Scalable Vector Graphics (SVG)

http://www.w3.org/TR/SVG11
http://www.w3.org/TR/SVG12 (WD)

SVG: 2D-Grafik in XML-Syntax

Erinnerung: Raster-Grafik vs. Vektor-Grafik

- **Raster-Grafik**
  - Als Ausgabemedium wird ein Pixel-Raster unterstellt
  - Elementare Anweisungen: Kontrolle jedes einzelnen Pixels

```
set(x1, y1, color1);
set(x1+1, y1+1, color1);...# Dünne Diagonale mit Farbe "color1"
```

- **Vektor-Grafik**
  - Entspricht dem Malen mit Stiften
    (charakterisiert durch Farbe, Dicke, Form der Spitze, Effekte)
  - Elementare Anweisungen:

```
select_pen(color1, width1);
move_to (x1, y1);
draw_to (x2, y2);   ...
```

  - Siehe auch "Logo", *turtle graphics*

SVG: 2D-Grafik in XML-Syntax

Stärken und Schwächen

- **Raster-Grafik**
  - Ideal für Photos und komplexe Szenen
  - Filter-Effekte und Verfremdungen einfach durch lokale Pixeloperationen möglich
- **Vektor-Grafik**
  - Ideal für technische Zeichnungen, Diagramme, Animationen
  - Skalierungen und Transformationen einfach durch Koordinatentransformationen möglich
- **Hybride Verfahren** dominieren den Alltag
  - Rastergrafiken: Skalierungsalgorithmen, *anti-aliasing*, Ebenen, ...
  - Vektorgrafiken: (Client-seitige) Filtereffekte definierbar

SVG: 2D-Grafik in XML-Syntax

- Grenzen der HTML-Möglichkeiten
  - Anzeige von Text: ok
  - Anzeige von (gerasterten) Bildern: ok
  - Anzeige von Grafiken: NEIN
  - Animationen: NEIN
- Heutige Auswege: Plug-ins, Scripting, Applets
  - JavaScript, DOM z.B. für dynamisches Verhalten, Animation
  - PDF Viewer als Plug-in für "beliebige" Dokumente
  - VRML Plug-ins für 3D-Grafik
  - Java Applets für diverse Client-seitige Aktivitäten
  - Flash Plug-in für 2D-Grafiken & Animationen!
  - ...

SVG: 2D-Grafik in XML-Syntax

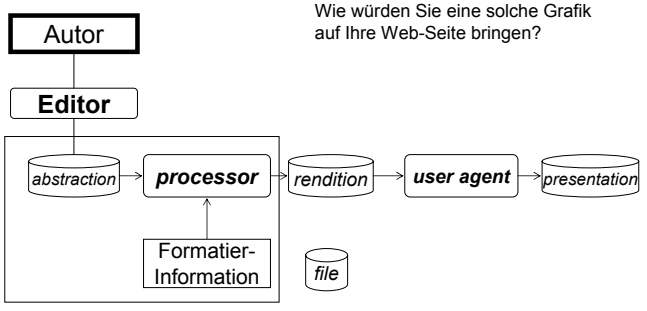
- Nachteile von Plug-ins & Applets
  - Keine echte Integration mit dem Rest der Anzeige
  - Proprietäre Techniken
  - Installationsaufwand für Plug-ins
  - Verfügbarkeit auf Client-Seite nicht voraussehbar bzw. garantiert
  - ...
- Ausweg:
  - Standardisierung einer 2D-Grafiksprache "SVG" durch das W3C
  - "Nahtlose" Integration einer Lösung für 2D-Grafik in die "XML-Generation" des WWW
  - Verwendung bereits existierender Standards
    XML 1.x, Namespaces, ...
  - Grundlage für bzw. Kooperation mit anderen Standards
    SMIL für Animationen, DOM, CSS, ...

SVG: 2D-Grafik in XML-Syntax

- Vorteile des SVG-Standards außerhalb des WWW
  - Standardisiertes Austauschformat für 2D-Grafiken
    Bessere Kooperationsmöglichkeiten, etwa zwischen Produkten von Adobe, Corel, Microsoft, ... - wenn gewollt!
  - Öffentlich verfügbares Know-How zur Entwicklung hochwertiger, wiederverwendbarer Grafiken
  - Mächtiger Sprachumfang!
  - Vereinheitlichung von Technologien durch XML-Grundlage
  - Leichte Integration mit diversen Textquellen dank *namespaces*
  - Kompakte, skalierbare Grafiken - ideal auch für mobile Geräte
  - Modularität erleichtert Wiederverwendbarkeit, etwa durch Einrichtung von Symbol-Bibliotheken

SVG: 2D-Grafik in XML-Syntax

Testfrage, zur Verdeutlichung:
Wie würden Sie eine solche Grafik auf Ihre Web-Seite bringen?



... und eine solche Animation?

SVG: Einführung

- "Historische" Entwicklung von SVG

– SVG 1.0	W3C-Empfehlung seit 5.9.2001
– **SVG 1.1**	**W3C-Empfehlung seit 14.1.2003**
– SVG 1.2	W3C-Entwurf seit 11.11.2002, aktueller WD vom 13. April 2005
– SVG Mobile Profiles	W3C-Empfehlung seit 14.1.2003

- Unterschiede

– SVG 1.0	Erstes Release überhaupt, eine DTD-Datei
– SVG 1.1	Modularisierung: Viele DTD *entities*; Errata Nichts wirklich Neues, Namensraum gleich!
– SVG 1.2	Neue Eigenschaften (Ergänzungen), hier nicht behandelt

- Konsequenz hier:
  - **Beschränkung auf SVG 1.0**, gelegentlich Bezug auf SVG 1.1 !

SVG: Einführung

- Beispiele für Firmen, deren Vertreter an SVG mitwirkten:

  DTP, CAD/CAM:
  Adobe Systems Inc., Autodesk, Corel Corp., Macromedia, Microsoft, Quark

  Mobile devices:
  Ericsson, Nokia

  Browsers:
  AOL, Netscape Comm. Corp., Opera

  "*Paper business*", *images*:
  Agfa-Gevaert, Canon, Eastman Kodak, Sharp, Xerox Corp.

  Systems:
  Apple, Hewlett-Packard, IBM, Sun Microsystems

  Organizations:
  OASIS, W3C

SVG: Einführung

- Eigener Anspruch von SVG
  - Eine Sprache zur Beschreibung von 2D-Grafiken in XML
  - 3 unterstützte Grafikobjekte

    Vektorgrafik-Formen (z.B. Linien, Polygone, Kurven), **Bilder**, **Text**

    Grafikobjekte können gruppiert, gestaltet, transformiert und in bestehende Objekte eingefügt werden.
  - Text kann aus beliebigen XML Namensräumen stammen
  - Zu den Eigenschaften von SVG zählen:

    Verschachtelte Transformationen, Begrenzungspfade (*clipping*), Semitransparenz (*alpha masks*), Filtereffekte, Schablonenobjekte, Erweiterbarkeit
  - SVG-Zeichnungen können dynamisch und interaktiv sein!

    Animation mittels *scripting* und XML DOM

SVG: Einführung

- SVG-Software

  Der SVG-Standard ist noch sehr neu. Daher wird erst in Zukunft mit voller Unterstützung zu rechnen sein.
  - ***Browser***:

    Mozilla Firefox (seit V 1.5 , 2005-11-29 mit nativer SVG-Unterstützung!)

    IE, nur mittels Plug-in
  - ***Plug-ins***:

    z.B. von Adobe und Corel
  - ***Stand-alone viewer***:

    z.B. "display" aus dem ImageMagick-Paket unterstützt SVG *basics*

- Im Kurs verwendet:

  squiggle Viewer aus dem Java-basierten Apache-Projekt batik (1.6)

  Mozilla Firefox 1.5

  PC-Demos mit Adobe's SVGView.exe 3.01 für Windows, IE

Scalable Vector Graphics Tutorium

http://svg.tutorial.aptico.de/

SVG: *Tutorial*

- Bemerkungen
  - Das W3C-Dokument zum SVG-Standard ist umfangreich und nicht einfach zu lesen für Einsteiger.
  - Es geht weit über unsere ca. einstündige Einführung in SVG hinaus.
- Methodischer Ansatz
  - Wir beschränken uns auf einige grundlegende SVG-Möglichkeiten.
  - Dabei legen wir Wert auf die Grafikobjekte, nicht Texte und Bilder.
  - Es wird keine systematische, theoretisch untermauerte Einführung versucht, sondern ein praxisnaher, beispielorientierter Tutoriumsansatz.
  - Als Grundlage dient u.a. das SVG-Tutorium der Fa. Aptico. (http://svg.tutorial.aptico.de/)

SVG: *Tutorial*

- Unterscheide:
  - "standalone" SVG-Dokumente
  - In anderen Dokumenten eingebettete SVG-Fragmente
- SVG - ein Fall für XML
  - SVG-Dokumente sind normale XML-Dokumente: Unicode-Basis, XML-Deklaration, Namensräume, Entities
  - Die SVG-Grundlage bilden DTD-Dateien.

SVG: *Tutorial*

- Konventionen für SVG allgemein
  - MIME-Type image/xml+svg
  - Extension svg, svgz (*gzip compressed SVG*)
- XML-Vorgaben für SVG 1.0
  - Standard: http://www.w3.org/TR/SVG10/
  - Namensraum: http://www.w3.org/2000/svg
  - FPI (PUBLIC id) -//W3C//DTD SVG 1.0//EN
  - SYSTEM id http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd
- XML-Vorgaben für SVG 1.1
  - Standard: http://www.w3.org/TR/SVG11/
  - Namensraum: http://www.w3.org/2000/svg
  - FPI (PUBLIC id) -//W3C//DTD SVG 1.1//EN
  - SYSTEM id http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd

SVG: *Tutorial*

- Grundgerüst eines SVG 1.0-Dokuments:

```
<?xml version="1.0" 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="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <!-- Ein Kommentar in einer leeren Grafik ... -->
</svg>
```

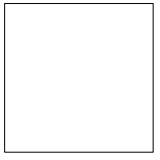
  - Attribute **width** und **height** bestimmen die Gesamtgröße der Grafik.
  - Längeneinheiten in SVG: em, ex, px (default), pt, pc, cm, mm, in; %

    Beispiel: `<svg width="10cm" height="8cm" ...>`

SVG: *Tutorial*

- Ein rotes Rechteck, ausgefüllt:

```
<?xml version="1.0" 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="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="200" height="200" style="fill:red"/>
</svg>
```



– Attribute **x** und **y:**
  Lage der oberen linken Ecke.

– Attribut **style**:
  Zahlreiche Möglichkeiten, vgl. CSS

SVG: *Tutorial*

- SVG-Elemente sind verschachtelbar!

```
<?xml version="1.0" 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="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <rect x="0" y="0" width="200" height="200" style="fill:red"/>
  <svg x="10" y="10" width="100" height="100">
    <rect x="0" y="0" width="100" height="100"
          style="fill:blue"/>
  </svg>
 </svg>
```

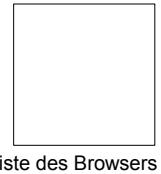
– Attribute **x** und **y:**
  Jeweils Lage der oberen linken Ecke.

– Inneres SVG-Element:
  Lage relativ zum Eltern-Element!

SVG: *Tutorial*

- Titel und Beschreibung:

```
<?xml version="1.0" 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="200" height="200" xmlns="http://www.w3.org/2000/svg">
  <title>Beispiel: title und desc</title>
  <desc>Ein einfaches schwarzes Rechteck</desc>
  <rect x="5" y="5" width="190" height="190"/>
</svg>
```



– **"title":**
  Art der Verwendung ist Sache des UA, z.B. Titelleiste des Browsers

– **"desc":**
  Möglichst präzise Beschreibung des Inhalts. Ersatztext für Browser ohne SVG-Unterstützung.
  Suchmaschinen werten diese Elemente aus - W3C: Empfehlung!

SVG: *Tutorial*

- Grundformen: **Rechtecke**

```
<?xml version="1.0" 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="410" height="110" xmlns="http://www.w3.org/2000/svg">
  <title>das rect-Element</title>
  <desc>9 (8 sichtbare) Rechtecke</desc>
  <!-- vier Rechtecke in der ersten Zeile -->
  <rect x="10" y="10" width="90" height="40" />

  <!-- das folgende Rechteck wird nicht dargestellt -->
  <rect x="110" y="10" width="90" height="40" fill="none" />

  <rect x="210" y="10" width="90" height="40" rx="5" ry="10"
        fill="none" stroke="black" />
  <rect x="310" y="10" width="90" height="40" rx="5" fill="none"
        stroke="black" />
  <!-- (Fortsetzung nächste Folie!) -->
```

SVG: *Tutorial*

- Grundformen: **Rechtecke**

```
<!-- die vier Rechtecke in der zweiten Zeile -->
<rect x="10" y="60" width="90" height="40" fill="#ff0000" />
<rect x="110" y="60" width="90" height="40" ry="5"
      fill="blue" stroke="black" />
<rect x="210" y="60" width="90" height="40"
      fill="red" stroke="blue" stroke-width="4" />
<rect x="310" y="60" width="90" height="40"
      style="fill:#33cc33; stroke:rgb(0,0,0);" />
<!-- das Rechteck, welches die gesamte Grafik umrahmt -->
<rect x="1" y="1" width="408" height="108"
      fill="none" stroke="blue" />
</svg>
```

SVG: *Tutorial*

- Grundformen: **Rechtecke**
  - Attribut **fill:**
    Füllfarbe. Werte entsprechen CSS-Möglichkeiten, z.B.
    Voreingestellte Farbnamen ("red") sowie "none", default: "black"
    Kompakte Hex-Notation ("#cc33cc")
    Explizite RGB-Angaben ("rgb(0,0,0)")
  - Attribut **stroke:**
    Rahmenlinie. Werte entsprechen Farbangaben, analog "fill"
    Default ist "none", also: Kein Rahmen!
  - Attribut **stroke-width**:
    Rahmenstärke, eine Länge
    Default ist "1px", also 1 Pixel
  - Attribute **rx, ry**:
    Rundungsradien der Ecken, in x- bzw. y-Richtung.
    Default ist "0", also keine Rundungen. rx oder ry genügt, wenn rx=ry.

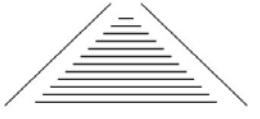
SVG: *Tutorial*

- Grundformen: **Linien**

```
<?xml version="1.0" 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="400" height="200" xmlns="http://www.w3.org/2000/svg">
  <title>das line-Element</title>
  <desc>nur Linien</desc>
  <!-- schwarze Linien -->
  <line x1="80" y1="160" x2="320" y2="160" style="stroke:black;
       stroke-width:2px;" />
  <line x1="90" y1="150" x2="310" y2="150" style="stroke:black;
       stroke-width:2px;" />
  <!-- usw. (weitere 10), dann: rote Linien -->
  <line x1="40" y1="165" x2="180" y2="30" style="stroke:red;
       stroke-width:2px;" />
  <line x1="360" y1="165" x2="220" y2="30" style="stroke:red;
       stroke-width:2px;" />
</svg>
```


SVG: *Tutorial*

- Grundformen: **Linien**
  - Abbildung zum Linienbeispiel:



  - Attribute **x1, y1, x2, y2**:
    Endpunkte der Linie
  - Attribut **style**:
    Nimmt im CSS-Stil Angaben zur Strichbreite und -Farbe auf, analog zu <rect>.
  - ACHTUNG:
    Attribut "fill" ist hier nicht sinnvoll!

SVG: *Tutorial*

- Grundformen: **Kreise** & **Ellipsen** <circle>, <ellipse>
  - Attribute **cx** und **cy**:
    Mittelpunktkoordinaten
  - Attribute **r** (Kreis) bzw. **rx, ry** (Ellipse):
    Kreisradius bzw. Radien der Ellipse
  - Attribute **fill, stroke, stroke-width**:
    Analog zu Rechtecken
- Beispiele
  - 2 Beispiele aus dem Aptico-Tutorial (nächsten Folien)
  - Beachte: Später erscheinende Objekte können die früheren überdecken !

SVG: *Tutorial*

- Grundformen: **Kreise** & **Ellipsen**

```
<?xml version="1.0" 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="410" height="210" xmlns="http://www.w3.org/2000/svg">
  <title>das circle-Element</title>
  <desc>ein Kopf aus Kreisen</desc>
  <!-- 2 schwarze Ohren -->
  <circle cx="150" cy="50" r="40" fill="black" />
  <circle cx="260" cy="50" r="40" fill="black" />
  <!-- der Kopf -->
  <circle cx="205" cy="100" r="80" fill="yellow" stroke="black" />
  <!-- die Augen -->
  <circle cx="180" cy="67" r="14" fill="white" stroke="black" />
  <circle cx="230" cy="67" r="14" fill="white" stroke="black" />
  <circle cx="180" cy="70" r="10" fill="black" />
  <circle cx="230" cy="70" r="10" fill="black" />
  <!-- (Fortsetzung nächste Folie...) -->
```


SVG: *Tutorial*

- Grundformen: **Kreise** & **Ellipsen**

```
  <!-- (Fortsetzung:) -->
  <!-- die Nase -->
  <circle cx="205" cy="90" r="15" fill="red" />
  <!-- der Mund -->
  <circle cx="205" cy="140" r="17" fill="red" stroke="black" />
  <circle cx="205" cy="130" r="17" fill="yellow" />
</svg>
```

  - Ergebnis:

SVG: *Tutorial*

- Grundformen: **Kreise** & **Ellipsen**

```
<?xml version="1.0" 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="410" height="210" xmlns="http://www.w3.org/2000/svg">
  <title>das ellipse-Element</title>
  <desc>4 Ellipsen und ein Kreis</desc>
  <!-- 4 farbige Ellipsen -->
  <ellipse cx="200" cy="50" rx="20" ry="40" style="fill:green;" />
  <ellipse cx="250" cy="100" rx="40" ry="20" style="fill:blue;" />
  <ellipse cx="200" cy="150" rx="20" ry="40" style="fill:yellow;" />
  <ellipse cx="150" cy="100" rx="40" ry="20" style="fill:red;" />
  <!-- der Kreis in der Mitte -->
  <circle cx="200" cy="100" r="19" style="fill:black;
  stroke:white; stroke-width:3px" />
</svg>
```

SVG: *Tutorial*

- Grundformen: **Kreise** & **Ellipsen**
  - Abbildung zum Ellipsenbeispiel:

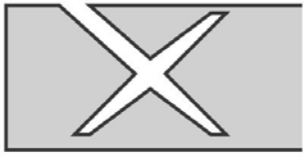
SVG: *Tutorial*

- Grundformen: **Polygon, Polygonzug**
  <polygon>,<polyline>
  - Attribut **points**:
    Koordinatenliste, wahlweise mit Blank oder Kommata getrennt.
    Empfehlung: x- und y-Wert mit Blank trennen, Paare mit Kommata!
  - Attribute **fill, stroke, stroke-width**:
    Analog zu Rechtecken
- Bemerkungen
  - Polygonzüge werden implizit geschlossen (durch eine gedachte Verbindung zwischen Anfangs- und Endpunkt).
  - Die so eingeschlossene Fläche wird standardmäßig gefüllt, was mit fill="none" explizit abzuschalten ist.

SVG: *Tutorial*

- Grundformen: **Polygonzug**

```
<?xml version="1.0" 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="410" height="210" xmlns="http://www.w3.org/2000/svg">
  <title>das polyline-Element</title>
  <desc>eine Polylinie mit Füllung</desc>
  <!-- eine gefüllte (!) Polylinie -->
  <polyline fill="lightgray" stroke="red" stroke-width="5px"
   points="400 10, 120 10, 200 80, 280 20, 300 20 220 100,
           300 180, 280 180, 200 120, 120 180, 100 180 180 100,
           80 10, 10 10, 10 200, 400 200" />
</svg>
```



SVG: *Tutorial*

- Styling, CSS
  - SVG baut auf CSS auf!
  - Die durch das Attribut "style" in verschiedenen Elementen bestehenden Gestaltungsmöglichkeiten lassen sich ersetzen durch gewöhnliche CSS-Regeln.
- Unterscheide externe vs. interne CSS-Regeln:
  - Externe CSS-Dateien:
    Wie gewohnt per PI einzubinden (hier nicht gezeigt, vgl. CSS-Kap.)
  - Interne CSS-Regeln:
    CSS-Regeln lassen sich auch in SVG-Daten einbetten.
    Siehe folgendes Beispiel!

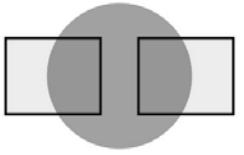
SVG: *Tutorial*

- Styling, CSS

```
<?xml version="1.0" 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="400" height="200" xmlns="http://www.w3.org/2000/svg">
  <title>das defs-Element und das style-Element</title>
  <desc>Stylevorschriften mit CSS zu Begin des Dokuments</desc>
  <!-- Definitionen: Style-Vorschriften, Symbole, etc. -->
  <defs> <style type="text/css">
      <![CDATA[
          rect {fill:yellow; stroke:black; stroke-width:.1cm;}
          .blauTransparent {fill:blue; stroke:black; opacity:.4;}
      ]]> </style> </defs>
  <!-- zwei gleich angemalte Rechtecke -->
  <rect x="50" y="50" width="125" height="100" />
  <rect x="225" y="50" width="125" height="100" />
  <!-- ein überlappender Kreis mit Transparenz -->
  <circle cx="200" cy="100" r="95" class="blauTransparent" />
</svg>
```


SVG: *Tutorial*

- Styling, CSS
  - Abbildung zum Styling-Beispiel:



- Bemerkungen
  - neu: "opacity" (Transparenz)
  - neu: Klassenbildung (beachte Punkt vor *identifier*)
  - Zentrale Stilpflege: Alle "rect"-Elemente gleich behandelt
  - Kompakterer SVG-Code: Stilangaben ausgelagert!

SVG: *Tutorial*

- Gruppenbildung <g>
  - Mehrere SVG-Elemente können zu einer Gruppe zusammengefasst werden. Es genügt, sie mit dem Element "g" zu umgeben.
  - Gruppen sind kaskadierbar (Gruppen in Gruppen). Dadurch lassen sich komplexe Formen bilden und als neue Einheiten behandeln, z.B. gemeinsam verschieben oder allgemein transformieren.
  - Attribut **style**
    Gemeinsame Stil-Spezifikationen der Gruppe können in g erfolgen.
  - Attribut **id**
    Gruppen können Namen erhalten und darüber referenziert werden!

SVG: *Tutorial*

- Definition & Benutzung von Objekten: <defs>,<use>
  - Einfache SVG-Elemente wie auch Gruppen können zunächst nur definiert werden (Bibliotheksbildung).
  - Späterer Abruf ist dann per Referenz möglich, wobei lokale Angaben hinzugefügt bzw. geerbt werden.
  - Technik der Referenzierung: ID-Attributvergabe und Linkangaben mittels Standards „XLink“ und „XPointer“.
  - **Element "defs"**
    Container zur Aufnahme diverser Definitionen.
  - **Element "symbol"**
    Zur Definition eines Objekts, i.w. Gruppierungswirkung.
    Verwendung wie "**g**", aber nur via "**use**" wirksam!
  - **Element "use"**
    Referenzierung vorher definierter Objekte.

SVG: *Tutorial*

- Definition & Benutzung von Objekten

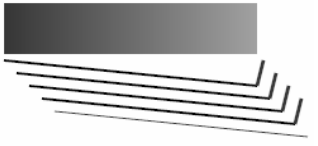
```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="250" height="120"
  xmlns:xlink="http://www.w3.org/1999/xlink" >
  <title>das defs-Element und das use-Element</title>
  <desc>Referenzen mit use</desc>
  <!-- Im Definitionsbereich (defs-Container) werden ein
  Farbverlauf und eine Linie definiert. Die hier festgelegten
  Objekte werden erst nach Referenzierung angezeigt -->
  <defs>
    <linearGradient id="farbverlauf">
      <stop offset="0%" stop-color="red" />
      <stop offset="99%" stop-color="#33cc33" /> </linearGradient>
    <symbol id="li_grp" style="stroke:black; stroke-width:2" >
      <line id="li" x1="5" y1="50" x2="205" y2="70" />
      <line id="li2" x1="205" y1="70" x2="210" y2="50"
        stroke="red" stroke-width="3" /> </symbol>
  </defs>  <!-- Fortsetzung auf der nächsten Folie... -->
```


SVG: *Tutorial*

- Definition & Benutzung von Objekten

```
  <!-- Fortsetzung -->
  <!-- Das Rechteck referenziert den Farbverlauf -->
  <rect x="5" y="5" width="200" height="40"
      style="fill:url(#farbverlauf);" />
  <!-- 4 Instanzen der Liniegruppe, 1 der ersten Linie -->
  <!-- Farbe der Linie "li" wird verändert; swidth geht verloren! -->
  <use              xlink:href="#li_grp" />
  <use x="10" y="10" xlink:href="#li_grp" />
  <use x="20" y="20" xlink:href="#li_grp" />
  <use x="30" y="30" xlink:href="#li_grp" />
  <use x="40" y="40" xlink:href="#li" style="stroke:red" />
</svg>
```

- Ergebnis:



SVG: *Tutorial*

- Transformationen: Das Attribut **transform**
  - Typische Verwendung: Innerhalb <use>
  - Wirkung:

    Ausführung einer oder mehrerer Transformationen vor der Übernahme des referenzierten Objekts.
  - Transformationsarten

Verschieben	**translate**(tx [ty])
Skalieren	**scale**(sx [sy])
Rotieren	**rotate**(a [cx cy])
Verzerren	**skewX**(a), **skewY**(a)
Allgemeine T.	**matrix**(a b c d e f)

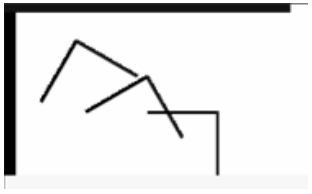
SVG: *Tutorial*

- Transformationen: Ein Beispiel

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="250" height="120"
  xmlns:xlink="http://www.w3.org/1999/xlink" >
  <title>Transformationen</title>
  <desc>Referenzen mit use</desc>
  <defs>
    <symbol id="KK" stroke-width="4" >
      <line x1="0" y1="0" x2="0" y2="50" />
      <line x1="0" y1="0" x2="50" y2="0" stroke="blue"/>
    </symbol></defs>
  <use xlink:href="#KK" transform="scale(4 3)"/>
  <use xlink:href="#KK" transform="translate(50  25) rotate(30)"/>
  <use xlink:href="#KK" transform="translate(100 50) rotate(60)"/>
  <use xlink:href="#KK" transform="translate(150 75) rotate(90)"/>
</svg>
```

SVG: *Tutorial*

- Transformationen: Ein Beispiel
  - Ergebnis:



SVG: *Tutorial*

- Animationen:
  - Ziel: Objekte einem zeitlichen Verlauf aussetzen
  - Typische Angaben:
    Welches Objekt ist zu animieren?
    Welche Eigenschaft soll animiert werden?
    Startwert, Endwert?
    Zeitverlauf: Von ... bis ...
    Ende-Verhalten: Ende-Zustand beibehalten, verschwinden, ...
  - Beispiele:
    Änderungen der Durchsichtigkeit (Ein- und Ausblenden)
    Farbänderungen, Dickenänderungen, ...
    Kontinuierliche Transformationen von ... bis ...,
    Bewegung entlang von Pfaden

SVG: *Tutorial*

- Animationen: Das Auto-Beispiel
  - Wir wollen Animationen hier nur kurz streifen.
  - On-line Demo "Auto". Code-Auszug:

```
  <!-- ... Nun die Straße -->
 <line x1="0" y1="268" x2="600" y2="268" stroke="black" />

  <animateTransform xlink:href="#rad_hinten" type="rotate"
       attributeName="transform" begin="0s" dur="2s"
       from="0 85 250" to="360 85 250" repeatCount="7" />
  <animateTransform xlink:href="#rad_vorn" type="rotate"
       attributeName="transform" begin="0s" dur="2s"
       from="0 205 250" to="360 205 250" repeatCount="7" />

  <animateTransform xlink:href="#auto" type="translate"
       attributeName="transform" begin="0s" dur="10s"
       from="-250" to="600" />
  <animateTransform xlink:href="#auto" type="translate"
       attributeName="transform" begin="10s" dur="4s"
       from="-250" to="50" fill="freeze" />
</svg>
```


SVG: *Tutorial*

Einbettung von SVG in HTML

- Es gibt folgende Möglichkeiten:
  1. Als eigenständiges SVG-Dokument aufrufen
     (nichts Neues...)
  2. Per Verweis einbetten
     mittels **`<object>, <embed>, <img>`**
  3. SVG-Code nativ in XHTML-Dokument einbetten
     dabei Unterscheidung durch Gebrauch von Namensräumen!
  4. Mit einem externen Verweis einbinden
     mittels **`<a>`**
  5. Als Referenz von Style-Eigenschaften
     `body{background-image:url(myfile.svg);}`

SVG: *Tutorial*

- Beispiel mit <object> (Multimedia-Objekt)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/html4/transitional.dtd">
<html>
  <head>
    <title>SVG in HTML</title>
  </head>
  <body style="margin-left:20px; margin-top:10px;">
    <p style="font-family:Verdana;">
       SVG Grafik durch das object-Tag eingebunden.
       <br> Wird von Firefox und IE korrekt dargestellt.</p>
    <object data="smilie.svg" type="image/svg+xml" width="280"
          height="280"> Sie benötigen einen SVG-Viewer </object>
   </body>
</html>
```

- Tipp:
  - Achten Sie auf passende Größen in HTML und SVG.

SVG: *Tutorial*

- Beispiel mit <embed> (Netscape-Variante)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/html4/transitional.dtd">
<html>
  <head>
    <title>SVG in HTML</title>
  </head>
  <body style="margin-left:20px; margin-top:10px;">
    <p style="font-family:Verdana;">
       SVG Grafik mit dem embed-Tag eingebunden.
       <br> Wird von Firefox und IE korrekt dargestellt.</p>
    <embed src="smilie.svg" type="image/svg+xml"
         width="280" height="280" />
   </body>
</html>
```


SVG: *Tutorial*

- Beispiel mit <img>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/html4/transitional.dtd">
<html>
  <head>
    <title>SVG in HTML</title>
  </head>
  <body style="margin-left:20px; margin-top:10px;">
    <p style="font-family:Verdana;">
       SVG Grafik durch das img-Tag eingebunden.
       <br> Wird noch nicht dargestellt.</p>
    <img src="smilie.svg" width="280" height="280"
       border="0" alt="SVG Grafik"/>
   </body>
</html>
```


SVG: *Tutorial*

- Direkte Einbettung in XHTML

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
  <head><title>SVG in XHTML</title></head>
  <body style="margin-left:20px; margin-top:10px;">
   <p style="font-family:Verdana;">
       SVG Quellcode im XHTML-Quellcode eingebettet
       <br/> Wird von Firefox 1.5, aber nicht von IE dargestellt!</p>
   <svg xmlns="http://www.w3.org/2001/svg10" width="280" height="280">
   <!-- <title>, <desc>, <defs>; <use>...<use> ... SVG-Inhalt! -->
   </svg>
  </body>
</html>
```

- **NEU:** Methode erst seit Erscheinen von Firefox 1.5 (Nov. 2005) generell verfügbar!

SVG: *Tutorial*

- Beispiel mit <a>

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/html4/transitional.dtd">
<html>
  <head>
    <title>SVG in HTML</title>
  </head>
  <body style="margin-left:20px; margin-top:10px;">
    <p style="font-family:Verdana;">
       SVG Grafik als Verweisziel, mit dem a-Tag realisierbar.
       <br> Wird von allen SVG-fähigen Browsern dargestellt.</p>
    <a href="smilie.svg">SVG Grafik<a/>
   </body>
</html>
```


SVG: *Tutorial*

- SVG-Hintergrundbild, mittels CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
   "http://www.w3.org/TR/html4/transitional.dtd">
<html>
  <head><title>SVG in HTML</title></head>
  <style type="text/css">
   body {background-image:url(smilie.svg);
         background-attachment-fixed;
         background-position:40px 40px;
         background-repeat:no-repeat;}
  </style>
  <body style="margin-left:20px; margin-top:10px;">
    <p style="font-family:Verdana;">
       SVG Grafik als Referenz einer Style-Eigenschaft (Hintergrund).
       <br> Wird noch nicht dargestellt.</p>
   </body>
</html>
```