<i>[Aug 2005]</i>	<i>[Jan 2006]</i>	<i>[Mar 2006]</i>
SVG 1.2	11 Nov 2002	–	<i>[Aug 2005]</i>	<i>[Oct 2005]</i>	<i>[May 2006]</i>	<i>[July 2006]</i>
DOM Level 3 Events	01 Sep 2000	–	31 Mar 2003	<i>[Dec 2004]</i>	<i>[Mar 2005]</i>	<i>[May 2005]</i>
DOM Level 3 XPath	18 Jun 2001	–	28 Mar 2002	31 Mar 2003	<i>[Mar 2005]</i>	<i>[May 2005]</i>
sXBL	01 Sep 2004	05 Apr 2005	<i>[Aug 2005]</i>	<i>[Oct 2005]</i>	<i>[Mar 2006]</i>	<i>[May 2006]</i>
SVG Print	15 July 2003	<i>[Jun 2005]</i>	<i>[Sep 2005]</i>	<i>[Nov 2005]</i>	<i>[Apr 2006]</i>	<i>[Jun 2006]</i>
Authoring Tool Guidelines	<i>[Aug 2005]</i>	–	–	–	–	–
Accessibility Techniques	<i>[Sep 2005]</i>	–	–	–	–	–

Legend: **FWD** = First working draft; **LC** = last call for comments (i.e., last WD); **CR** = Candidate Recommendation; **PR** = Proposed Recommendation; **REC** = W3C Recommendation. *[Feb 2005]* = expected date.



W3C-SVG-Spezifikationen



→ <http://www.w3.org>

Scalable Vector Graphics (SVG) 1.1 Specification

W3C Recommendation 14 January 2003

This version:

<http://www.w3.org/TR/2003/REC-SVG11-20030114/>

Latest version:

<http://www.w3.org/TR/SVG11/>

Previous version:

<http://www.w3.org/TR/2002/PR-SVG11-20021115/>

Editors:

Jon Ferraiolo, Adobe Systems <jon.ferraiolo@adobe.com> (version 1.0)

藤沢 淳 (FUJISAWA Jun), Canon <fujisawa.jun@canon.co.jp> (modularization and DTD)

Dean Jackson, W3C/CSIRO <dean@w3.org> (version 1.1)

Authors:

See [author list](#)

Please refer to the [errata](#) for this document, which may include some normative corrections.

This document is also available in these non-normative packages: [zip archive of HTML](#) (without external dependencies) and [PDF](#).

See also the [translations](#) of this document.

Copyright © 2003 W3C® ([MIT](#), [ERCIM](#), [Keio](#)), All Rights Reserved. W3C [liability](#), [trademark](#), [document use](#) and [software licensing](#) rules apply.

Beteiligte Autoren / Firmen

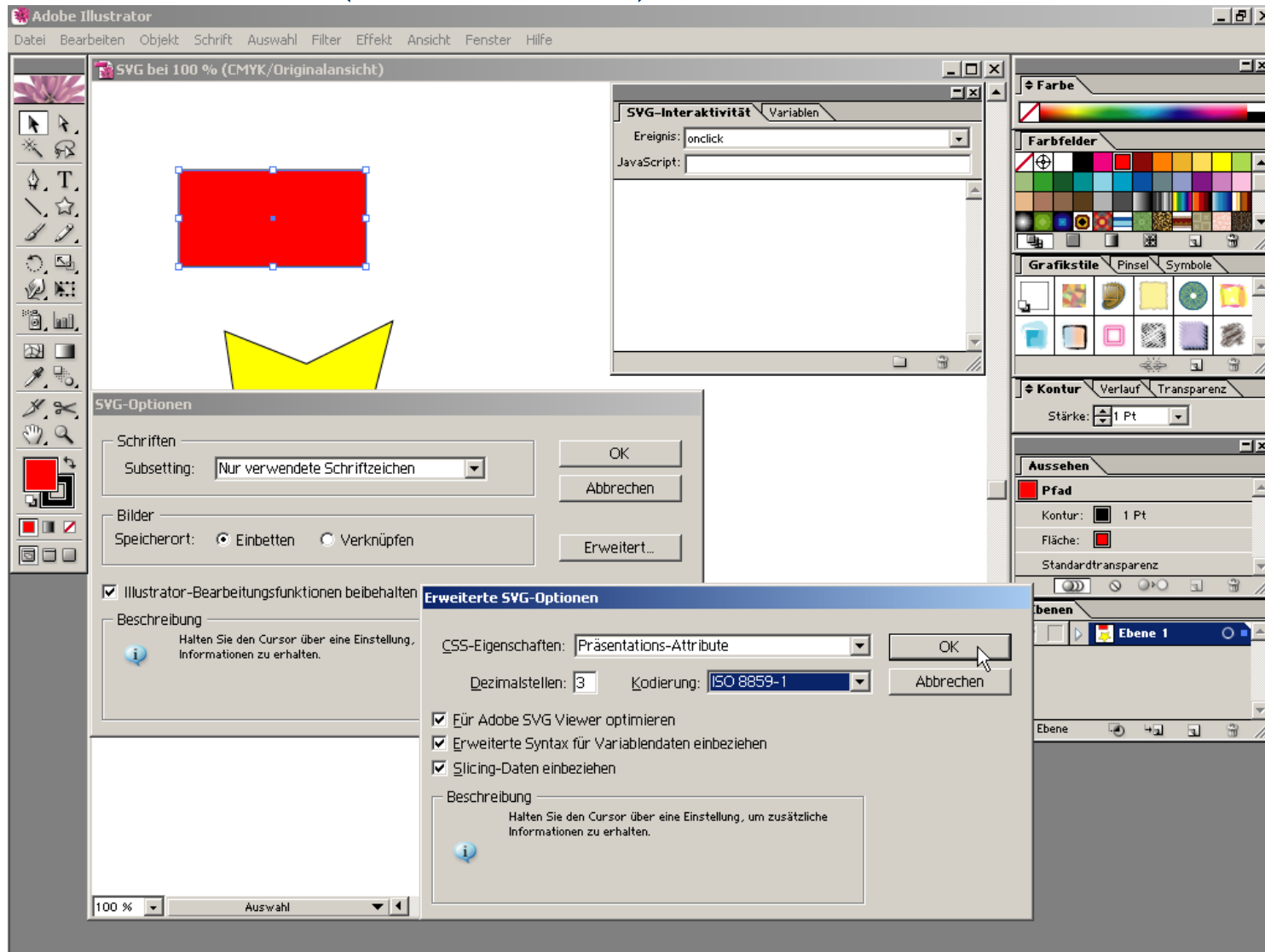
Authors:

- Ola Andersson, ZOOMON AB
- Phil Armstrong, Corel Corporation
- Henric Axelsson, Ericsson AB
- Robin Berjon, Expway
- Benoît Bézaire, Corel Corporation
- John Bowler, Microsoft Corporation
- Craig Brown, Canon Information Systems Research Australia
- Mike Bultrowicz, Savage Software
- Tolga Capin, Nokia
- Milt Capsimalis, Autodesk Inc.
- Mathias Larsson Carlander, Ericsson AB
- Jakob Cederquist, ZOOMON AB
- Charilaos Christopoulos, Ericsson AB
- Richard Cohn, Adobe Systems Inc.
- Lee Cole, Quark
- Don Cone, America Online Inc.
- Alex Danilo, Canon Information Systems Research Australia
- Thomas DeWeese, Eastman Kodak
- David Dodds, Lexica
- Andrew Donoho, IBM
- David Duce, Oxford Brookes University
- Jerry Evans, Sun Microsystems
- Jon Ferraiolo, Adobe Systems Inc.
- Darryl Fuller, Schema Software
- 藤沢 淳 (FUJISAWA Jun), Canon
- Scott Furman, Netscape Communications Corporation
- Brent Getlin, Macromedia
- Peter Graffagnino, Apple
- Rick Graham, BitFlash
- Vincent Hardy, Sun Microsystems Inc.
- Vincent Hardy, Sun Microsystems Inc.
- 端山 貴也 (HAYAMA Takanari), KDDI Research Labs
- Lofton Henderson, OASIS
- Jan Christian Herlitz, Excsoft
- Alan Hester, Xerox Corporation
- Bob Hopgood, RAL (CCLRC)
- 石川 雅康 (ISHIKAWA Masayasu), W3C
- Dean Jackson, W3C/CSIRO (*W3C Team Contact*)
- Christophe Jolif, ILOG S.A.
- Lee Klosterman, Hewlett-Packard
- 小林 亜令 (KOBAYASHI Arei), KDDI Research Labs
- Thierry Kormann, ILOG S.A.
- Yuri Khramov, Schema Software
- Kelvin Lawrence, IBM
- Håkon Lie, Opera
- Chris Lilley, W3C (*Working Group Chair*)
- Philip Mansfield, Schema Software
- Kevin McCluskey, Netscape Communications Corporation
- 水口 充 (MINAKUCHI Mitsuru), Sharp Corporation
- Luc Minnebo, Agfa-Gevaert N.V.
- Tuan Nguyen, Microsoft Corporation
- 小野 修一郎 (ONO Shuichiro), Sharp Corporation
- Antoine Quint, Fuchsia Design (formerly of ILOG)
- 相良 毅 (SAGARA Takeshi), KDDI Research Labs
- Troy Sandal, Visio Corporation
- Peter Santangeli, Macromedia
- Haroon Sheikh, Corel Corporation
- Brad Sipes, ZOOMON AB
- Peter Sorotokin, Adobe Systems Inc.
- Gavriel State, Corel Corporation
- Robert Stevahn, Hewlett-Packard
- Timothy Thompson, Eastman Kodak
- 上田 宏高 (UEDA Hirotaka), Sharp Corporation
- Rick Yardumian, Canon Development Americas
- Charles Ying, Openwave Systems Inc.
- Shenxue Zhou, Quark



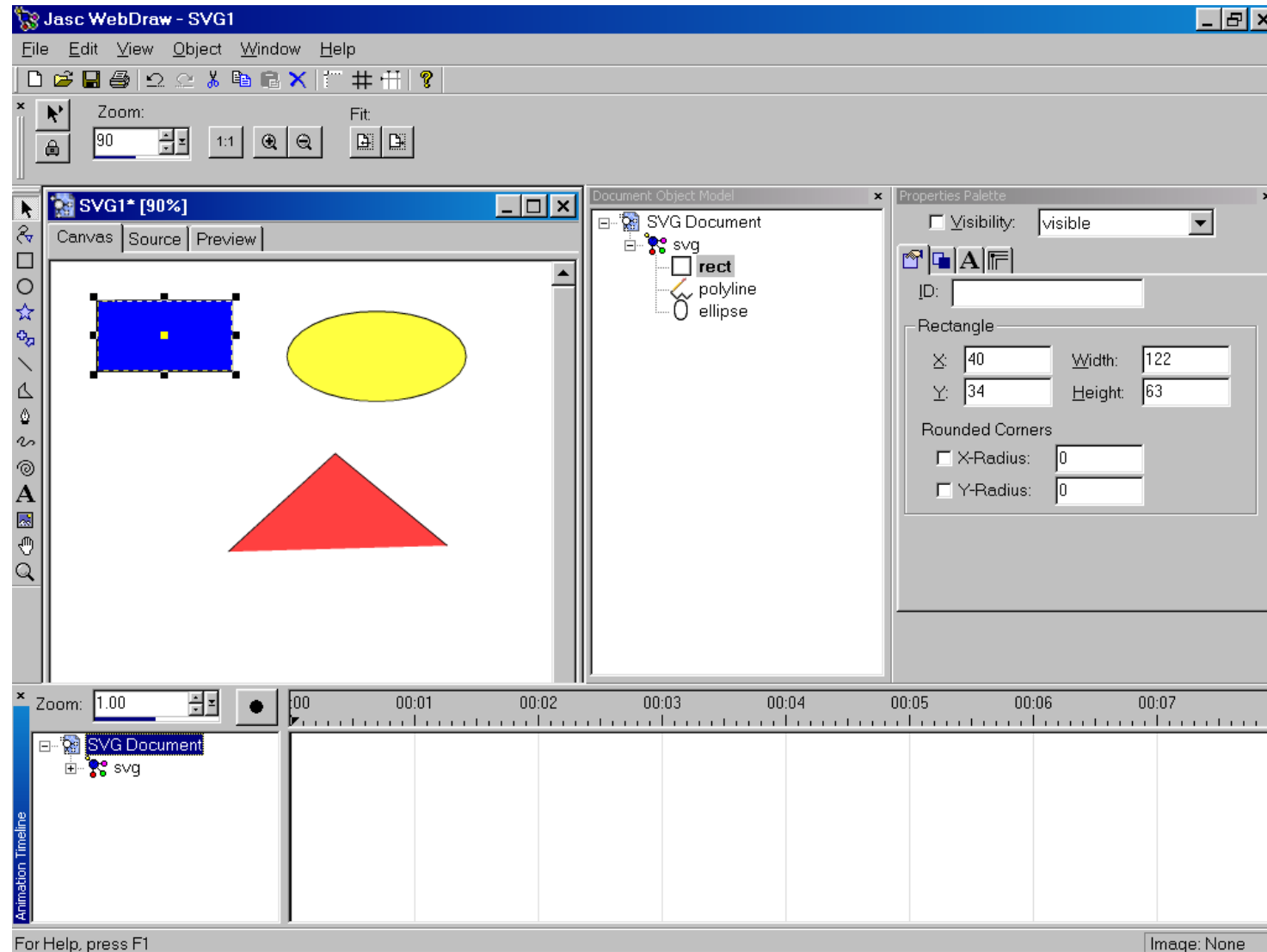
Software zur SVG-Erstellung

⇒ Adobe Illustrator (adobe.com):



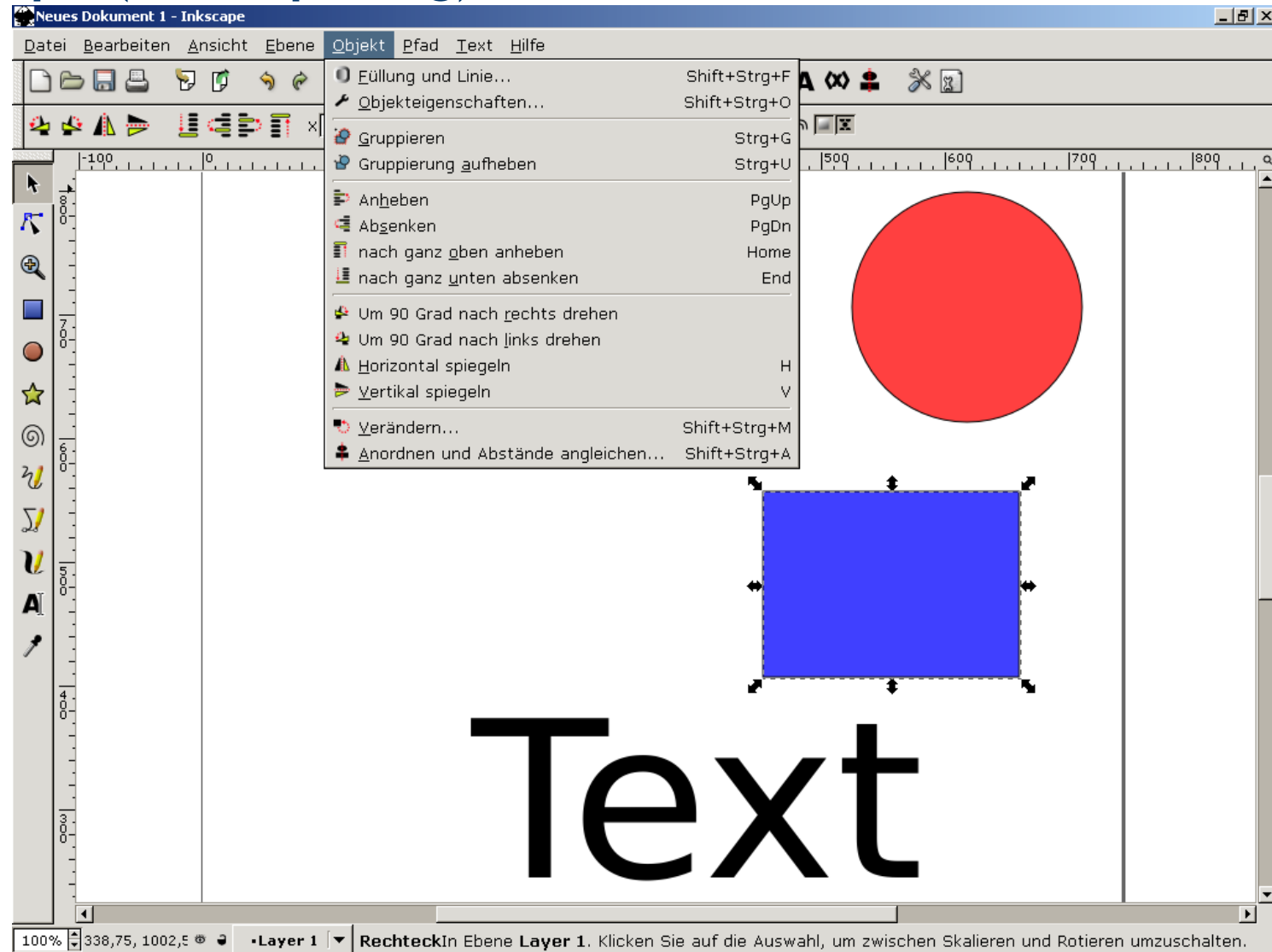
Software zur SVG-Erstellung

⇒ Jasc WebDraw (jasc.com):



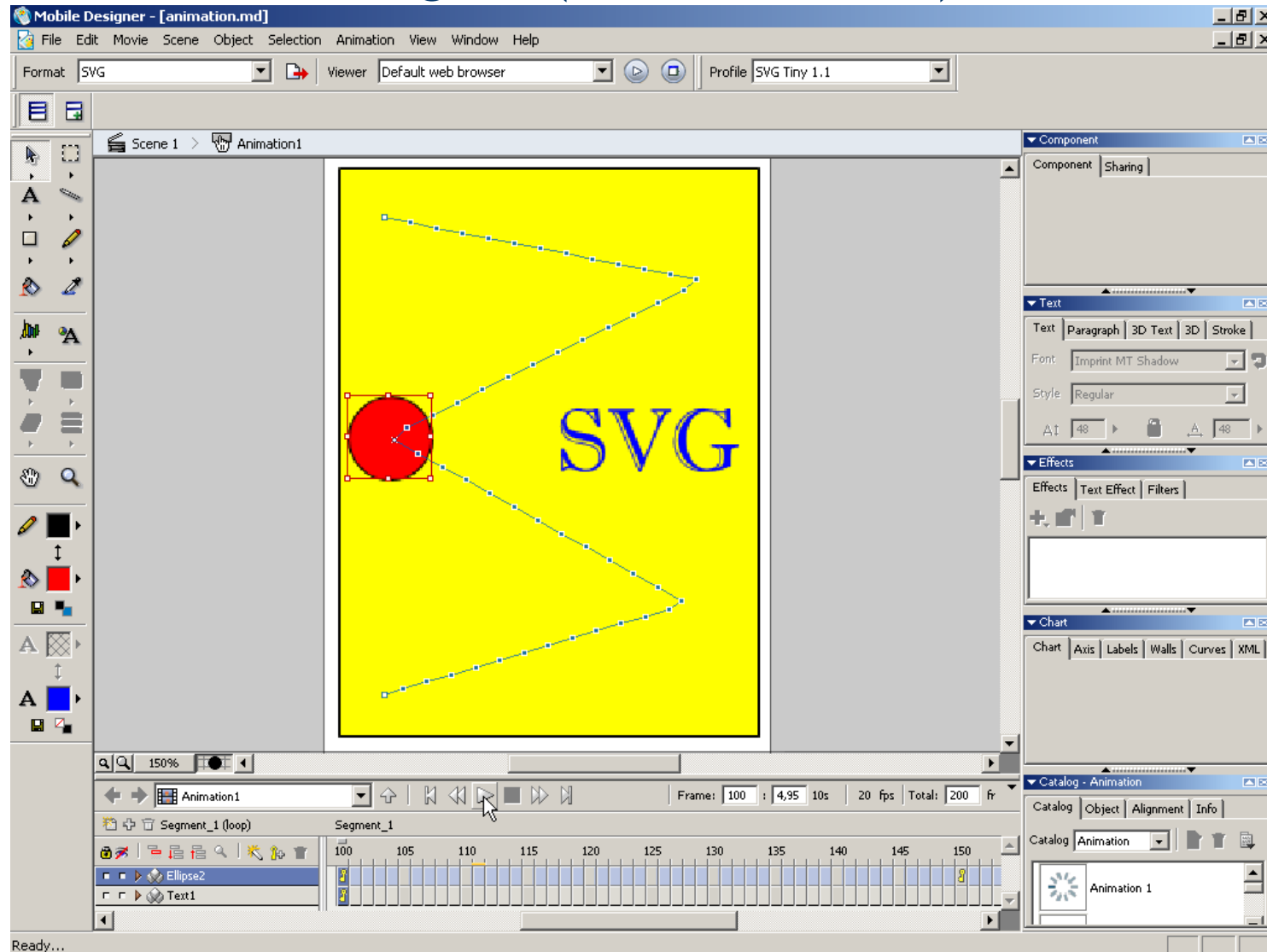
Software zur SVG-Erstellung

⇒ Inkscape (inkscape.org):



Software zur SVG-Erstellung

⇒ Beatware Mobile Designer (beatware.com):



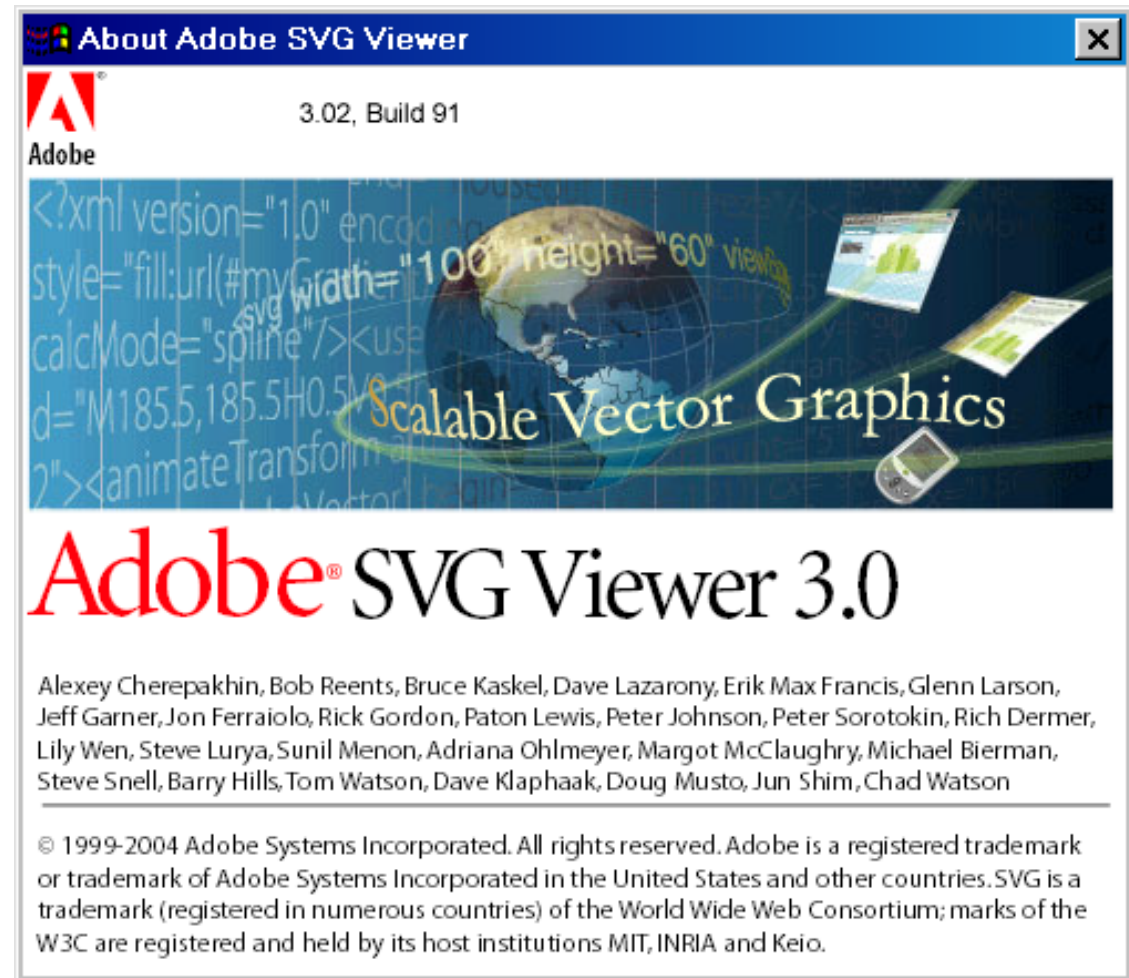
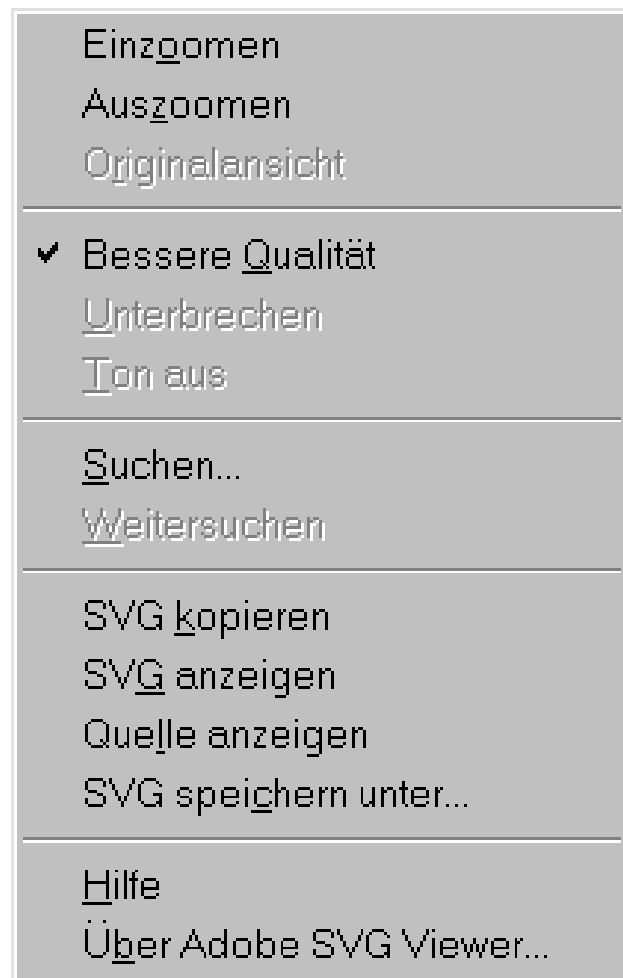
Software zur SVG-Darstellung

- ⇒ Adobe SVG Viewer (ASV 3.0x, 6.0 preview 1) als Browser Plug-in für IE, Netscape/Mozilla, Opera
- ⇒ Corel SVG Viewer (CSV 2.0x)
- ⇒ Browser mit nativer SVG-Unterstützung:
 - spezielle Mozilla/Firefox-Builds (ab FF 1.5 SVG nativ)
 - Opera ab Version 8 (Beta 3) stellt SVG-Tiny 1.1 dar
- ⇒ Batik Squiggle aus dem Batik SVG Toolkit (Java-basiert)
- ⇒ Einbindung über Java-Applets wie z. B. TinyLine SVG Player for Web
- ⇒ Handys mit integriertem Viewer (K700i, S65, ...)



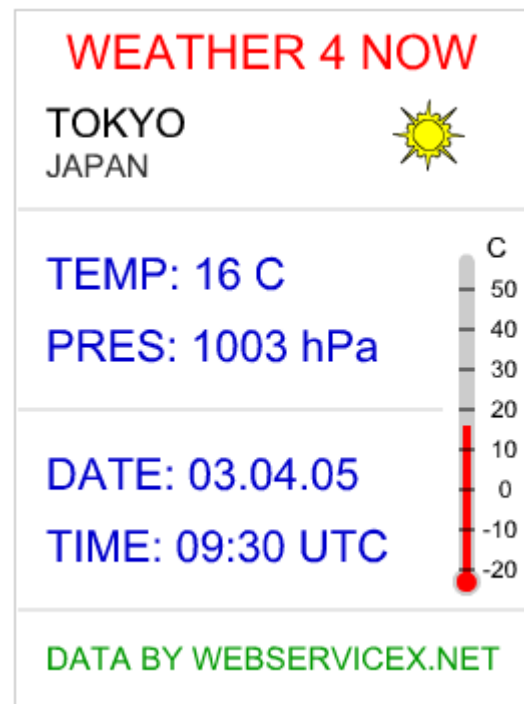
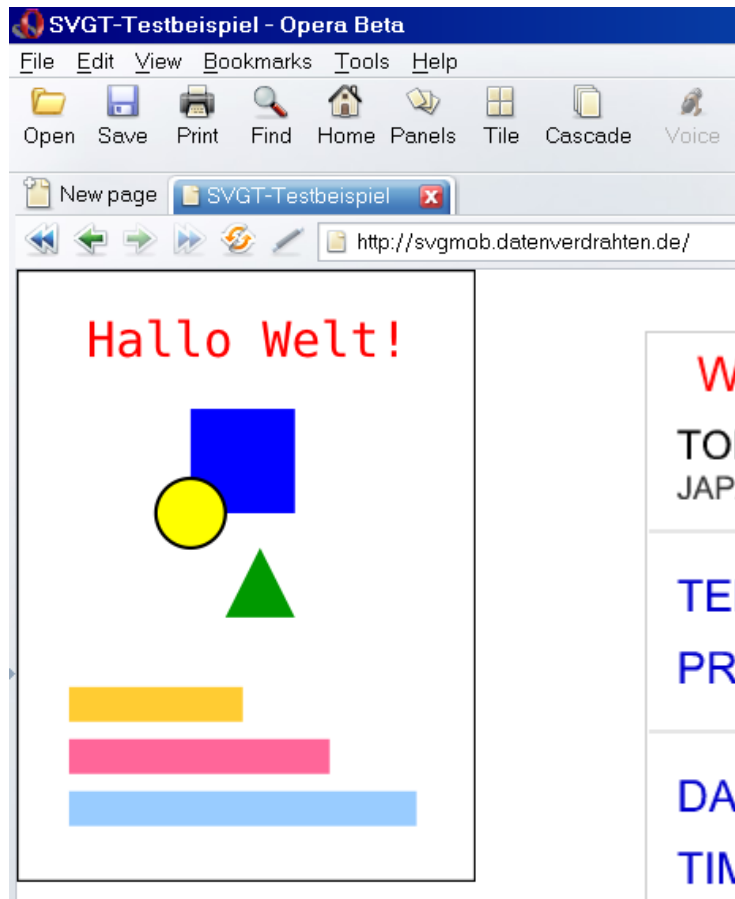
Software zur SVG-Darstellung

⇒ ASV 3.0x - Kontextmenü:



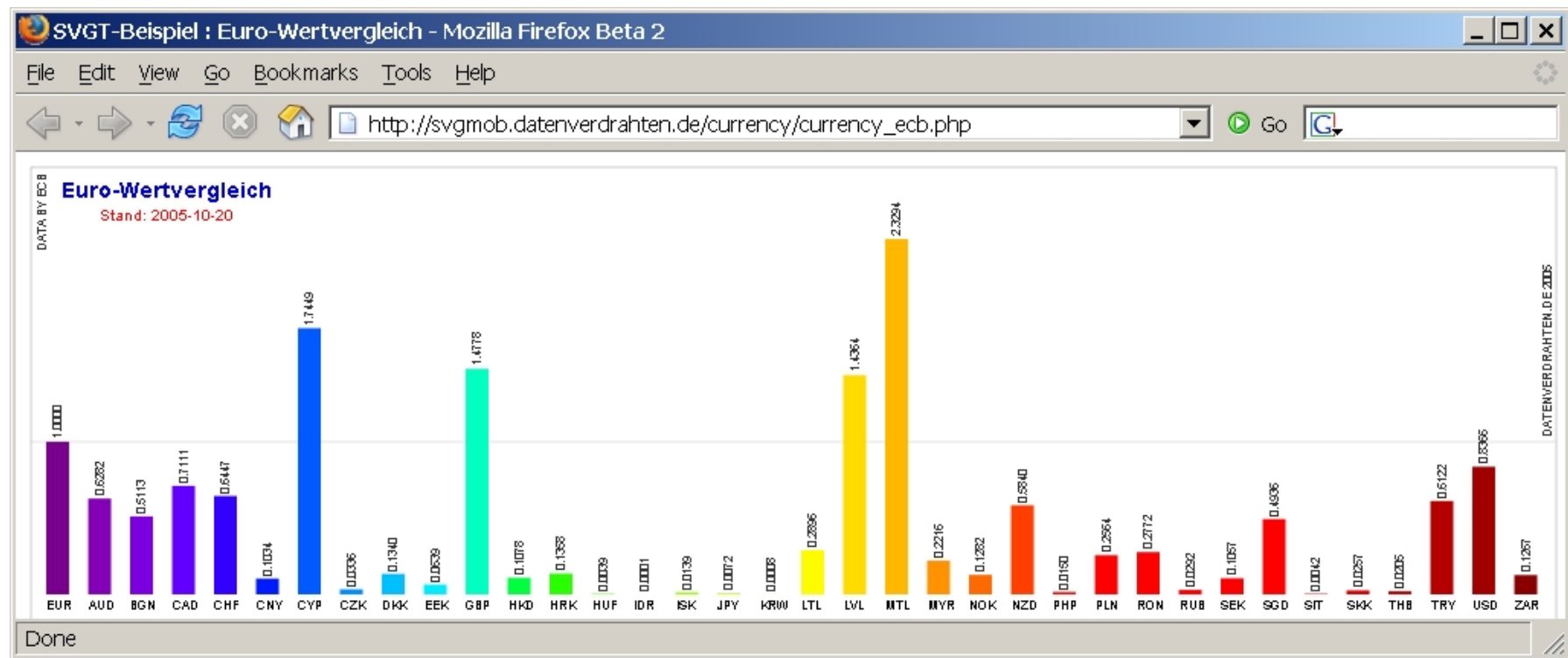
Software zur SVG-Darstellung

⇒ Opera 8 | TinyLine Applet | Handy:



Software zur SVG-Darstellung

⇒ Mozilla Firefox ab Version 1.5 (aktuell Beta 2) :



Syntaxregeln zur Erstellung von XML-Dokumenten

⇒ Kriterien für die Wohlgeformtheit (well-formedness):

- Es gibt genau ein Wurzelement, welches alle Inhalte umschließt.
- Alle Elemente sind durch Anfangs- und Endtags korrekt ausgezeichnet. (leere Elemente in der Form `<el ... ></el>` oder `<el ... />`)
- Alle Elemente sind korrekt ineinander verschachtelt. (`<a>...` und nicht `<a>...`)
- Attributwerte stehen in Anführungszeichen (paarweise “...” oder ‘...’).
- Regeln für die Schreibweise werden eingehalten.
- Reservierte Zeichen sind je nach Kontext ggf. maskiert:
`< = < ;` | `> = > ;` | `& = & ;` | `“ = " ;` | `‘ = ' ;`
- Ungeparste Inhalte in CDATA-Abschnitten möglich: `<![CDATA[...]]>`
- Kommentare (einzeilig/mehrzeilig): `<!-- ... -->`



SVG im Überblick

⇒ SVG-XML-Grundgerüst:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
```

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"  
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

```
<svg xmlns="http://www.w3.org/2000/svg"  
  xmlns:xlink="http://www.w3.org/1999/xlink">
```

```
<title>optionaler Titel</title>
```

```
<desc>optionale Beschreibung</desc>
```

```
<defs>Stylesheets, Skriptcode, Referenzen</defs>
```

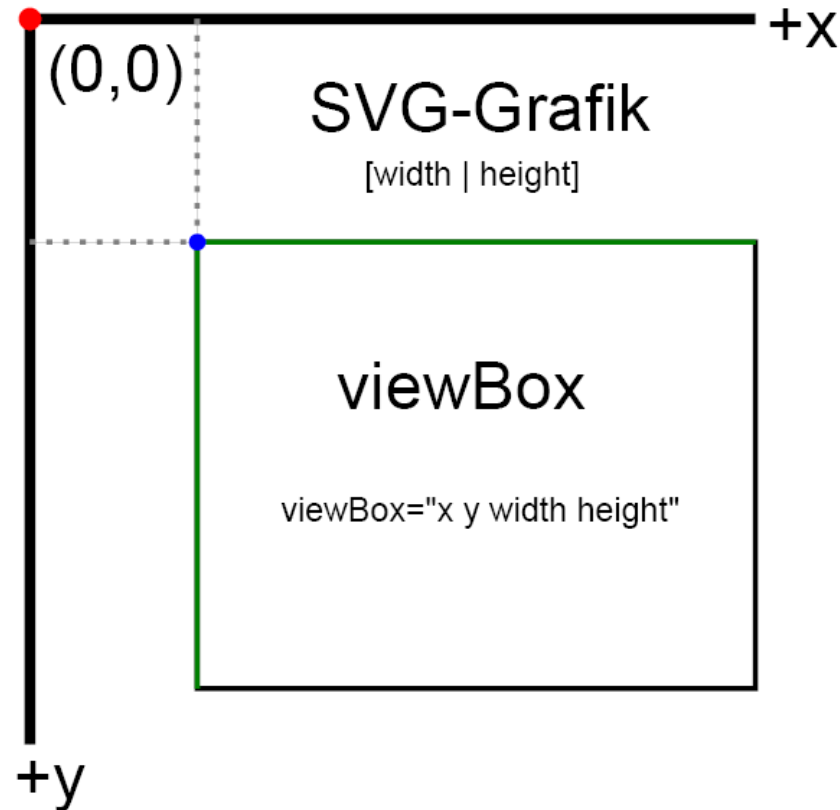
```
<!-- weitere SVG-Inhalte -->
```

```
</svg>
```



SVG im Überblick

⇒ SVG-Koordinatensystem:

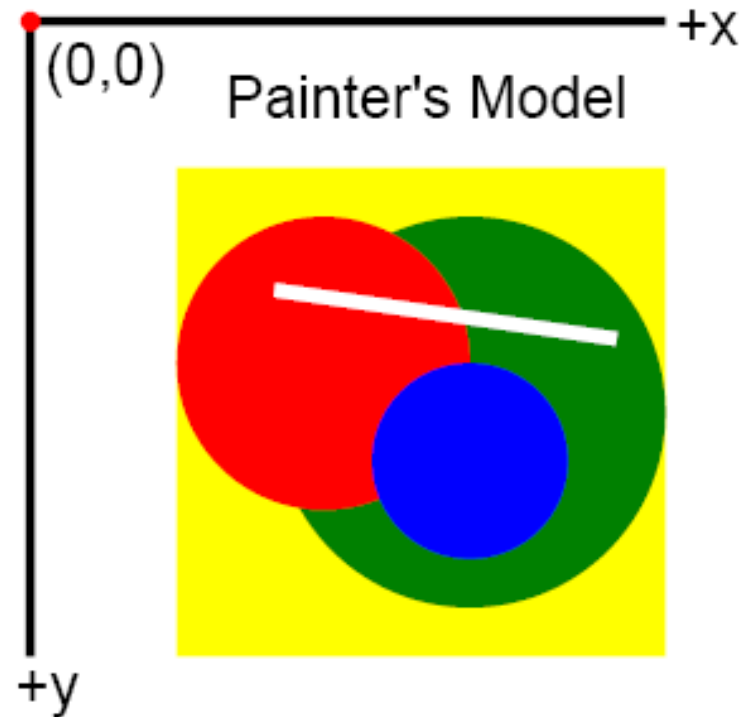


⇒ Einheiten (für Koordinaten und Längen, analog zu CSS):
%, em, ex, px, pc, pt, mm, cm, in [Zahlenwert ohne Einheit = px]



SVG im Überblick

⇒ SVG-Grafikaufbau:



⇒ Grafikinhalte werden in der Reihenfolge ihrer Definition im XML-Code auf der Zeichenfläche angeordnet / überlagert.

SVG im Überblick

⇒ Grafische Grundformen: Rechteck

```
<rect x="..." y="..." width="..." height="..."  
      rx="..." ry="..." />
```

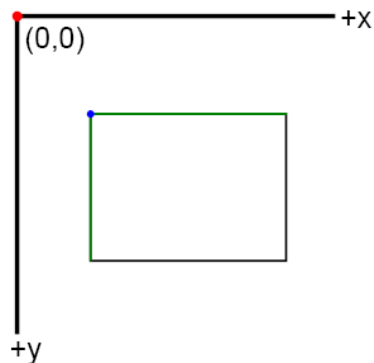
x = x-Koordinate

y = y-Koordinate

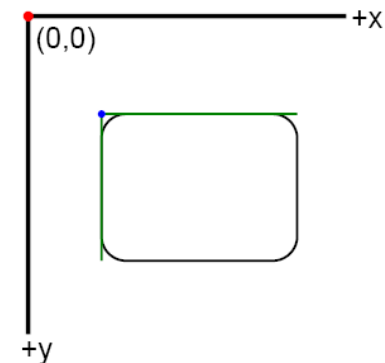
width = Breite

height = Höhe

rx, ry = Radien für abgerundete Ecken (optional)



```
<rect x="30" y="40"  
      width="80" height="60" />
```



```
<rect x="30" y="40" width="80"  
      height="60" rx="10" ry="10" />
```

SVG im Überblick

⇒ Grafische Grundformen: Kreis

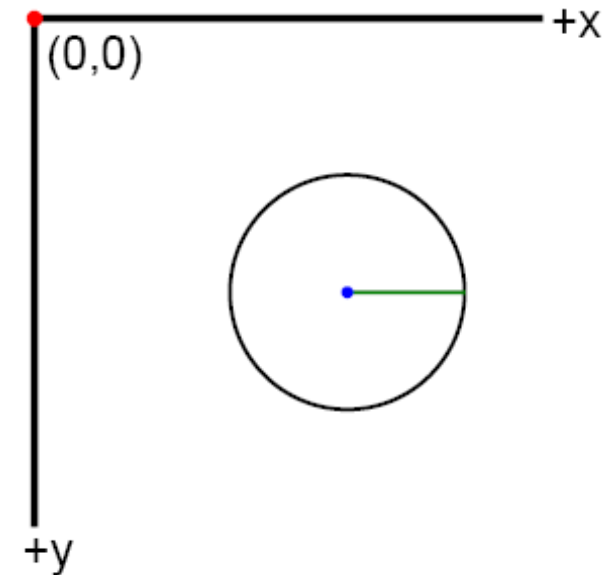
```
<circle cx="..." cy="..." r="..." />
```

cx = x-Koordinate des Mittelpunktes

cy = y-Koordinate des Mittelpunktes

r = Radius

```
<circle cx="80" cy="70" r="30" />
```



SVG im Überblick

⇒ Grafische Grundformen: Ellipse

```
<ellipse cx="..." cy="..." rx="..." ry="..." />
```

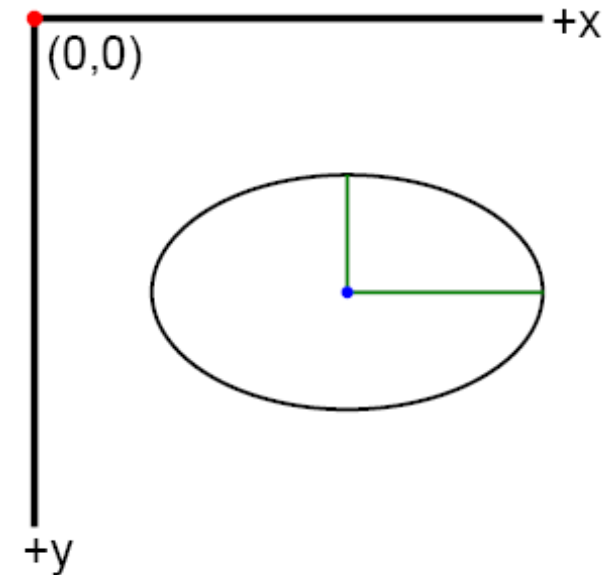
cx = x-Koordinate des Mittelpunktes

cy = y-Koordinate des Mittelpunktes

rx = Radius der Halbachse in x-Richtung

ry = Radius der Halbachse in y-Richtung

```
<ellipse cx="80" cy="70"  
  rx="50" ry="30" />
```



SVG im Überblick

⇒ Grafische Grundformen: Linie

```
<line x1="..." y1="..." x2="..." y2="..." />
```

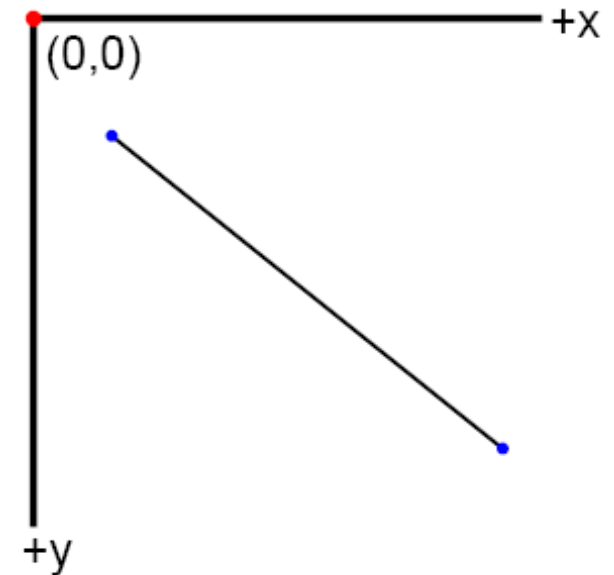
x1 = x-Koordinate des ersten Punktes

y1 = y-Koordinate des ersten Punktes

x2 = x-Koordinate des zweiten Punktes

y2 = y-Koordinate des zweiten Punktes

```
<line x1="20" y1="30"  
      x2="120" y2="110" />
```



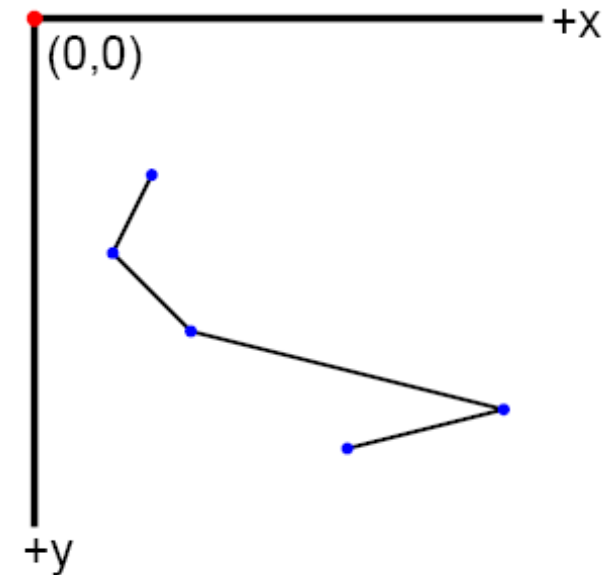
SVG im Überblick

⇒ Grafische Grundformen: Mehrfachlinie

```
<polyline points="x1,y1 x2,y2, ... xn,yn"/>
```

`points` = Liste der einzelnen durch Linien zu verbindenden Punkte (x,y)
Trennung durch Kommata / Leerzeichen

```
<polyline points="30,40 20,60  
40,80, 120,100 80,110"/>
```



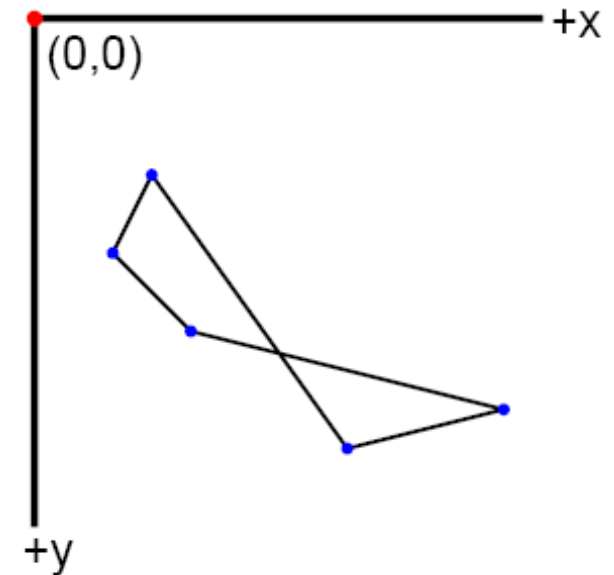
SVG im Überblick

⇒ Grafische Grundformen: Polygon (n-Eck, $n \geq 3$)

```
<polygon points="x1,y1 x2,y2, ... xn,yn"/>
```

`points` = Liste der einzelnen durch Linien zu verbindenden Punkte (x,y)
Trennung durch Kommata / Leerzeichen
letzter Punkt wird mit erstem Punkt verbunden

```
<polygon points="30,40 20,60  
40,80, 120,100 80,110"/>
```



SVG im Überblick

⇒ Grafische Spezialformen: Pfade

```
<path d="..." />
```

d = Pfad-Daten mit diesen Parametern:

[M,m]x,y	: moveto
[L,l]x,y	: lineto
[H,h]x	: horizontal lineto
[V,v]y	: vertical lineto
[C,c]x1,y1 x2,y2 x,y	: cubic Bézier curveto
[S,s]x2,y2 x,y	: smooth cubic curveto
[Q,q]x1,y1 x,y	: quadratic Bézier curveto
[T,t]x,y	: smooth quadratic curveto
[A,a]rx,ry x-axis-rotation large-arc,sweep x,y	: elliptical arc
[Z,z]	: closepath
[...]groß: absolute, klein: relative Koordinaten	

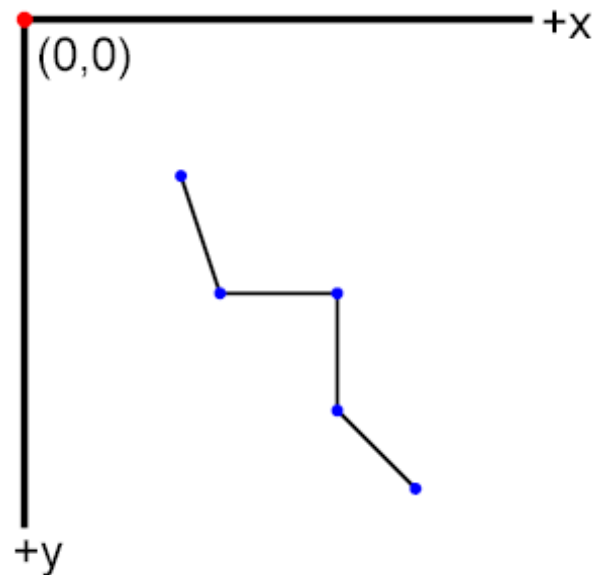


SVG im Überblick

⇒ Grafische Spezialformen: Pfade

```
<path d="M40,40 L50,70 H80 V100 L100,120"/>
```

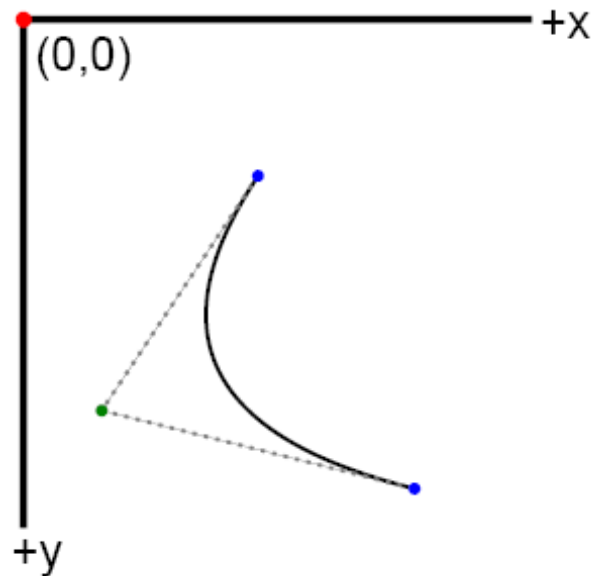
```
<path d="M40,40 l10,30 h30 v30 l20,20"/>
```



SVG im Überblick

⇒ Grafische Spezialformen: Pfade

```
<path d="M60,40 Q20,100 100,120"/>
```



quadratische Bézier-Kurve
mit 1 Stützpunkt (Anfasser)
= P_1

Pierre Bézier (1910 bis 1999),
französischer Mathematiker,
beschrieb Anfang der 1960er
die Bézierkurven-Polynome.

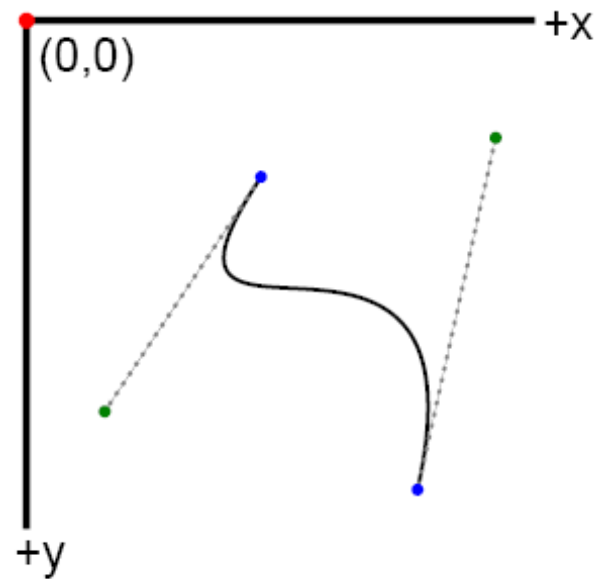
$$C(t) = (1-t)^2 P_0 + 2t(1-t) P_1 + t^2 P_2, t \in [0,1]$$



SVG im Überblick

⇒ Grafische Spezialformen: Pfade

```
<path d="M60,40 C20,100 120,30 100,120"/>
```



kubische Bézier-Kurve
mit 2 Stützpunkten
(Anfassern) = P_1 und P_2

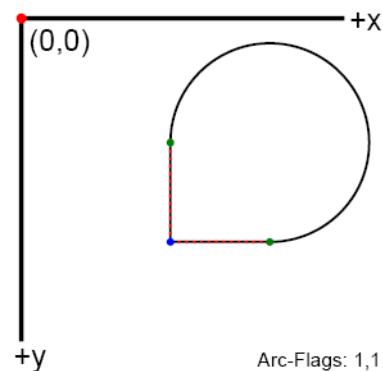
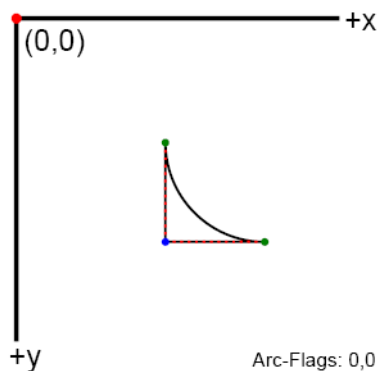
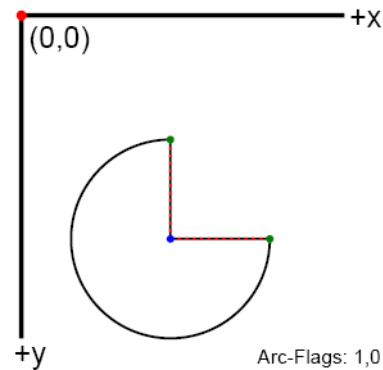
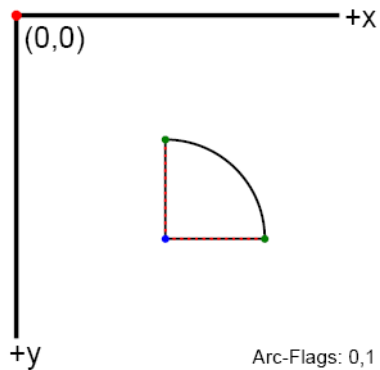
$$C(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3, t \in [0,1]$$



SVG im Überblick

⇒ Grafische Spezialformen: Pfade

```
<path d="M60,90 L60,50 A40,40 0 0,1 100,90Z"/>
```



large-arc
(Größe):

(0) 0...180°
(1) >180°

sweep
(Richtung):

(0) negativ
(1) positiv

x-Rotation



SVG im Überblick

⇒ Grafikobjekte / Inhalte gruppieren:

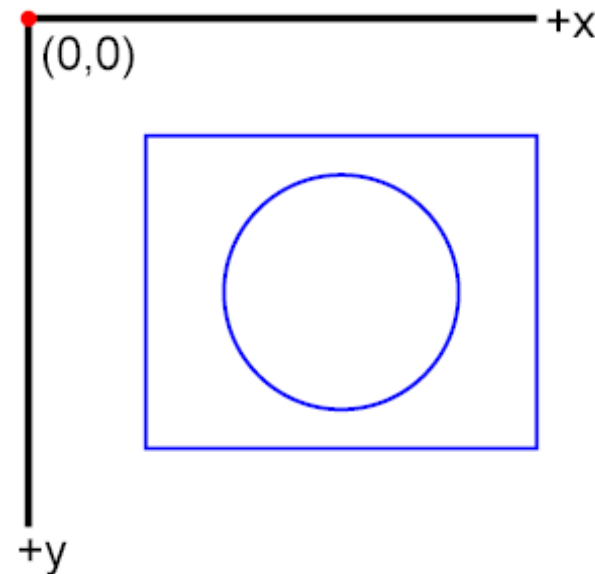
```
<g fill="none" stroke="blue" stroke-width="1">
```

```
<circle cx="80" cy="70" r="30"/>
```

```
<rect x="30" y="30" width="100" height="80"/>
```

```
</g>
```

Kreis und Rechteck erhalten
als Mitglieder der Gruppe
(Kindelemente des g-Elements)
dieselben Eigenschaften.



SVG im Überblick

⇒ Textinhalte (einzelne Texte oder mehrere Zeilen):

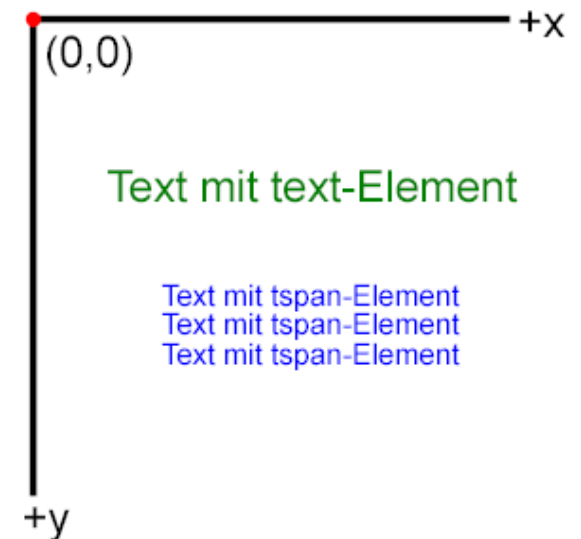
```
<text x="..." y="...">Textinhalt</text>
```

x = x-Koordinate der Grundlinie

y = y-Koordinate der Grundlinie

```
<text x="..." y="...">  
  <tspan x="..." dy="...">...</tspan>  
  <tspan x="..." dy="...">...</tspan>  
  <tspan x="..." dy="...">...</tspan>  
</text>
```

dy = vertikaler Zeilenabstand
(z. B. Angabe in em)

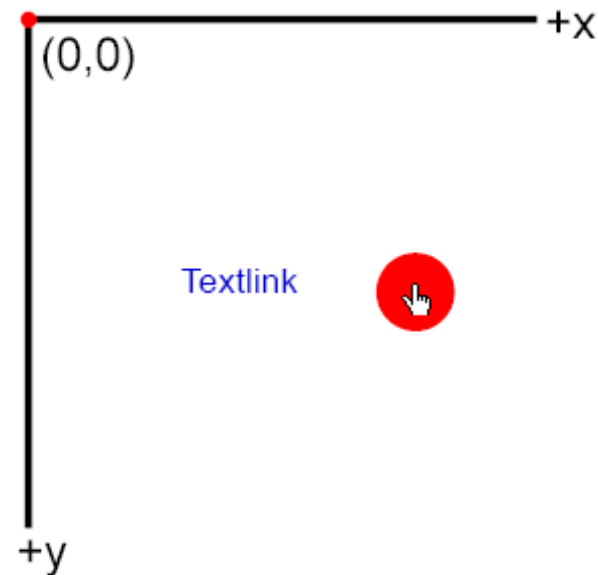


SVG im Überblick

⇒ Hyperlinks (ähnlich zu Links in HTML):

```
<a xlink:href="...">  
  <text x="..." y="...">Linktext</text>  
</a>
```

```
<a xlink:href="...">  
  <!-- Grafikobjekt,  
       z. B. Kreis -->  
  <circle ... />  
</a>
```



Hinweis: XLink-Namensraum im Wurzelement angeben!



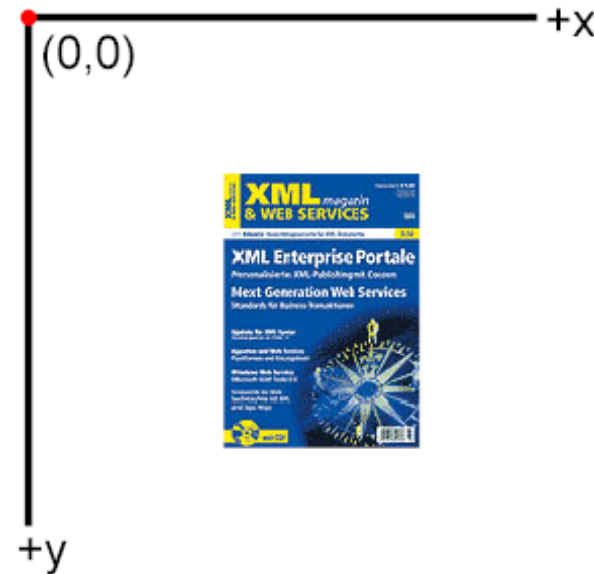
SVG im Überblick

⇒ externe Bilder (ähnlich zu Bildern in HTML):

```
<image xlink:href="..."  
  x="..." y="..." width="..." height="..." />
```

Bildformate:

- GIF
- JPEG
- PNG
- SVG



Hinweis: XLink-Namensraum im Wurzelement angeben!

SVG im Überblick

⇒ Formatierung mit Präsentationsattributen:

- roter Kreis mit blauem Rand der Stärke 2 Pixel:

```
<circle cx="..." cy="..." r="..." fill="red"  
stroke="blue" stroke-width="2"/>
```

- grüner Text, 14 Pixel hoch in der Schriftart Arial (bzw. Alternativen):

```
<text x="..." y="..." fill="#090"  
font-family="Arial, sans-serif"  
font-size="14">Text in SVG</text>
```



SVG im Überblick

⇒ Formatierung mit CSS-Eigenschaften (style-Attribut):

- roter Kreis mit blauem Rand der Stärke 2 Pixel:

```
<circle cx="..." cy="..." r="..." style="fill: red;  
stroke: blue; stroke-width: 2px"/>
```

- grüner Text, 14 Pixel hoch in der Schriftart Arial (bzw. Alternativen):

```
<text x="..." y="..." style="fill: #090;  
font-family: Arial, sans-serif;  
font-size: 14px">Text in SVG</text>
```



SVG im Überblick

⇒ Formatierung mit CSS-Eigenschaften (style-Element):

```
<defs>
  <style type="text/css">
    <![CDATA[
      /* CSS-Definitionen */

      circle
      {
        fill: red;
        stroke: blue;
        stroke-width: 2px;
      }

    ]]>
  </style>
</defs>
```

- Selektoren aus CSS 2
- Eigenschaften z. T. aus CSS 2 übernommen und SVG-eigene



SVG im Überblick

⇒ Formatierung mit CSS-Eigenschaften (externes Stylesheet):

```
<?xml version="1.0" ... ?>  
<?xml-stylesheet href="datei.css" type="text/css"?>  
<!DOCTYPE svg ...>
```

```
<svg ...>
```

```
</svg>
```



```
circle  
{  
  fill: red;  
  stroke: blue;  
  stroke-width: 2px;  
}  
...
```



SVG im Überblick

⇒ Wichtige Attribute bzw. CSS-Eigenschaften:

- Füllfarbe: **fill** (keine Füllung mit Wert *none*)
- Rahmenfarbe: **stroke**
- Rahmenstärke: **stroke-width**
- Rahmen/Linien gestrichelt: **stroke-dasharray**

- Deckkraft: **opacity** (0...1, 1=volle Deckkraft)
auch **fill-opacity** und **stroke-opacity**

- Schriftformatierung: **font-size**, **font-family**, **font-style**
- Textauszeichnungen: **text-decoration**, **text-transform**

- Anzeige von Objekten: **display**, **visibility**

- weitere aus CSS bekannte sowie für SVG definierte Eigenschaften und Werte



SVG im Überblick

⇒ Farbwerte (Attribute / Eigenschaften fill, stroke usw.):

Beispiel Farbe rot

- hexadezimal 6-stellig (#RRGGBB): **#FF0000**
- hexadezimal 3-stellig (#RGB): **#F00**
- dezimale RGB-Schreibweise: **rgb(255,0,0)**
- prozentuale RGB-Schreibweise: **rgb(100%,0%,0%)**
- Farbwort: **red**



SVG im Überblick

⇒ SVG-Einbindung in HTML:

- W3C-konform über die Element object oder iframe:

```
<object data="datei.svg" width="..." height="..."  
  type="image/svg+xml">  
  <!-- Alternativinhalt -->  
</object>
```

```
<iframe src="datei.svg" width="..." height="..."  
  frameborder="0">  
  <!-- Alternativinhalt -->  
</iframe>
```

- ggf. für Netscape 4.x mittels embed oder object/embed-Kombination:

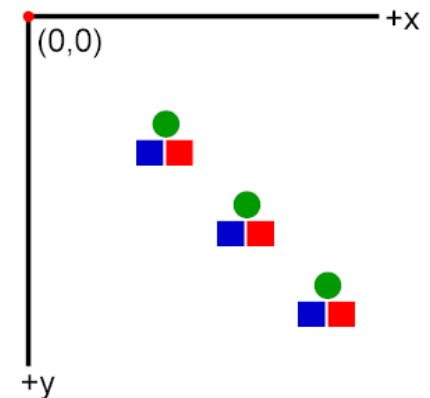
```
<embed src="datei.svg" width="..." height="..."  
  type="image/svg+xml">  
  <!-- Alternativinhalt -->  
</embed>
```



SVG im Überblick

⇒ Spezielle SVG-Elemente: **symbol** (Symbolobjekte)

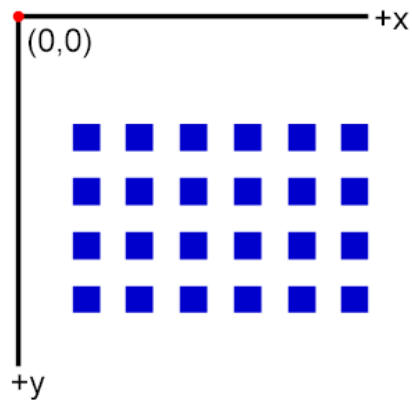
```
<defs>
  <symbol id="einsymbol">
    <circle cx="11" cy="10" r="5" fill="#090"/>
    <rect x="0" y="16" width="10" height="10"
      fill="#00C"/>
    <rect x="11" y="16" width="10" height="10"
      fill="#F00"/>
  </symbol>
</defs>
...
<use xlink:href="#einsymbol" x="40" y="30"/>
<use xlink:href="#einsymbol" x="70" y="60"/>
<use xlink:href="#einsymbol" x="100" y="90"/>
```



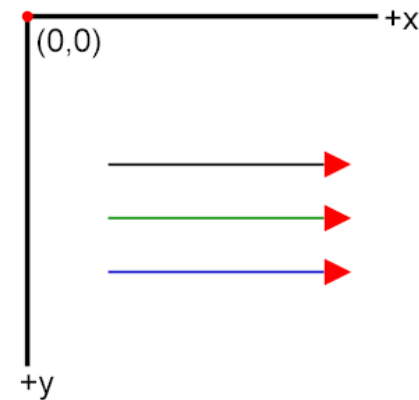
SVG im Überblick

⇒ Weitere spezielle SVG-Elemente:

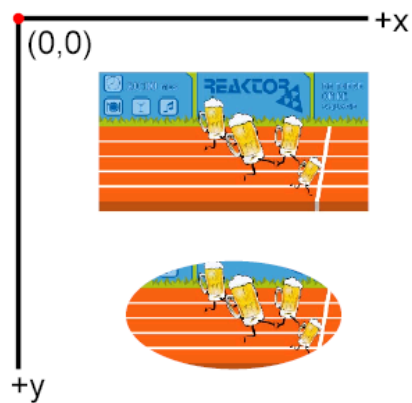
pattern



marker



clipPath



mask



SVG im Überblick

⇒ Farbverläufe: linearGradient / radialGradient

```
<defs>
  <linearGradient id="eineID" ...>
    <stop offset="0%" stop-color="..." />
    ...
    <stop offset="100%" stop-color="..." />
  </linearGradient>
</defs>
```

Attribute zur Steuerung von Verläufen:

- x1, y1, x2, y2 (jeweils 0...1 bzw. 0%...100%)
- spreadMethod="pad" | "repeat" | "reflect"
- stop-Element kann noch stop-opacity erhalten

Einbindung als Füllung: `fill="url(#eineID)"`

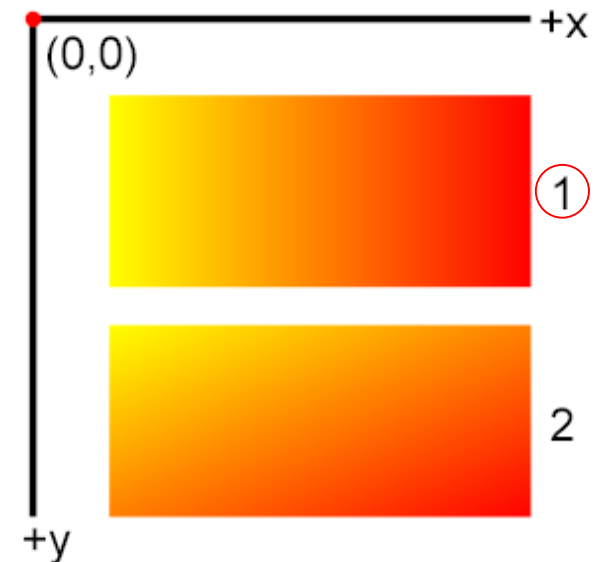


SVG im Überblick

⇒ Linerare Farbverläufe: linearGradient

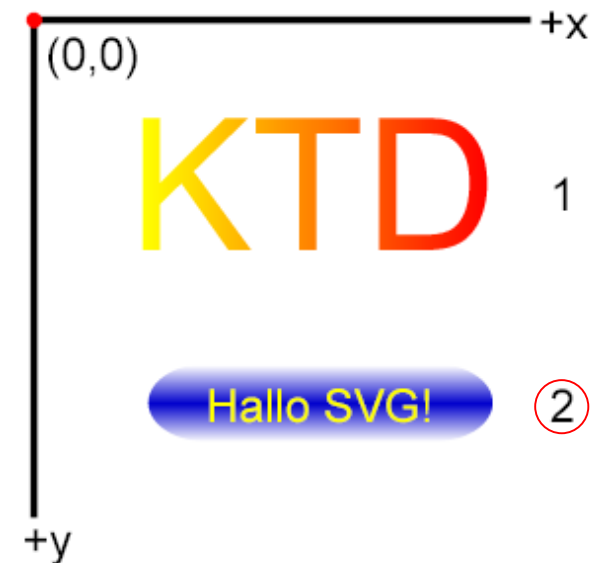
1

```
<linearGradient id="lin1">  
  <stop offset="0%" stop-color="#FF0"/>  
  <stop offset="100%" stop-color="#F00"/>  
</linearGradient>
```



2

```
<linearGradient id="vert">  
  x1="0%" y1="0%" x2="0%" y2="100%">  
  <stop offset="0%" stop-color="#FFF"/>  
  <stop offset="50%" stop-color="#00C"/>  
  <stop offset="100%" stop-color="#FFF"/>  
</linearGradient>
```

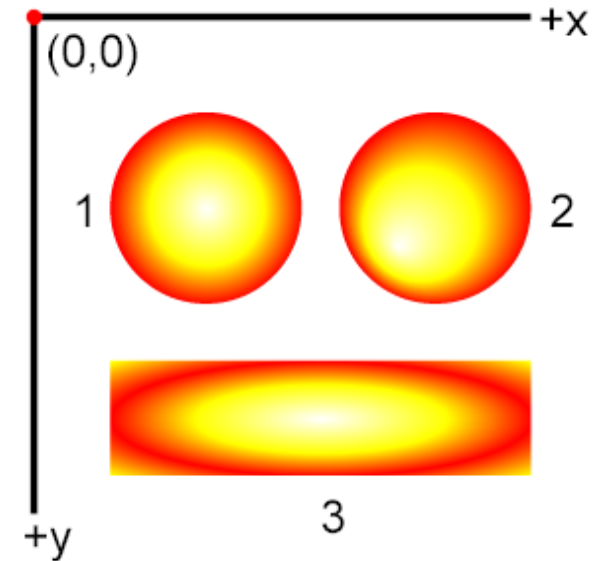


SVG im Überblick

⇒ Radiale Farbverläufe: radialGradient

2

```
<radialGradient id="rad2" fx="30%" fy="70%">  
  <stop offset="0%" stop-color="#FFF"/>  
  <stop offset="60%" stop-color="#FF0"/>  
  <stop offset="100%" stop-color="#F00"/>  
</radialGradient>
```



SVG im Überblick

⇒ Filter-Elemente: Allgemeines

```
<defs>
  <filter id="eineID" ...>
    <feFilterElement .../>
    ...
    <feFilterElement .../>
  </filter>
</defs>
```

Einbindung mit filter-Attribut:

```
filter="url(#eineID) "
```



SVG im Überblick

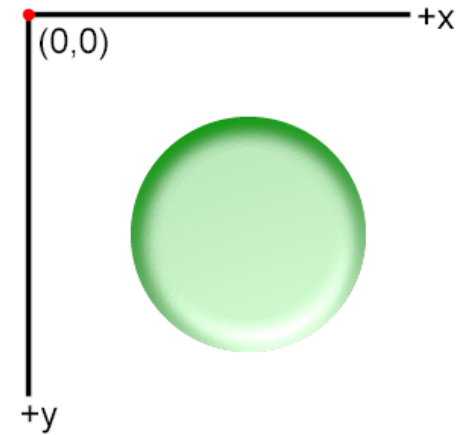
⇒ Filter-Elemente: Übersicht (16 Grundfilter und 8 weitere als Kindelemente)

<code><feBlend></code>	- zwei Objekte überblenden
<code><feColorMatrix></code>	- Farbänderung durch Matrixberechnung
<code><feComponentTransfer></code>	- Farbkomponenten neu berechnen
<code><feComposite></code>	- zwei Objekte zusammenfügen
<code><feConvolveMatrix></code>	- Unschärfe- und Prägefilter
<code><feDiffuseLighting></code>	- indirekter Beleuchtungseffekt
<code><feDisplacementMap></code>	- Pixelverschiebung
<code><feFlood></code>	- Objekt mit einer Farbe und Transparenz füllen
<code><feGaussianBlur></code>	- Weichzeichner (Schatten bzw. Unschärfe)
<code><feImage></code>	- Bild zuweisen/laden
<code><feMerge></code>	- beliebig viele Objekte zusammenfügen
<code><feMorphology></code>	- Verdicken und Verdünnen eines Objektes
<code><feOffset></code>	- Objekt verschieben
<code><feSpecularLighting></code>	- direkter Beleuchtungseffekt
<code><feTile></code>	- Bild innerhalb eines Objektes kacheln
<code><feTurbulence></code>	- Texturerzeugung (Perlin-Noise-Funktion)



SVG im Überblick

⇒ Filter-Elemente: Beispiel

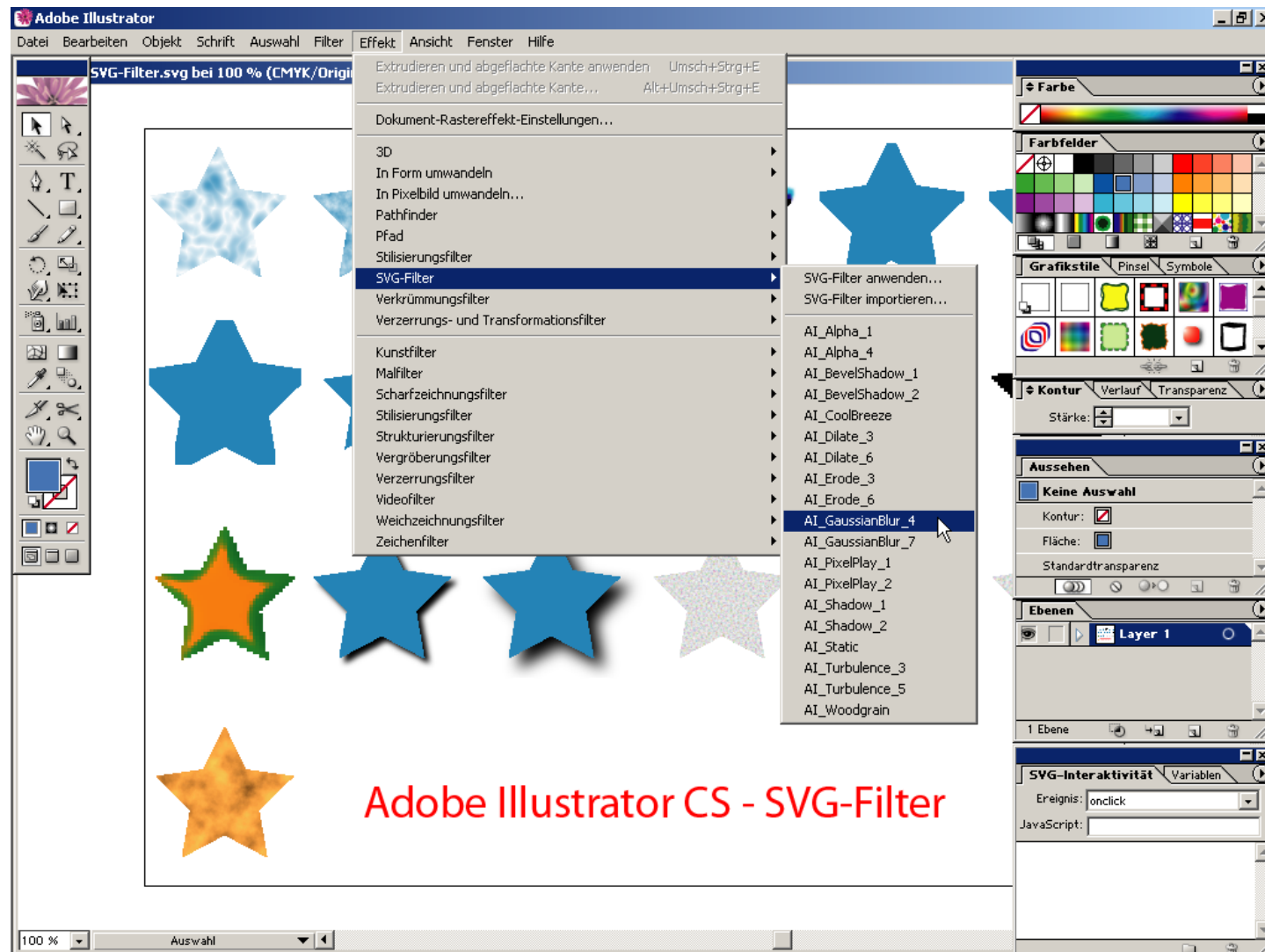


```
<defs>
  <filter id="einfileter">
    <feGaussianBlur in="SourceAlpha" stdDeviation="4"
      result="out1"/>
    <feSpecularLighting in="out1" specularExponent="10"
      surfaceScale="3" result="out2">
      <fePointLight x="150" y="300" z="300"/>
    </feSpecularLighting>
    <feComposite in="SourceGraphic" in2="out2"
      operator="arithmetic" k1="0" k2="1" k3="1" k4="0"/>
  </filter>
</defs>
...
<circle cx="75" cy="75" r="40" fill="#090" filter="url(#einfileter)"/>
```



SVG im Überblick

⇒ Filter-Elemente: SVG-Filter in Adobe Illustrator CS



Adobe Illustrator CS - SVG-Filter

SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Allgemeines
- Transformationen werden Elementen oder Gruppen von Elementen mittels **transform**-Attribut zugewiesen.
- Transformiert wird das Koordinatensystem und nicht das jeweilige Objekt!
- Transformationsarten:
 - rotate**(Winkel in Grad) oder **rotate**(Winkel in Grad [,Drehpunkt_x,Drehpunkt_y])
= Drehen
 - scale**(Faktor für x- und y-Richtung) bzw. **scale**(Faktor_x [,Faktor_y])
= Skalieren (Vergrößern >1 bzw. Verkleinern <1)
 - skewX**(Winkel in Grad)
skewY(Winkel in Grad)
= Neigen in Richtung der x- bzw. y-Achse
 - translate**(Distanz_x,Distanz_y)
= Verschieben in x- bzw. y-Richtung

Hinweis:

± Winkel in Grad für im
Uhrzeigersinn (+) bzw.
gegen den Uhrzeigersinn (-)



SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Allgemeines
- Transformationen können kombiniert (nacheinander ausgeführt) werden:
transform="operation2(...) operation1()" [Abarbeitung von rechts nach links]
- Transformationen lassen sich in Matrix-Schreibweisen formulieren:
transform="matrix(a,b,c,d,e,f)"

= 3x3-Matrix mit sechs relevanten Parametern

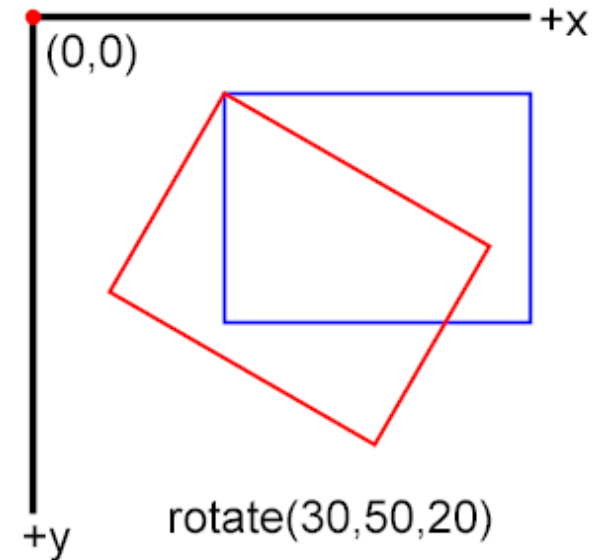
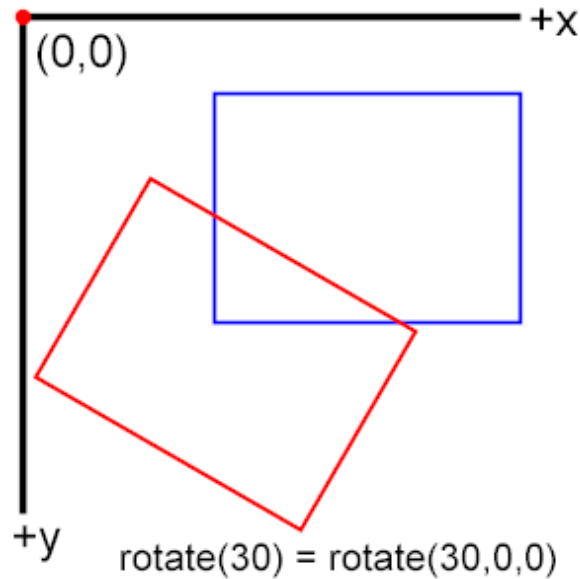
$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix}$$

mit folgenden Zuordnungsvorschriften

$\begin{bmatrix} \cos(a) & -\sin(a) & 0 \\ \sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & \tan(a) & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ \tan(a) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$
rotate	scale	skewX	skewY	translate

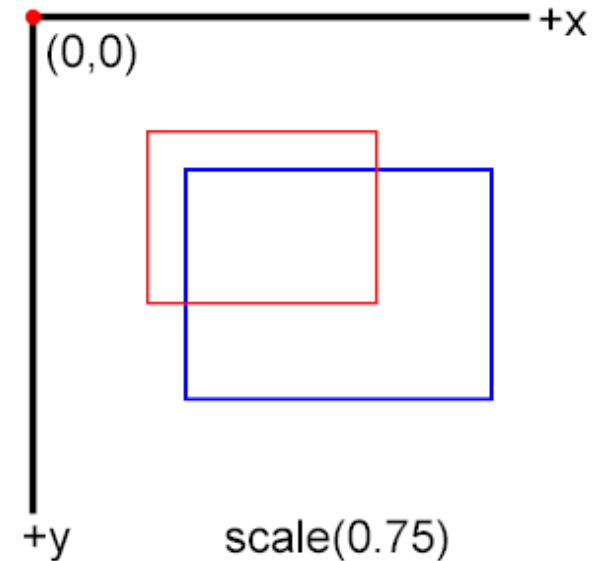
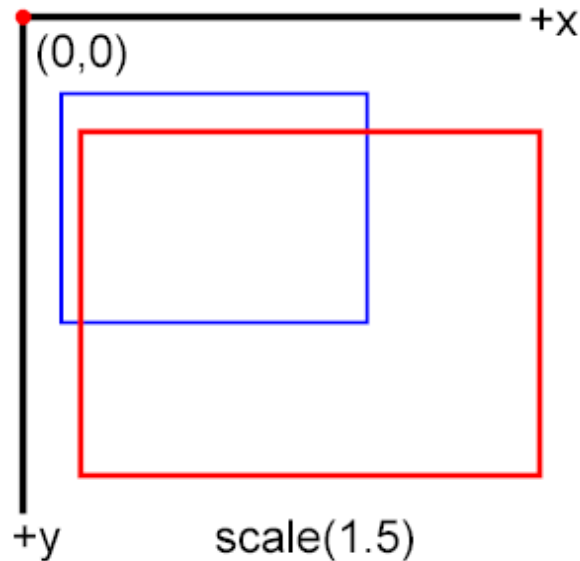
SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Beispiele
- **Rotation** um den Ursprung bzw. um einen anderen Drehpunkt:



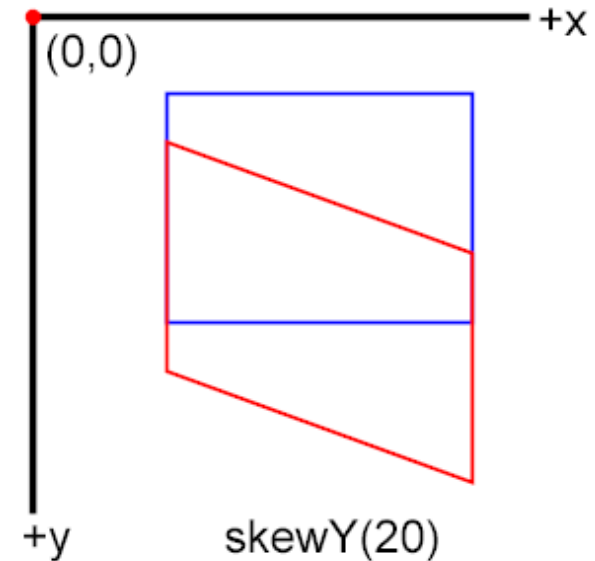
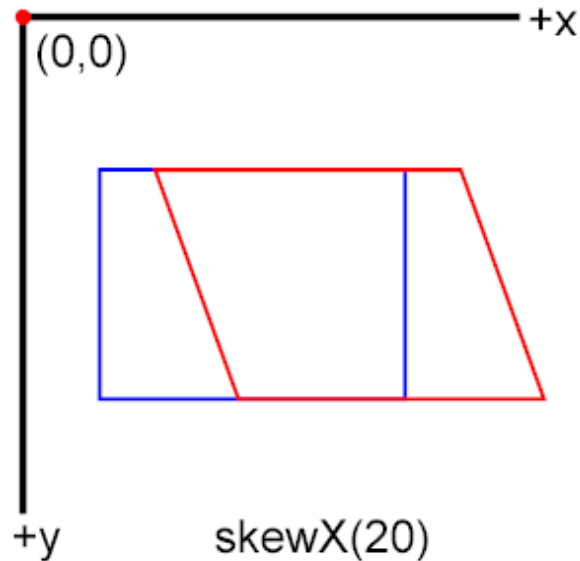
SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Beispiele
- **Skalierung** (Vergrößerung bzw. Verkleinerung):



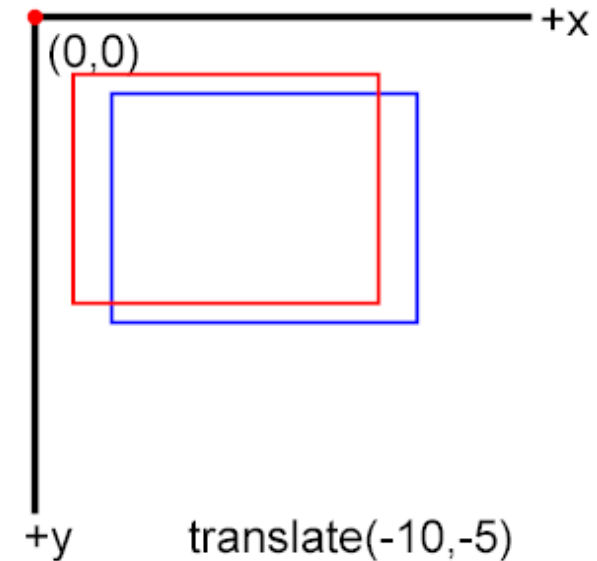
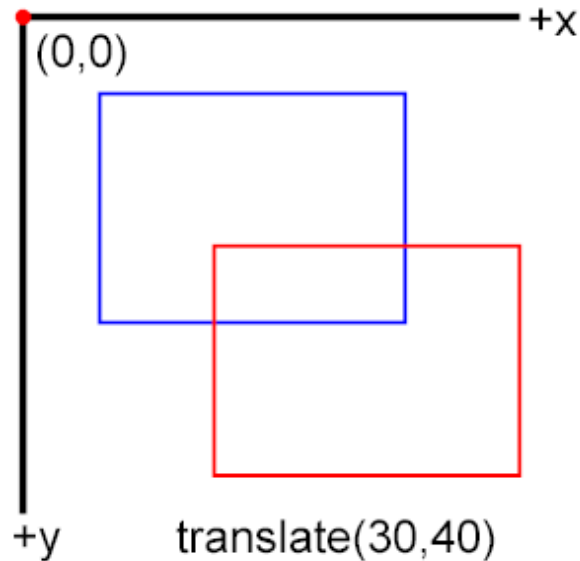
SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Beispiele
- **Neigung** (der Achsen in x- bzw. y-Richtung):



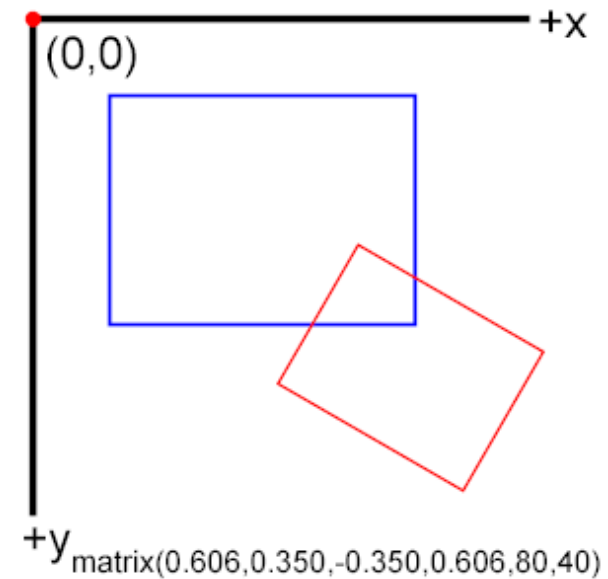
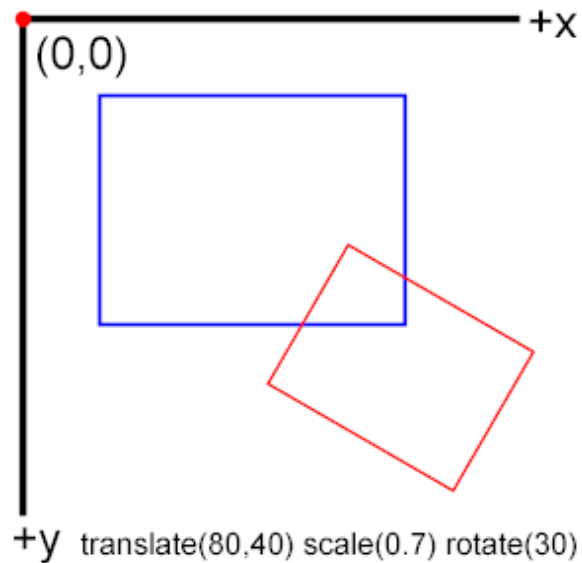
SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Beispiele
- **Verschiebung** (in x- bzw. y-Richtung):



SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Beispiele
- Kombination von Transformationen:



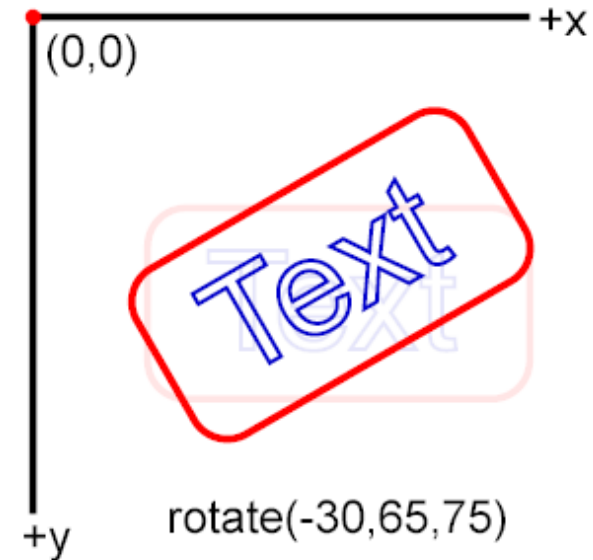
Rotieren --> Skalieren --> Verschieben



SVG im Überblick

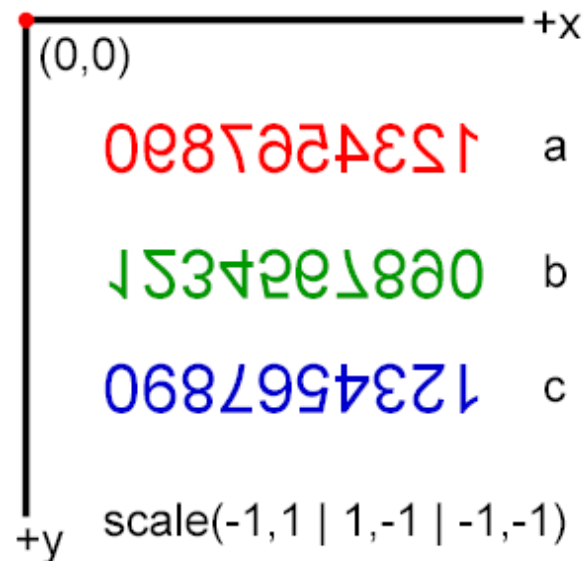
- ⇒ Koordinatensystem-Transformationen: Beispiele
- Transformation einer Gruppe von Objekten:

```
<g transform="...">  
  <rect ... />  
  <text ... />  
</g>
```



SVG im Überblick

- ⇒ Koordinatensystem-Transformationen: Beispiele
- **Spiegeln mit `scale()`** durch Setzen von -1 für x-/y-Skalierung:



Hinweis:

Spiegelung erfolgt an den Achsen, deshalb zusätzliche Verschiebung mit **`translate()`** sinnvoll.



SVG im Überblick

⇒ Animationen: Allgemeines

- Animationselemente (aus der **SMIL**-Spezifikation
SMIL = Synchronized Multimedia Integration Language):

animate	Animation skalarer XML-Attribute / CSS-Eigenschaften
animateColor	speziell zur Animation von Farbwerten (Farbübergänge) vorgesehen (akzeptiert alle animate-Attribute, auch accumulate="sum")
animateMotion	Bewegung an einem Pfad entlang
set	Animation von nichtnumerischen Werten (z. B. visibility: visible hidden oder display: block none oder für Hover-Effekte mit Farbwerten, Opazität usw.) bei dur="indefinite" oder fill="freeze" bleibt der Effekt erhalten, sonst wird dieser wieder aufgehoben



SVG im Überblick

⇒ Animationen: Allgemeines

➤ Animationselemente (SVG-Erweiterungen von SMIL):

animateTransform Modifikation von Transformationen (Drehungen, Verschiebungen, Neigungen/Verzerrungen, Skalierungen)
Attribut **type** (analog zum **transform**-Attribut für SVG-Elemente)

animateMotion Kindelement **mpath** referenziert ein vorhandenes SVG-Pfadelement als Definitionsbereich der Animation
Attribut **path** (Pfadsyntax gemäß dem **d**-Attribut des SVG-Elements **path**)
SMIL erlaubt nur eine Untermenge der SVG-Pfadsyntax



SVG im Überblick

⇒ Animationen: Allgemeines

➤ Zuweisung von Animationen:

- als Kindelement eines zu animierenden SVG-Elements

```
<element>  
  <animationelement ... />  
</element>
```

- oder per Referenz über eine dem zu animierenden Element zugewiesene ID

```
<element id="abc" ... />  
<!-- ggf. weitere Inhalte -->  
<animationelement xlink:href="#abc" ... />
```



SVG im Überblick

⇒ Animationen: Allgemeines

➤ Festlegung der zu animierenden XML-Attribute oder CSS-Eigenschaften:

attributeName="..." erhält Name des Attributes oder der Eigenschaft

attributeType="CSS | XML | auto"

CSS = "animierbare" CSS-Eigenschaft

XML = "animierbares" XML-Attribut

auto = Standardwert, sucht in der Reihenfolge CSS, XML nach Eigenschaft oder Attribut mit dem bei **attributeName** angegebenen Wert

- zu animierbaren Attributen / Eigenschaften siehe Abschnitt 19.2.5 der SVG 1.1-Spezifikation



SVG im Überblick

⇒ Animationen: Allgemeines

➤ Attribute zur Steuerung:

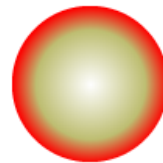
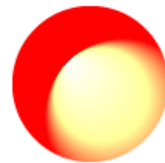
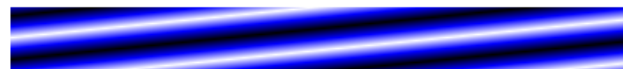
begin	Startzeitpunkt(e)
end	Endzeitpunkt
dur	Zeitdauer eines Animationsdurchlaufes
min	minimaler Zeitwert für einen Durchlauf
max	maximaler Zeitwert für einen Durchlauf
restart	Vorgaben für den Neustart einer Animation
repeatCount	Anzahl von Wiederholungen
repeatDur	Gesamtzeit für Wiederholungen der Animation
from	Startwert eines Attributes / einer Eigenschaft
to	Endwert eines Attributes/einer Eigenschaft
by	relative Änderung eines Attributes
fill	freeze / remove



SVG im Überblick

⇒ Animationen: Beispiele

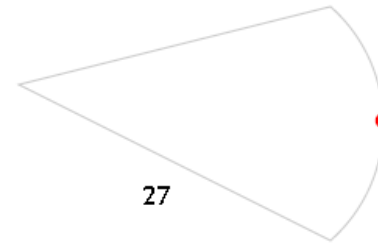
- Die folgenden Beispiele sind Bestandteil des Artikels „Mobile Vektoren“, erschienen in Internet Professionell 6/2003, online verfügbar unter:
http://www.vnunet.de/praxis/professional_computing/article20030504508.aspx



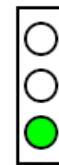
32

Hauptmenü 1	Hauptmenü 2	Hauptmenü 3
Untermenü 1.1		
Untermenü 1.2		
Untermenü 1.3		
StyleAssistant.de		
Untermenü 1.5		

34



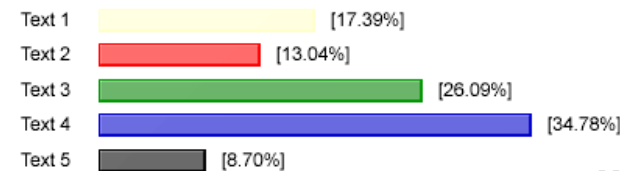
27



23



29



33

SVG im Überblick

⇒ Animationen: Beispiel 01

Der cx-Wert des Mittelpunktes eines Kreises wird 2 Sekunden nach dem Laden innerhalb von 5 Sekunden von 50 auf 300 erhöht. Es werden Möglichkeiten der **Referenzierung** von Animationselementen gezeigt. Die Animationen werden jeweils einmal ausgeführt. Danach gehen die Objekte in den Ausgangszustand zurück.

```
<circle id="k1" class="rot" cx="50" cy="40" r="20"/>
```

```
<animate attributeName="cx" attributeType="XML" begin="2s"  
  dur="5s" from="50" to="300" xlink:href="#k1"/>
```

```
<circle class="gruen" cx="50" cy="140" r="20">
```

```
  <animate attributeName="cx" attributeType="XML" begin="2s"  
    dur="5s" from="50" to="300"/>
```

```
</circle>
```



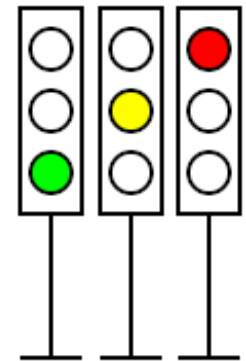
SVG im Überblick

⇒ Animationen: Beispiel 23

Simulation einer Ampel. Der grüne Kreis wird nach 6 Sekunden durch den gelben Kreis abgelöst. Nach weiteren 3 Sekunden erscheint 6 Sekunden lang der rote Kreis. 3 Sekunden später erscheint wieder gelb. Nach "gelb.end" schaltet die Ampel für 6 Sekunden auf grün und das Spiel beginnt erneut.

```
<circle id="k1" class="ampel" cx="100" cy="30" r="10"/>  
<circle id="k2" class="ampel" cx="100" cy="60" r="10"/>  
<circle id="k3" class="ampel" cx="100" cy="90" r="10"/>
```

```
<animate attributeName="fill" attributeType="CSS"  
  begin="0s;gelb.end" xlink:href="#k3"  
  dur="0.01s" from="none" to="#0F0" fill="freeze"/>  
<!-- weitere animate-Elemente ... -->  
<animate attributeName="fill" attributeType="CSS"  
  begin="prev.end+3s" xlink:href="#k2" id="gelb"  
  dur="0.01s" from="#FF0" to="none" fill="freeze"/>
```



SVG im Überblick

⇒ Animationen: Beispiel 29

Animation eines Pfades (der rote Mund des Smileys). Das **Pfad-Attribut d** wird vom Ausgangswert ("Mund nach unten") zum Endwert ("Mund nach oben") animiert.

```
<!-- Pfad fuer den roten Mund mit Animation -->
<path d="M 85,130 A 5,5 0 1,1 115,130"
      style="fill: #FF0; stroke: #F00; stroke-width: 2px">
  <animate attributeName="d" attributeType="XML"
    begin="0s" dur="10s" fill="freeze"
    from="M 85,130 A 5,5 0 1,1 115,130"
    to="M 85,110 A 5,5 0 1,0 115,110"/>
</path>
```

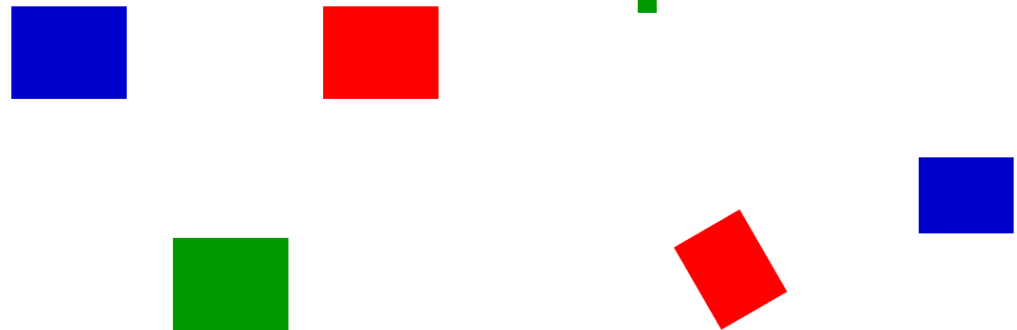


SVG im Überblick

⇒ Animationen: Beispiel 30

Es werden drei Rechtecke durch Änderung ihrer Transformationsparameter animiert. Das Element **animateTransform** erhält über das Attribut **type** die Art der Transformation zugewiesen (mögliche Werte: rotate, scale, skewX, skewY, translate).

```
<!-- Code fuer das gruene Rechteck -->  
<rect class="gruen" x="170" y="280" width="100"  
      height="80">  
  <animateTransform attributeName="transform"  
    attributeType="XML" type="scale"  
    from="1" to="0.2" begin="2s" dur="10s"  
    fill="freeze"/>  
</rect>
```



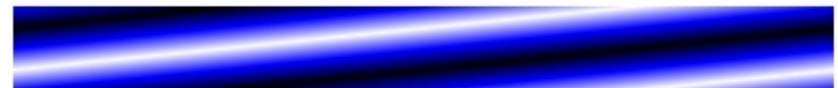
SVG im Überblick

⇒ Animationen: Beispiel 32/1

Es werden die Parameter von linearen Gradienten animiert (x1,y1,x2,y2).

```
<linearGradient id="animlingra1" x1="0" y1="0" x2="1" y2="1"
  spreadMethod="reflect">
  <stop offset="0%" style="stop-color: #000"/>
  <stop offset="50%" style="stop-color: #00F"/>
  <stop offset="100%" style="stop-color: #FFF"/>

  <animate attributeName="x1" attributeType="XML" begin="0s"
    dur="5s" from="0" to="1" repeatCount="indefinite"/>
  <!-- weitere animate-Elemente fuer x2 und y1 ... -->
  <animate attributeName="y2" attributeType="XML" begin="0s"
    dur="5s" from="1" to="0" repeatCount="indefinite"/>
</linearGradient>
```

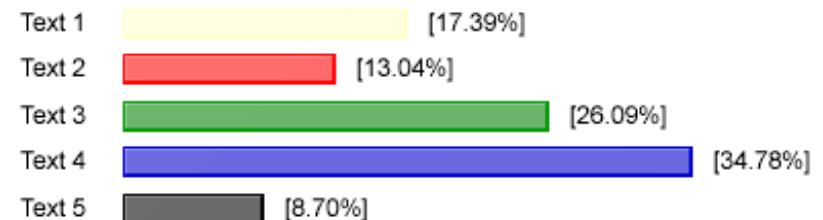


SVG im Überblick

⇒ Animationen: Beispiel 33/1

Die Grafik zeigt ein **Balkendiagramm**, wobei die Balken über eine Zeit von 10 Sekunden ihre maximale Breite erreichen. Nach insgesamt 12 Sekunden werden die Beschreibungstexte rechts neben den Balken eingeblendet. Animiert werden die XML-Eigenschaft **width** sowie die CSS-Eigenschaft **visibility**.

```
<!-- blaues Rechteck mit Animation -->  
<rect x="100" y="147" width="0" height="15" style="fill:  
#00C; filter: url(#flt)">  
  <animate attributeName="width" attributeType="XML"  
    begin="0s" dur="10s" fill="freeze"  
    from="0" to="278"/>  
</rect>
```

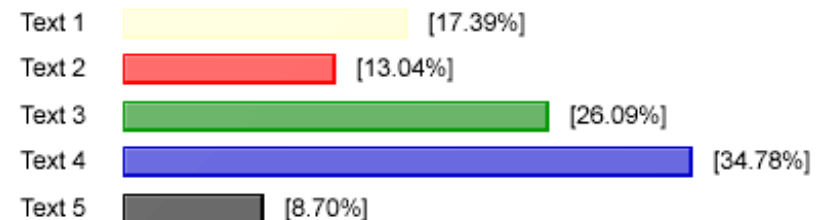


SVG im Überblick

⇒ Animationen: Beispiel 33/2

Die Grafik zeigt ein **Balkendiagramm**, wobei die Balken über eine Zeit von 10 Sekunden ihre maximale Breite erreichen. Nach insgesamt 12 Sekunden werden die Beschreibungstexte rechts neben den Balken eingeblendet. Animiert werden die XML-Eigenschaft **width** sowie die CSS-Eigenschaft **visibility**.

```
<!-- Text nach dem blauen Rechteck mit Animation -->  
<text x="388" y="158" style="font-size: 12px; text-anchor:  
  right; visibility: hidden">[34.78%]  
  <animate attributeName="visibility" attributeType="CSS"  
    begin="10s" dur="2s" fill="freeze"  
    from="hidden" to="visible" calcMode="discrete"/>  
</text>
```



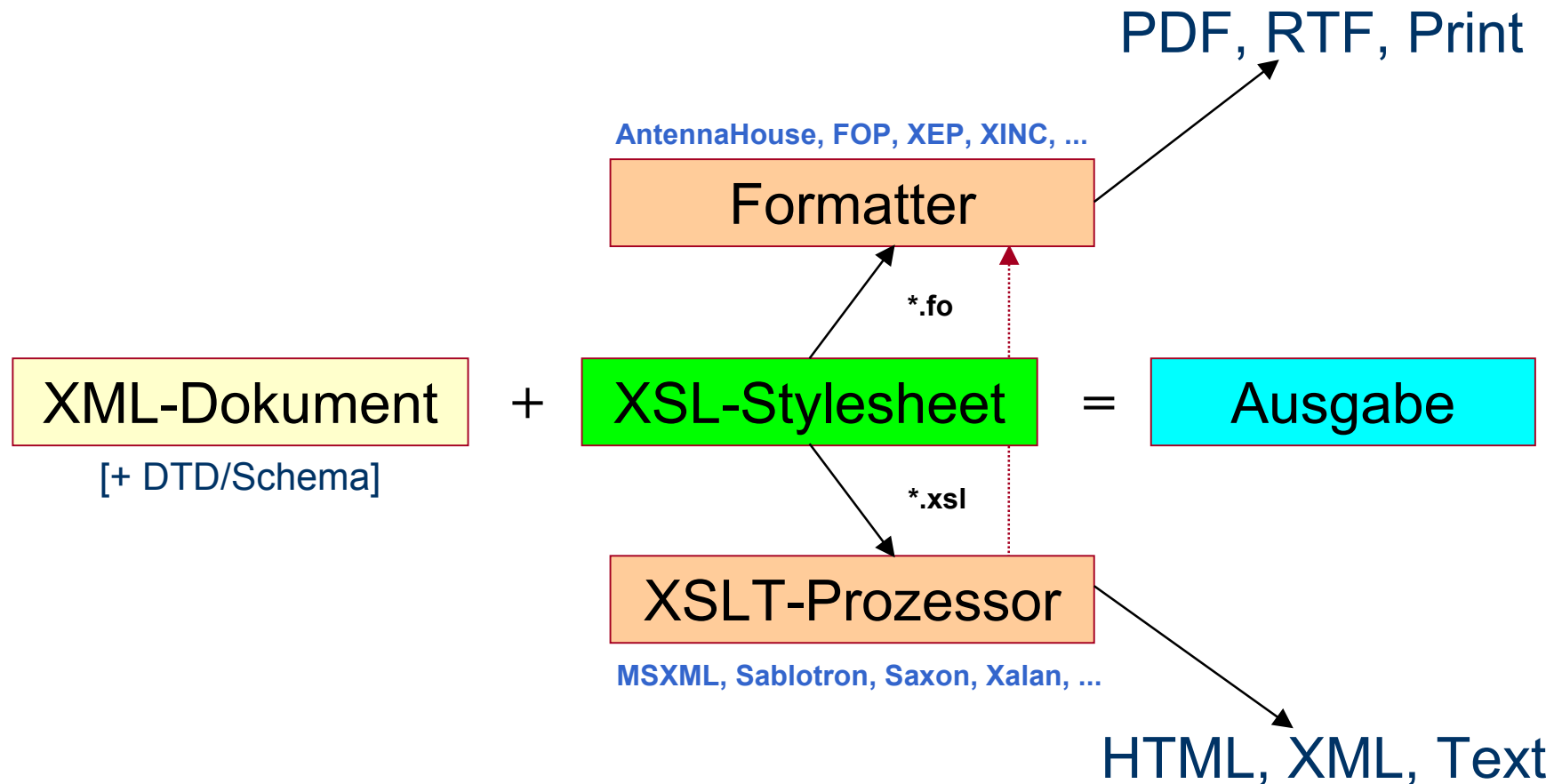
SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- XSLT dient zur Transformation von XML-Dokumenten (bzw. Inhalten), wobei die Ausgaben als HTML, XML oder Text erfolgen können.
- SVG ist ein XML-basiertes Format, sodass XSL-Transformationen zur Produktion von SVG-Dokumenten grundsätzlich möglich und in der Praxis auch sinnvoll sind.
- Zur praktischen Durchführung werden neben den Ausgangs-XML-Dokumenten geeignete XSL-Stylesheets sowie XSLT-Prozessoren benötigt.
- XSLT wird im Bereich des Single-Source- / Cross-Media-Publishings verwendet.



SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Prinzip des Single-Source- / Cross-Media-Publishings mit XSL (XSLT bzw. XSL-FO):



SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Beispiel zur Ausgabe von Rechtecken (XML-Daten / [daten.xml](#)):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rechtecke>
  <rechteck>
    <linksoben_x>50</linksoben_x>
    <linksoben_y>120</linksoben_y>
    <breite>180</breite>
    <hoehe>75</hoehe>
    <farbe>green</farbe>
  </rechteck>
  <!-- weitere Rechteck-Definitionen -->
</rechtecke>
```



SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Beispiel zur Ausgabe von Rechtecken (XSL-Stylesheet / [vorlage.xsl](#)):

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="xml" doctype-public="-//W3C//DTD SVG 1.1//EN"
  doctype-system="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"
  encoding="ISO-8859-1" version="1.0"
  media-type="image/svg+xml" indent="yes"/>

<xsl:template match="/">

<svg xmlns="http://www.w3.org/2000/svg">

<xsl:for-each select="rechtecke/rechteck">

  <rect x="{linksoben_x}" y="{linksoben_y}" width="{breite}" height="{hoehe}"
    fill="{farbe}"/>

</xsl:for-each>

</svg>

</xsl:template>

</xsl:stylesheet>
```



SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Beispiel zur Ausgabe von Rechtecken (SVG-Ergebnis / [ergebnis.svg](#)):

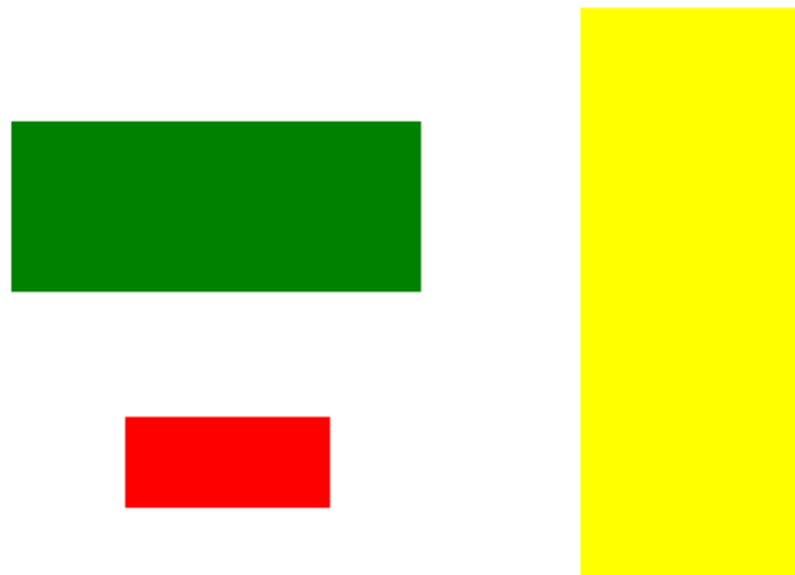
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="50" y="120" width="180" height="75" fill="green"/>
  <rect x="300" y="70" width="100" height="250" fill="yellow"/>
  <rect x="100" y="250" width="90" height="40" fill="red"/>
</svg>
```

Umsetzung mit XMLStarlet (<http://xmlstar.sourceforge.net>)
`xml tr vorlage.xsl daten.xml > ergebnis.svg`



SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Beispiel zur Ausgabe von Rechtecken (SVG-Ergebnis / [grafisch](#)):



SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Beispiel Einwohner-Daten des Statistischen Bundesamtes (destatis.de)

- Rohdaten:

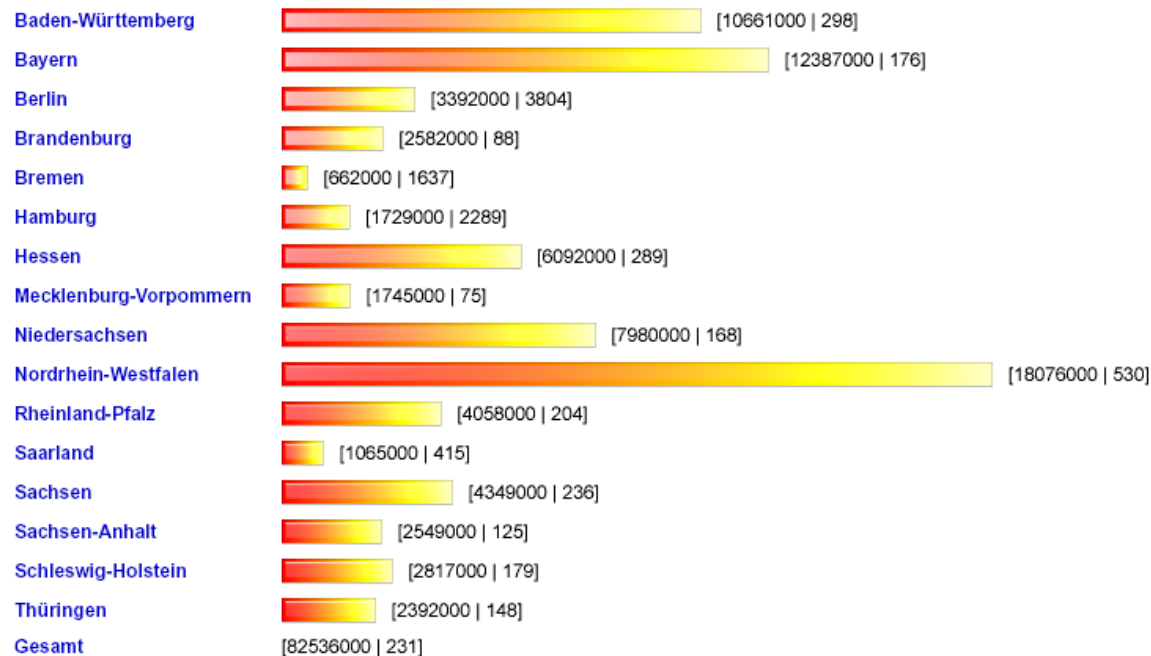
Fläche und Bevölkerung					
Land	Fläche Km ²	Bevölkerung *			Einwohner je km ² *
		insgesamt	männlich	weiblich	
			1 000		Anzahl
Jahr/Monat/Stichtag	31.12.2002				
Baden-Württemberg	35 751,64	10 661	5 230	5 431	298
Bayern	70 549,32	12 387	6 061	6 327	176
Berlin	891,75	3 392	1 651	1 741	3 804
Brandenburg	29 476,67	2 582	1 276	1 306	88
Bremen	404,28	662	320	342	1 638
Hamburg	755,26	1 729	839	890	2 289
Hessen	21 114,88	6 092	2 985	3 107	288
Mecklenburg-Vorpommern	23 173,46	1 745	864	881	75
Niedersachsen	47 617,97	7 980	3 907	4 074	168
Nordrhein-Westfalen	34 082,76	18 076	8 799	9 278	530
Rheinland-Pfalz	19 846,91	4 058	1 991	2 066	204
Saarland	2 568,53	1 065	517	548	415
Sachsen	18 413,29	4 349	2 112	2 237	236
Sachsen-Anhalt	20 444,72	2 549	1 242	1 307	125
Schleswig-Holstein	15 762,90	2 817	1 376	1 440	179
Thüringen	16 172,21	2 392	1 174	1 218	148
Deutschland	357 026,55	82 537	40 345	42 192	231
* Ergebnisse der Bevölkerungsfortschreibung.					
Aktualisiert am 01. Oktober 2003					

SVG im Überblick

- ⇒ SVG aus XML-Daten mittels XSLT erzeugen
- Beispiel Einwohner-Daten des Statistischen Bundesamtes (destatis.de)
 - SVG-Ergebnis (einwohner.svg):

Einwohnerzahlen am Stichtag 31.12.2002

Legende: [Einwohner gesamt | Einwohner pro km²]



Diese Grafik wurde aus XML-Daten mittels XSLT erzeugt.

Datenquelle: Statistisches Bundesamt Deutschland

SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- SVG-Dokumente lassen sich mit serverseitigen Sprachen/Technologien aufbauen und ausgeben.
- Verwendung dynamischer Textfunktionen von ASP, JSP, Perl, PHP usw.
- Daten können entsprechend als SVG-Inhalte aufbereitet werden.
- Wesentlich ist das Sendes des korrekten Inhaltstyps (MIME-Type)
`image/svg+xml`
- Bei Verwendung des Apache-Webserver in `httpd.conf` oder `.htaccess` konfigurieren (in `mime.types` ohne `AddType`):

```
AddType image/svg+xml .svg .svgz
```

- Alternativ Informationen im Header mitsenden (für PHP):
`header("Content-Type: image/svg+xml");`



SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Umsetzung des Einwohner-Beispiels mit PHP

- Daten in Excel-Tabelle
und Ausgabe als CSV-Datei
einwohner.dat

...

Sachsen-Anhalt;2549000;20444.72

...

	A	B	C
1	Baden-Württemberg	10661000	35751.64
2	Bayern	12387000	70549.32
3	Berlin	3392000	891.75
4	Brandenburg	2582000	29476.67
5	Bremen	662000	404.28
6	Hamburg	1729000	755.26
7	Hessen	6092000	21114.88
8	Mecklenburg-Vorpommern	1745000	23172.46
9	Niedersachsen	7980000	47617.97
10	Nordrhein-Westfalen	18076000	34082.76
11	Rheinland-Pfalz	4058000	19846.91
12	Saarland	1065000	2568.53
13	Sachsen	4349000	18413.29
14	Sachsen-Anhalt	2549000	20444.72
15	Schleswig-Holstein	2817000	15762.90
16	Thüringen	2392000	16172.21



SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Umsetzung des Einwohner-Beispiels mit PHP ([einwohner.php](#)):

```
<?php
// Dokumentenkopf ausgeben (XML-Prolog, Definitionen für Filter, Gradienten usw.) ...
// Daten einlesen, aufbereiten und ausgeben:
$daten=file("einwohner.dat");
$anzahl=count($daten);
$ew_max=1;

for($i=0;$i<$anzahl;$i++)
{
    $zeile=explode(";",trim($daten[$i]));
    $bl[$i]=$zeile[0]; // Bundesland
    $ew[$i]=$zeile[1]; // Einwohnerzahl
    $fl[$i]=$zeile[2]; // Fläche
    if($ew[$i]>$ew_max)$ew_max=$ew[$i]; // max. Einwohnerzahl ermitteln
}
...
```



SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Umsetzung des Einwohner-Beispiels mit PHP ([einwohner.php](#)):

```
// ... Fortsetzung der SVG-Ausgabe:
```

```
$max_breite=700;
```

```
$faktor=$max_breite/$ew_max;
```

```
for($i=0;$i<$anzahl;$i++)
```

```
{
```

```
    print "    <text x=\"20\" y=\"".intval($i*25+80).\"\">".$bl[$i].\"</text>\n\";
```

```
    print "    <text class=\"klein\" x=\"20\" y=\"".intval($i*25+90).\"\">[\".$ew[$i].\" | \"  
        .round($ew[$i]/$fl[$i]).\"]</text>\n\";
```

```
    print "    <rect x=\"180\" y=\"".intval($i*25+75).\"\" height=\"15\" „  
        width=\"".round($ew[$i]*$faktor).\"\" filter=\"url(#schatten)\"/>\n\";
```

```
}
```

```
?>
```

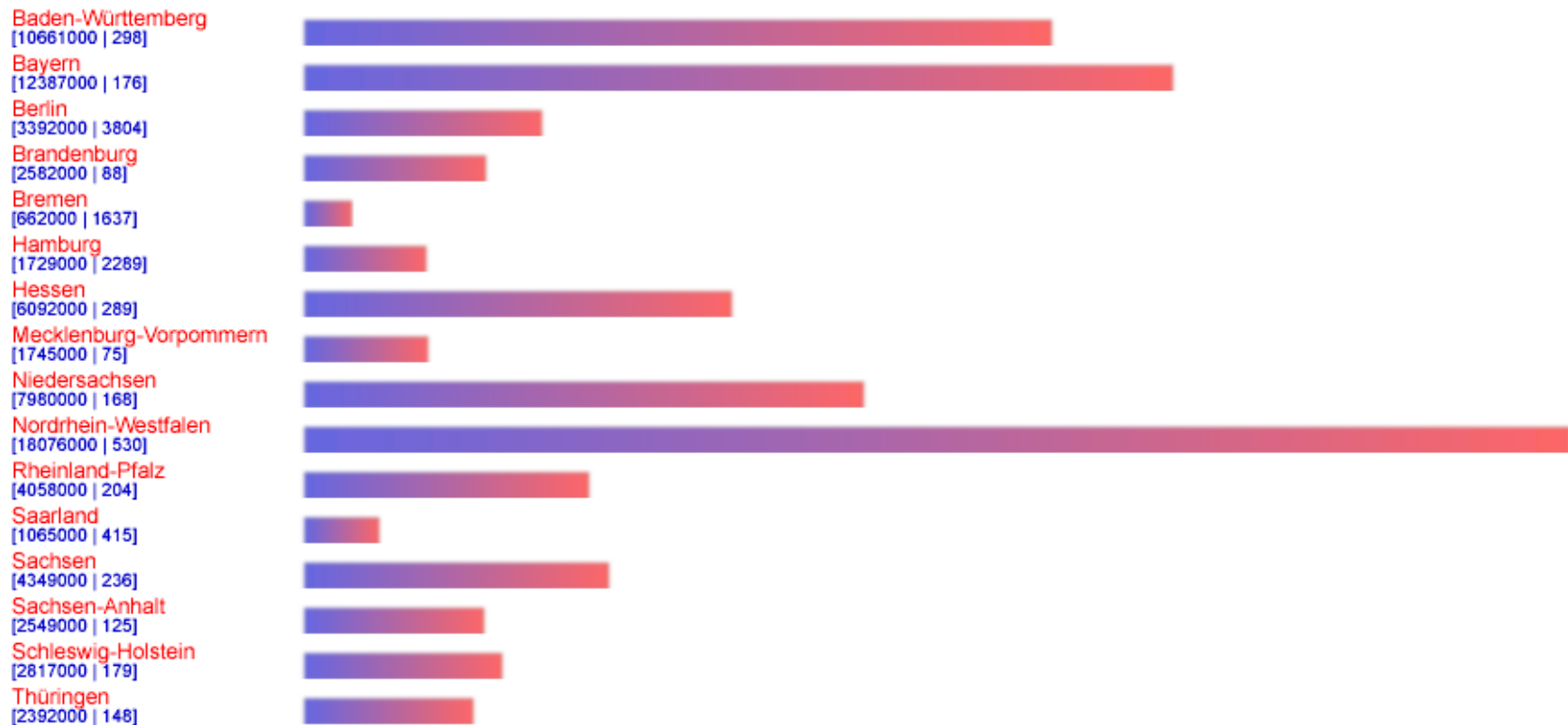


SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Umsetzung des Einwohner-Beispiels mit PHP - grafisches Ergebnis:

Einwohnerzahlen der Bundesländer (31.12.2002)

Legende: [Einwohner gesamt | Einwohner pro km²]



Datenquelle: Statistisches Bundesamt Deutschland

SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Weitere PHP-Anwendung mit SVG-Ausgabe - SVG::PHP:

SVG::PHP – Dateneingabe

Vorgaben

Titel:

Welches Redaktionstool verwenden Sie?

☒ Kreisdiagramm cx: cy: r:

☐ Balkendiagramm x: y: Balkenhöhe: max. Balkenbreite (=100%):

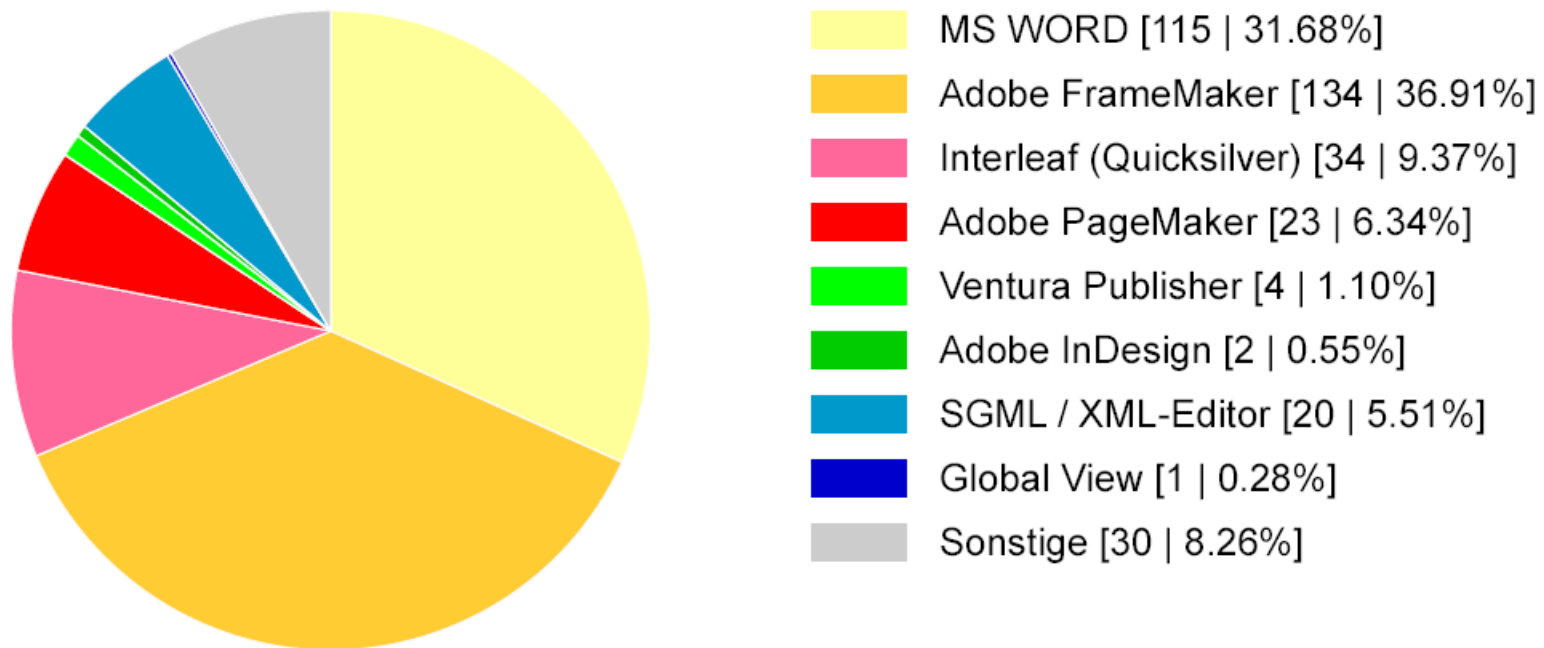
Daten

Nr.	Wert	Beschreibung	Farbe
01	<input type="text" value="115"/>	<input type="text" value="MS WORD"/>	<input type="text" value="#FFFF99"/>
02	<input type="text" value="134"/>	<input type="text" value="Adobe FrameMaker"/>	<input type="text" value="#FFCC33"/>
03	<input type="text" value="34"/>	<input type="text" value="Interleaf (Quicksilver)"/>	<input type="text" value="#FF6699"/>
04	<input type="text" value="23"/>	<input type="text" value="Adobe PageMaker"/>	<input type="text" value="#FF0000"/>
05	<input type="text" value="4"/>	<input type="text" value="Ventura Publisher"/>	<input type="text" value="#00FF00"/>
06	<input type="text" value="2"/>	<input type="text" value="Adobe InDesign"/>	<input type="text" value="#00CC00"/>
07	<input type="text" value="20"/>	<input type="text" value="SGML / XML-Editor"/>	<input type="text" value="#0099CC"/>
08	<input type="text" value="1"/>	<input type="text" value="Global View"/>	<input type="text" value="#0000CC"/>
09	<input type="text" value="30"/>	<input type="text" value="Sonstige"/>	<input type="text" value="#CCCCCC"/>

SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Weitere PHP-Anwendung mit SVG-Ausgabe - SVG::PHP:

Welches Redaktionstool verwenden Sie?



Datenquelle: tekomp e.V. 2003

Online verfügbar unter: <http://www.datenverdrahten.de/svgphp/>

SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Anwendung MSpec::SVG - Massenspektren darstellen
 - MySQL-Datenbank mit den Rohdaten (50000 Verbindungen):

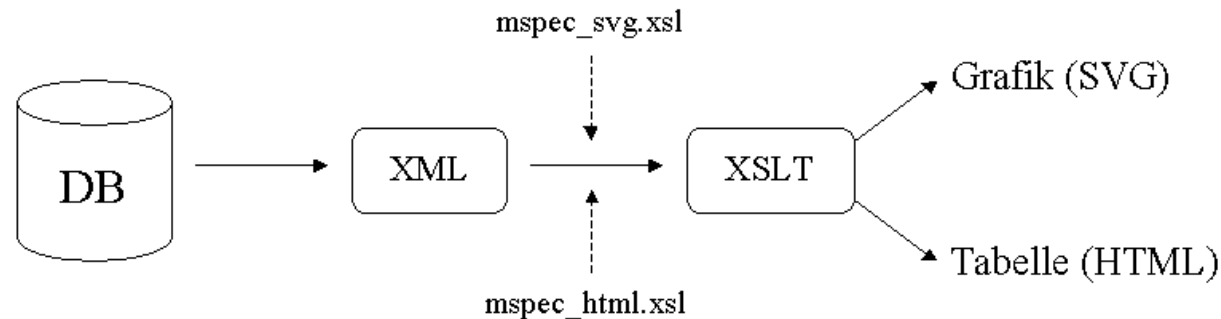
id	formel	verbindung	daten
1	C4H8	2-BUTENE, (E)-	1 16 2 0 12 0 13 0 14 8 15 36 16 0 24 0 25 8 26 11...
2	C4H8	2-BUTENE	2 12 12 4 13 8 14 20 15 76 25 20 26 160 27 364 28 ...
3	C4H8	2-BUTENE, (Z)-	1 16 2 0 12 0 13 0 14 8 15 36 16 0 19 0 24 0 25 8 ...
4	C4H8	1-PROPENE, 2-METHYL-	2 8 12 8 13 12 14 32 15 88 16 0 25 8 26 64 27 216 ...
5	C4H8	1-BUTENE	2 8 12 8 13 12 14 32 15 80 16 0 25 16 26 128 27 33...
6	C4H8	CYCLOBUTANE	24 4 26 128 27 276 28 796 29 100 30 4 34 0 36 0 37...
7	C6H5CLO	PHENOL, 2-CHLORO-	26 28 27 20 29 28 31 28 36 8 37 68 38 140 39 168 4...
8	C6H5CLO	PHENOL, 3-CHLORO-	18 4 26 8 27 12 28 52 29 12 31 16 32 4 35 4 36 20 ...
9	C6H5CLO	PHENOL, 4-CHLORO-	25 8 26 28 27 28 29 40 31 28 35 8 36 20 37 68 38 8...



SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Anwendung MSpec::SVG - Massenspektren darstellen

- Prinzip der Verarbeitung:



- Auswahlmenü:

The screenshot shows the user interface of the MSpec::SVG application. At the top, there is a search bar with the label "Summenformel:" and a text input field containing "C4H8". To the right of the input field is a blue button labeled "Suchen". A tooltip next to the button indicates that the search can be performed using formulas like "C4H8" or "C6H5ClO". Below the search bar is a yellow button labeled "Suche starten".

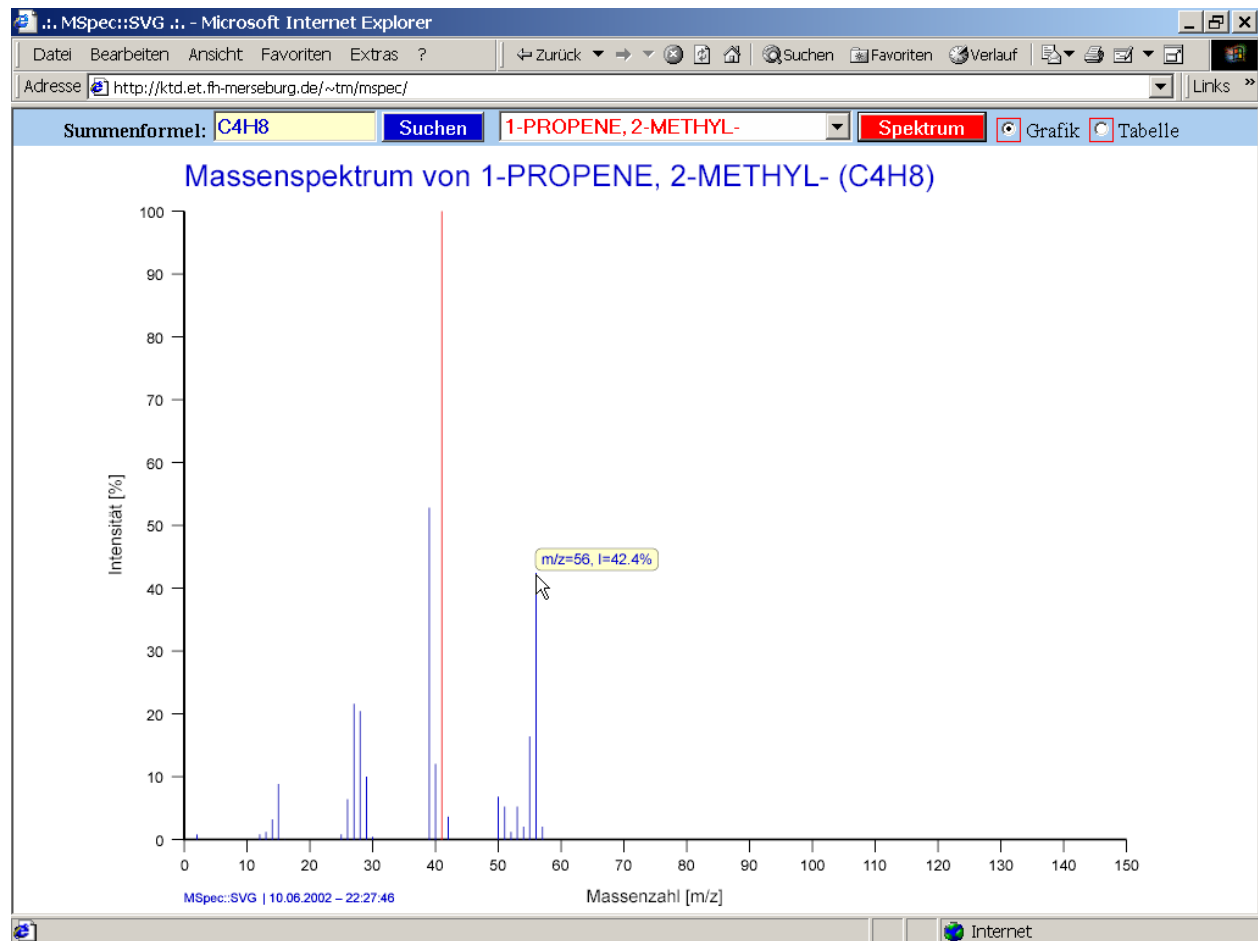
Below the search bar is a dropdown menu titled "=== Verbindung auswählen (6) ===". The menu is open, showing a list of chemical compounds: "2-BUTENE, (E)-", "2-BUTENE", "2-BUTENE, (Z)-", "1-PROPENE, 2-METHYL-", "1-BUTENE", and "CYCLOBUTANE". The "1-PROPENE, 2-METHYL-" option is currently selected and highlighted in blue.

To the right of the dropdown menu are three buttons: "Spektrum" (highlighted in red), "Grafik" (with a radio button), and "Tabelle" (with a radio button).

SVG im Überblick

- ⇒ SVG auf dem Webserver erzeugen
- Anwendung MSpec::SVG - Massenspektren darstellen

- Ansicht im Browser:



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript
(Grundlagen zur Entwicklung dynamischer, interaktiver SVG-Anwendungen)

- **Basis:**

SVG statisch, Kenntnisse zu Elementen, Attributen, CSS-Eigenschaften, XML-Dokumentstruktur.

- **Programmierung:**

clientseitige Skriptsprachen (JavaScript, ECMAScript), Event-Handling, Objektmodell (W3C-/SVG-DOM).

- **Verwandte Konzepte:**

DHTML, ActionScript (Flash/SWF).



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript

- Programmcode wird im Bereich von <defs>...</defs> abgelegt

intern (als CDATA-Abschnitt, da reservierte Zeichen wie `<` & `&` vorkommen können):

```
<script type="text/javascript">
```

```
<![CDATA[
```

```
    /* JS-Code */
```

```
]]>
```

```
</script>
```

extern (als einfache Textdatei oder auch gzip-komprimiert):

```
<script xlink:href="datei.js" type="text/javascript"/>
```

```
<script xlink:href="datei.js.gz" type="text/javascript"/>
```



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript

- Funktionen kapseln Codeblöcke für den späteren Aufruf:

```
var ...; // globale Variablen

function Funktionsname1(arg)
{
    var ...; // lokale Variablen
    /* weiterer Code der Funktion */
}

function Funktionsname2(arg1,arg2,...)
{
    var ...; // lokale Variablen
    /* weiterer Code der Funktion */
}
```

- Funktionen werden ereignisgesteuert aufgerufen
- Ablage erstellter Funktionen in Bibliotheken sinnvoll (*.js)



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript

- Die SVG verarbeitende Instanz (Browser, Plug-in) registriert bei der Interaktion ständig so genannte Ereignisse (Events):

Beispiel: **click**-Event beim Anklicken von Objekten

- Zur Reaktion auf diese Events dienen Event-Handler:

Beispiel: **onclick**-Event-Handler (Prefix on)

- Event-Handler werden als Element-Attribute verwendet:

Beispiel (Aufruf einer JS-Funktion):

```
<circle cx="50" cy="50" r="20" style="fill: #F00"  
  onclick="EineFunktion(evt)"/>
```



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript

- Einfaches SVG-Beispiel → Code

Hex



```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="340" height="100" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">

  <title>SVG-Beispiel</title>
  <desc>Es werden je ein Text, Rechteck, Kreis und Polygon definiert.</desc>

  <text id="tx" x="10" y="55" style="font-size: 40px">Hex</text>

  <rect id="re" x="135" y="20" width="40" height="40" style="fill: #00C"/>

  <circle id="kr" cx="230" cy="40" r="20" style="fill: #F00"/>

  <polygon id="po" points="280,60 300,20 320,60" style="fill: #090"/>

</svg>
```



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript

- Beispiel erweitert mit SVG-DOM-Zugriff

SVG-Objekte

```
<text id="tx" x="10" y="55" style="font-size: 40px">Hex</text>
<rect id="re" x="135" y="20" width="40" height="40" style="fill: #00C"/>
<circle id="kr" cx="230" cy="40" r="20" style="fill: #F00"
  onclick="ChangeColor(evt,'re','#F90')"/>
<polygon id="po" points="280,60 300,20 320,60" style="fill: #090"/>
```

JavaScript-Funktion

```
function ChangeColor(click_evt,objid,col)
{
  var svgdoc,obj,txt;

  svgdoc=click_evt.target.ownerDocument;
  obj=svgdoc.getElementById(objid);
  obj.style.setProperty("fill",col);

  txt=svgdoc.getElementById("tx");
  txt.firstChild.nodeValue=col;
}
```

Hex   

Hex   


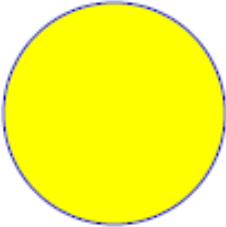
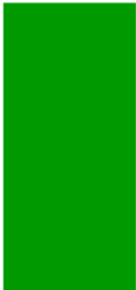

#F90   



SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript - Beispiele:
SVG-DOM-Scripting kompakt

SVG-DOM-Scripting kompakt16:04:29



Das ist der erste Testtext.
Das ist der zweite Testtext.
Das ist der dritte Testtext.

Koordinaten (an/aus):

Link mit Tooltip-Anzeige

Anzahl Rechtecke neue Kreis-Farbe (rot) Rechtecke: fill-opacity: 0.4

Radius des Kreises neuer Kreis-Radius neue Rechteck-Höhe

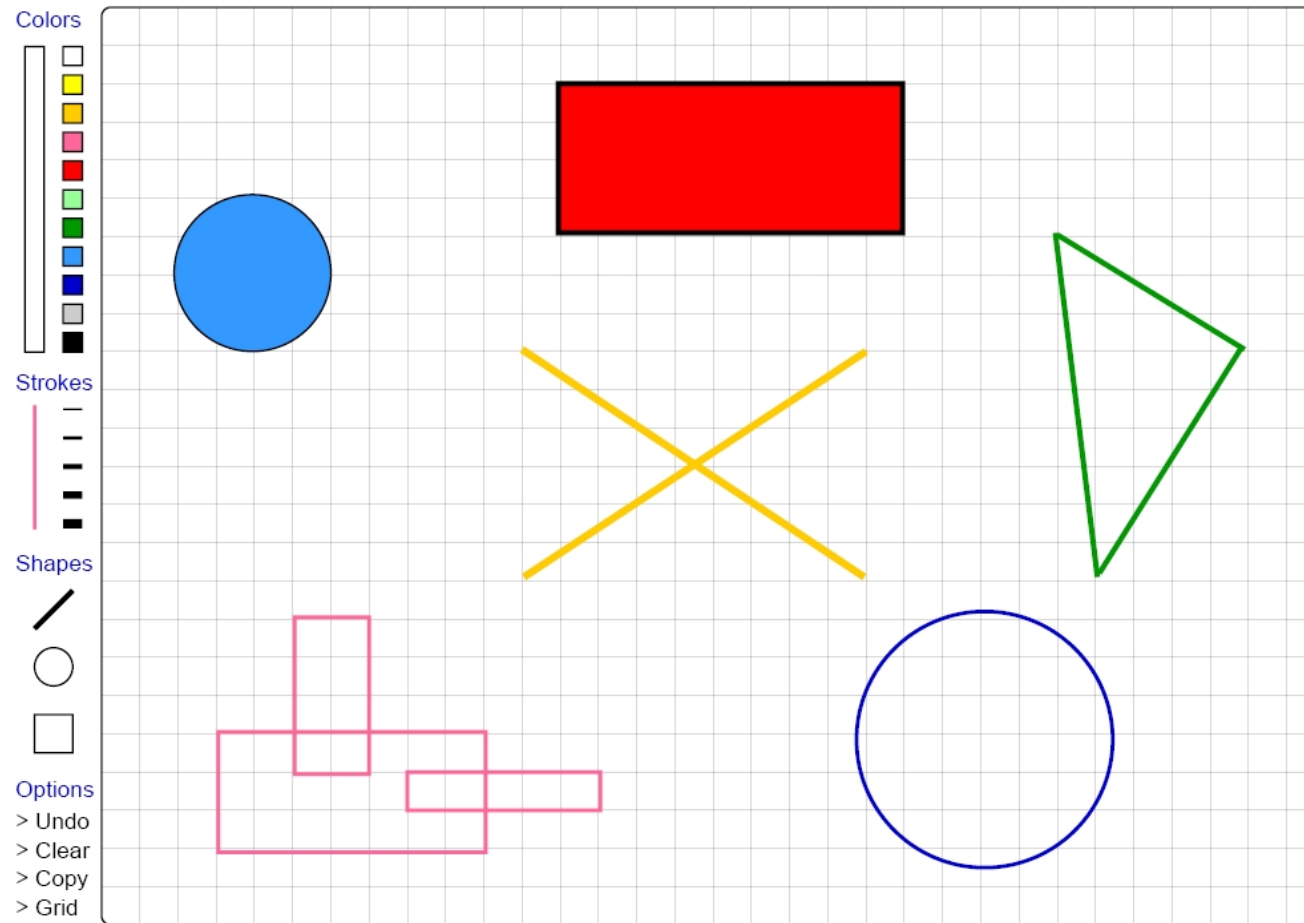
Wert von Testtext 2 Testtext 2 ersetzen Länge von Texttext 2

neues Element (line) neues tspan mit Textknoten tspan-Elemente entfernen

SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript - Beispiele: Mini-Zeichenprogramm

Grafikobjekte zeichnen (mit Wahl von Farbe und Strichstärke)



<http://svglibc.datenverdrahten.de/?doc=drawshapes>

SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript - Beispiele:
Periodensystem mit Online-Datenabfrage

Periodensystem der Elemente mit Online-Datenabfrage

Periode	Hauptgruppe I II		Nebengruppe										Hauptgruppe III IV V VI VII VIII							
	I	II	III	IV	V	VI	VII	VIII	I	II			III	IV	V	VI	VII	VIII		
1	H																		He	
2	Li	Be											B	C	N	O	F	Ne		
3	Na	Mg											Al	Si	P	S	Cl	Ar		
4	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr		
5	Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe		
6	Cs	Ba	La	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Rn		
7	Fr	Ra	Ac	Ku	Ha	Sg	Bh	Hs	Mt	Uun	Uuu	Uub								

Lanthanoide	Ce	Pr	Nd	Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb	Lu
Actinoide	Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No	Lr

Zuordnung der Farben:

- > Nichtmetalle
- > Halbmetalle
- > Hauptgruppen-Metalle
- > Nebengruppen-Metalle
- > Aktuelles Element

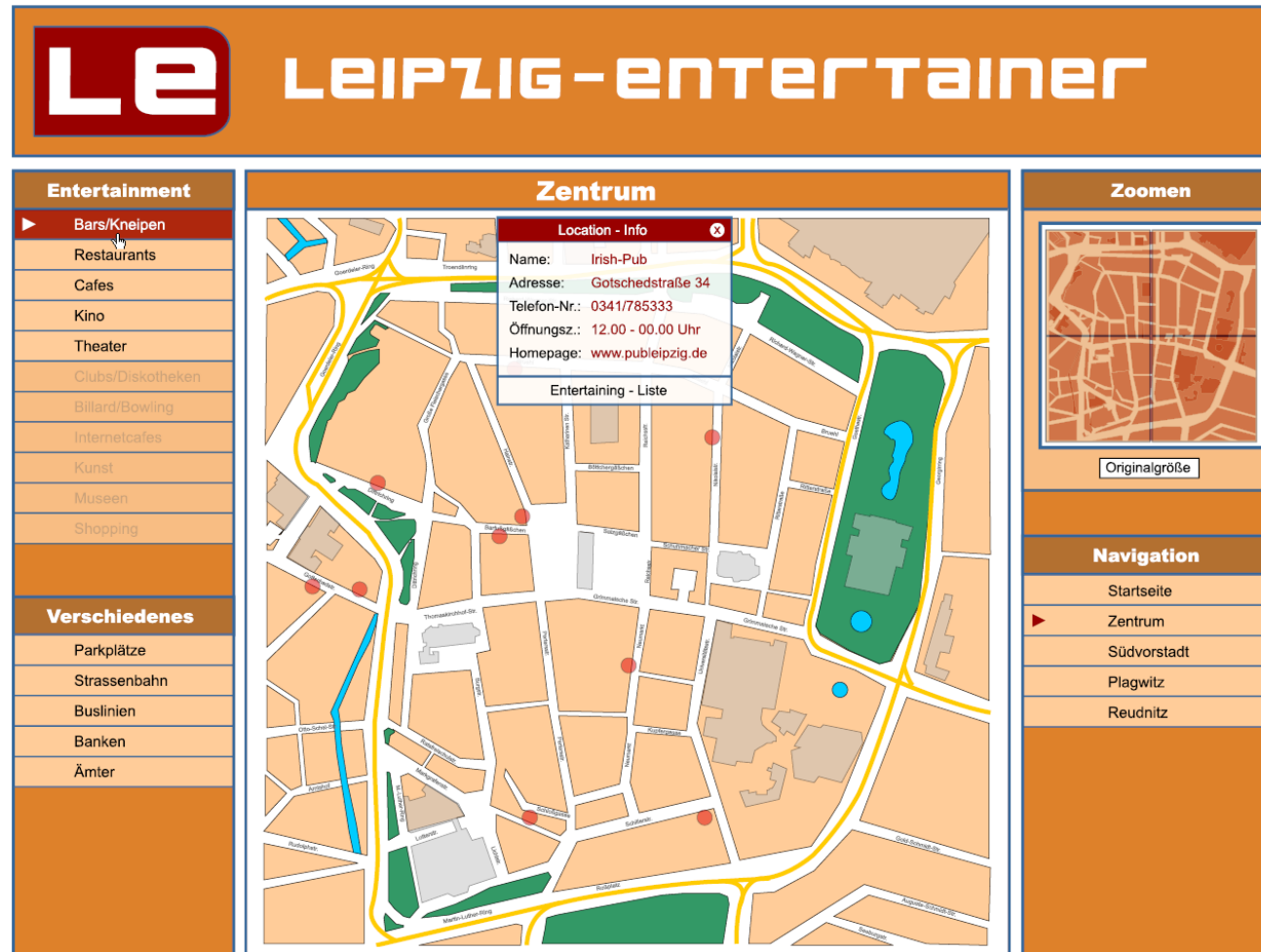
PHP-Script zur Datenabfrage

Symbol:	Ir	Density:	22650 kg / m ³
ElementName:	Iridium	MeltingPoint:	2683.000000 K
AtomicNumber:	77	BoilingPoint:	4800.000000 K
AtomicWeight:	192.220758 u	ElectroNegativity:	1.550000
AtomicRadius:	1.260000 Å	IonisationPotential:	8.680000 eV

<http://www.et.fh-merseburg.de/person/meinike/ptablesvg/>

SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript - Beispiele:
Leipzig-Entertainer (Diplomarbeit von Maik Boche 2004/05)



http://www.about-svg.de/index_LE.htm

SVG im Überblick

⇒ SVG-Aktionsprogrammierung mit JavaScript - Beispiele:
SVG – Learning By Coding (Beispielsammlung)

Adresse <http://svglbc.datenverdrahten.de/>

<Auswahl/>

- a.svg
- accessKey.svg
- ampel.svg
- animate.svg
- animateClock.svg
- animateColor.svg
- animateMotion.svg
- animateTransform.svg
- attr_node_methods.svg
- attributes.svg
- audio.svg
- barchart.svg
- baseline-shift.svg
- baseVal_animVal.svg
- begin_endElement.svg
- bezier-kurven.svg
- bezier-polynome.svg
- bildlinkfolge.svg
- browserEval.svg
- circle.svg
- clipPath.svg
- cloneNode.svg
- color_currentColor.svg
- color-profile.svg
- contextMenu.svg
- create_image.svg
- createEvent.svg
- create-methods.svg

SVG – Learning By Coding

[: XML-basierte Vektorgrafiken – Code und Informationen :]



Objekte und Effekte in SVG (Der rote Kreis, die Erklärungstexte und die Textlinks sind mit JavaScript-Funktionen verknüpft.)

- Rechteck
- Kreis
- Ellipse
- Polygon
- Polylinie
- Linie
- Plat
- linearer Gradient
- radialer Gradient
- Opazität (Durchlässigkeit)
- Spezialfilter
- Muster
- Gruppe+Transformation
- externes Bild
- Animation
- Textlink
- normaler Fließtext

© by Dr. Thomas Meinike 2002

TM/DeSVGDemo.svg

StyleAssistant.de 2002

[: **Don't grab this site! Download all the stuff!** (17.1 MB ZIP) :]

[: SVGT-Beispiel für Handys: <http://svgmob.datenverdrahten.de/> :]

[: SVGT-Anwendung mit Wetterinformationen [WEATHER 4 NOW](#) :]

[: SVG-Lehrmaterial zu [Vorlesungen und Übungen](#) (6.6 MB ZIP) :]

[: Tutorial: [Einsatz von SVG-Filtern unter Adobe Illustrator CS®](#) :]


SVG im Überblick

- ⇒ Mobile SVG-Anwendungen
- Entwicklung der Beispielanwendung WEATHER 4 NOW - Demonstration:

SVG - Learning By Coding :: WEATHER 4 NOW - Microsoft Internet Explorer

Adresse: http://svgmob.datenverdrahten.de/w4n/w4n_demo.php

SVGT-Anwendung – WEATHER 4 NOW

Auswahl [S(tadt)L(and)]	mit Java-Applet TinyLine SVG Player for Web	mit Plug-in Adobe SVG Viewer	Handy-Ansicht SIEMENS S65
<input type="radio"/> Handyformat <input checked="" type="radio"/> Zoomfaktor 2 <input type="radio"/> Amsterdam / Netherlands [AN] <input type="radio"/> Antalya / Turkey [AT] <input type="radio"/> Bruxelles / Belgium [BB] <input type="radio"/> Budapest / Hungary [BH] <input type="radio"/> Bombay / India [BI] <input type="radio"/> Berlin / Germany [BG] <input type="radio"/> Cairo / Egypt [CE] <input type="radio"/> Dresden / Germany [DG] <input type="radio"/> Dublin / Ireland [DI] <input type="radio"/> Helsinki / Finland [HF] <input type="radio"/> Hamburg / Germany [HG] <input type="radio"/> Leipzig / Germany [LG] <input type="radio"/> Melbourne / Australia [MA] <input type="radio"/> Moscow / Russia [MR] <input type="radio"/> Madrid / Spain [MS] <input type="radio"/> Oslo / Norway [ON] <input checked="" type="radio"/> Rio / Brazil [RB] <input type="radio"/> Rhodes / Greece [RG] <input type="radio"/> Roma / Italy [RI] <input type="radio"/> Sofia / Bulgaria [SB] <input type="radio"/> Stuttgart / Germany [SG] <input type="radio"/> Tokyo / Japan [TJ] <input type="radio"/> Vancouver / Canada [VC] <input type="radio"/> Wien / Austria [WA] <input type="radio"/> Zurich / Switzerland [ZS]	<p>WEATHER 4 NOW</p> <p>RIO BRAZIL</p> <p>☀</p> <p>TEMP: 27 C</p> <p>PRES: 1016 hPa</p> <p>DATE: 28.03.05</p> <p>TIME: 14:00 UTC</p> <p>DATA BY WEBSERVICEX.NET</p>	<p>WEATHER 4 NOW</p> <p>RIO BRAZIL</p> <p>☀</p> <p>TEMP: 27 °C</p> <p>PRES: 1016 hPa</p> <p>DATE: 28.03.05</p> <p>TIME: 14:00 UTC</p> <p>DATA BY WEBSERVICEX.NET</p>	

[Direkter Aufruf des SVG-Dokuments](#) (mit SVG-Viewer oder SVG-Handy)

[03/2005 by [Dr. Thomas Meinike](#) | [SVG - Learning By Coding](#) | [Datenbasis](#)]

http://svgmob.datenverdrahten.de/w4n_demo.php