Scalable Vector Graphics

Maurizio Tesconi

24 marzo 2015

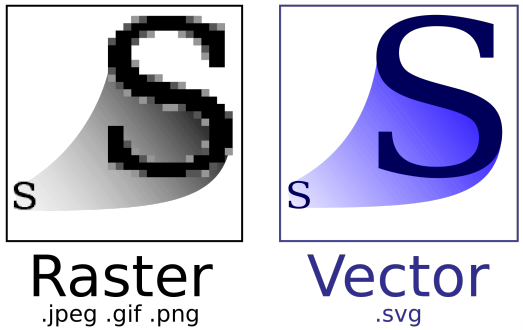
Raster graphics images

- Lossy (jpeg, jpeg2000)
- Lossless (gif, png, tiff, …)
- Fixed resolution
- Can be very large
- Original “information” is lost
- Difficult to add metadata
- Difficult to adapt to viewing environment
- Limited interaction (except for animated gifs…)
- Client/server side image maps for linking
- Proprietary solutions
- Proprietary authoring tools, players, …

Jpeg compression (Lossy)

Scalable Vector Graphics

- Resolution independent
- Keeps “content” information
- Textual encoding (too)
- Let the client side interpret the content
- Searchable
- Easy to edit, add links, etc
- Base it on XML
- Would integrate well with the rest of the Web



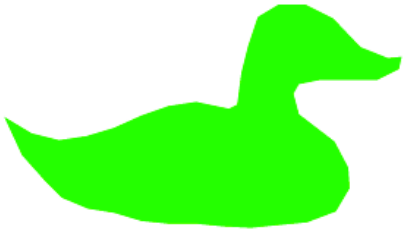
SVG versions

- W3C Working Group was created in 1998
- Specification quite stable by 2000, with first implementations
- Recommendations:
  - SVG 1.0 (September 2001)
  - SVG 1.1 and SVG Mobile (January 2003)
  - SVG 1.1 (second edition) (August 2011)
- SVG 1.1 and SVG 1.0 are functionally identical
- SVG 1.1 is a “modularization” of SVG 1.0
- Work is continuing towards SVG 1.2 and SVG 2.0

“SVG is the HTML for Graphics”

A Simple SVG Example

```
<svg width="600" height="300">
    <g transform="translate(10 10)">
        <g stroke="none" fill="lime">
            <path d="M 0.0 112 L 20 124 L 40 129 L 60 126 . . . </path>
        </g>
    </g>
</svg>
```

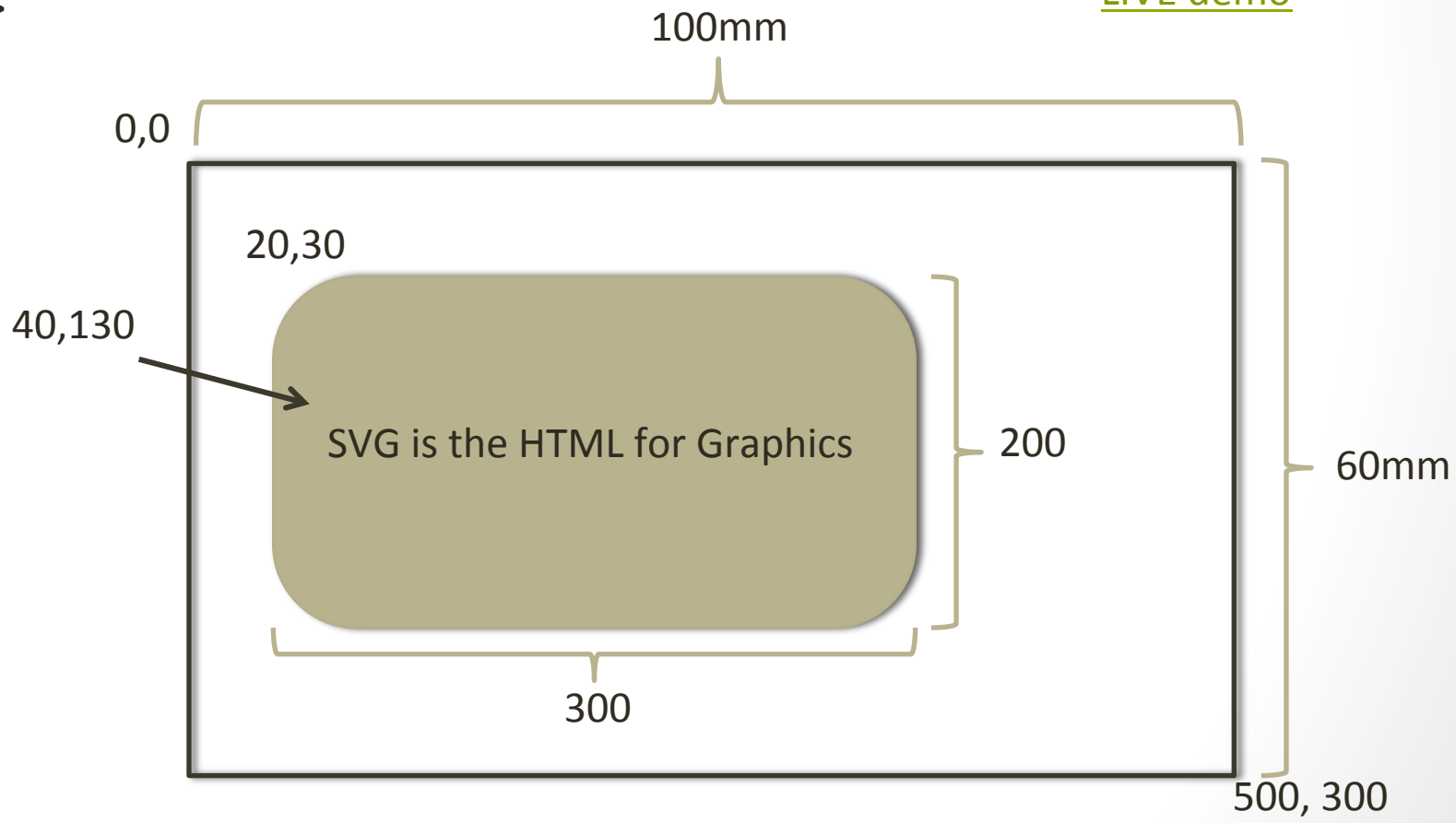


- svg is the top XML element, with a coordinate system
- g is used for (hierarchical) grouping
- path is a general tool for geometry definition
- paths can be filled, stroked, …
- paths can be transformed, animated, filtered, …

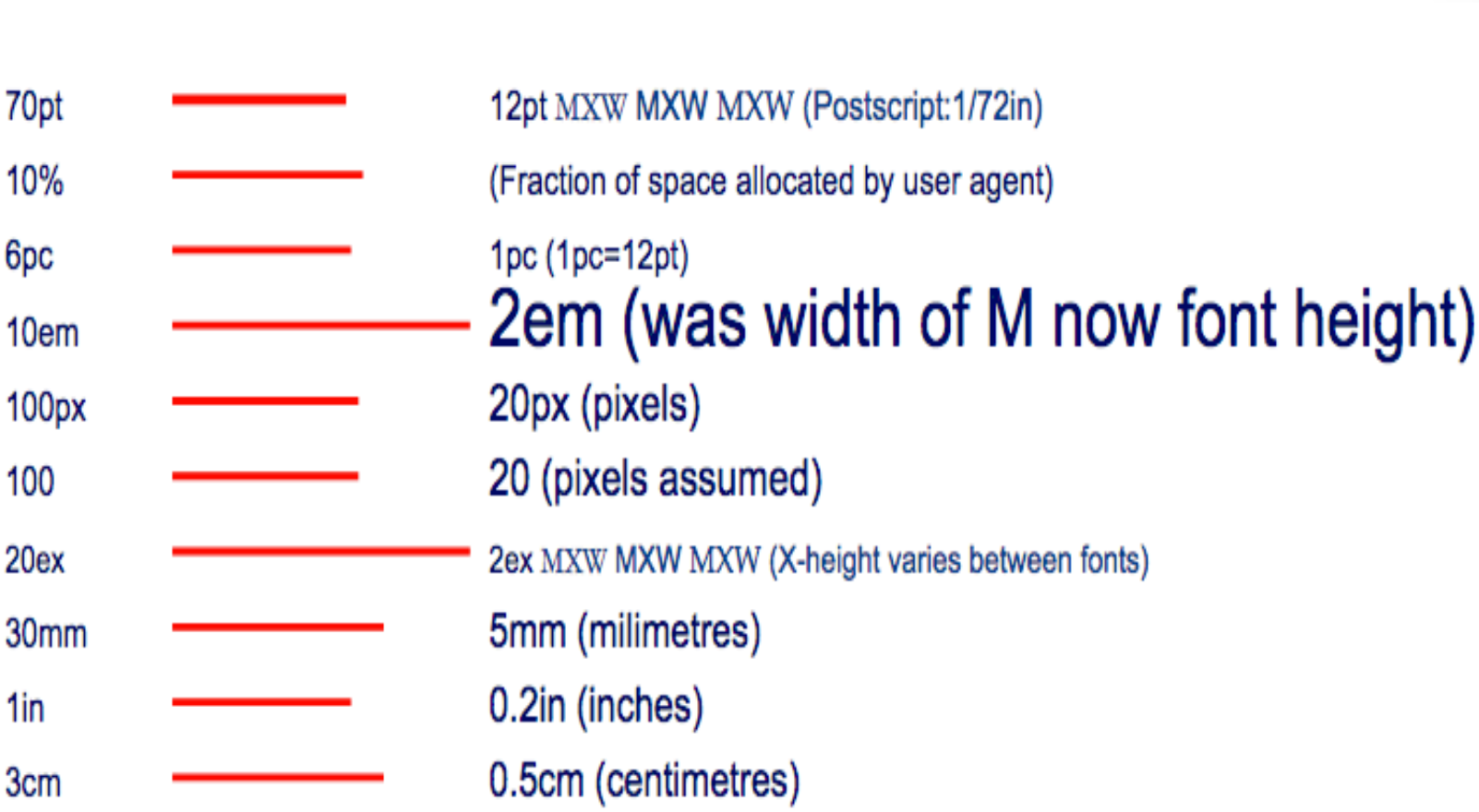
Example of Coordinate System

```
<svg width="100mm" height="60mm" viewBox="0 0 500 300">
        <rect x="20" y="30" width="300" height="200" rx="10" ry="10"/>
        <text x="40" y="130">SVG is the HTML for Graphics</text>
...
</svg>
```

LIVE demo



Large palette of units



SVG Coordinate Systems

- The canvas is the space or area where the SVG content is drawn.
- The viewport is the viewing area where the SVG will be visible.
- You can set the viewport with *width* and *height* attributes
- You can specify your own user coordinate system using the viewBox attribute.
- If the user coordinate system you choose has the same aspect ratio (ratio of height to width) as the viewport coordinate system, it will stretch to fill the viewport area

```
<svg width="800" height="600" viewbox="0 0 800 600">
   <!-- SVG content drawn onto the SVG canvas -->
</svg>
```

live DEMO

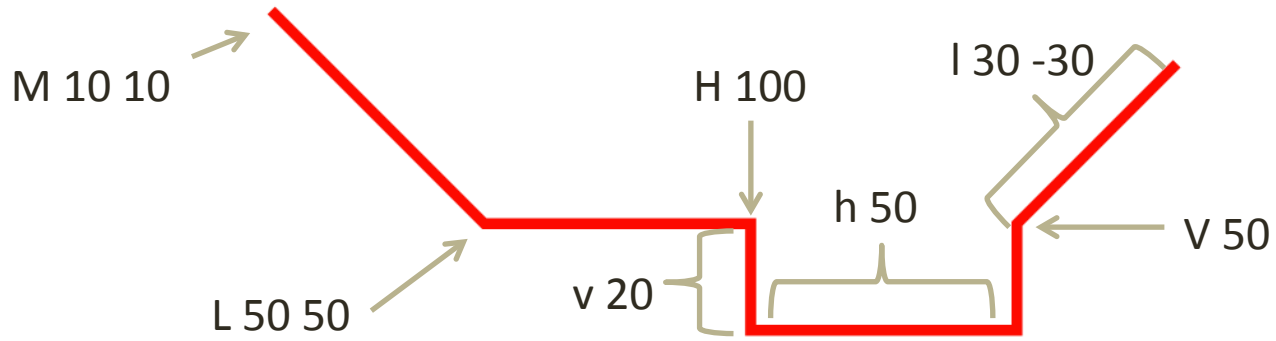
preserveAspectRatio = “none”

Geometry: Path Expressions

- Define a general contour with the d attribute:

<path d="M 0 112 L 20 124 L 40 129 L 60 126...z" />

- A sequence of command characters and coordinates:
  - M: move
  - L, H, V: line (general, horizontal, vertical)
  - Q, C: Bézier curve (quadratic, cubic)
  - A: (elliptical) arc
- Contour can be closed or open
- The character Z closes a path
- Lower case letter define relative coordinates

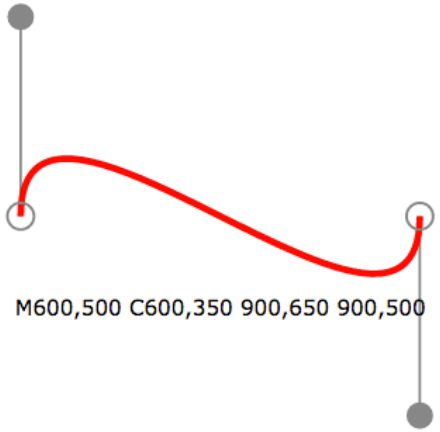


Path examples

M100,200 C100,100 400,100 400,200

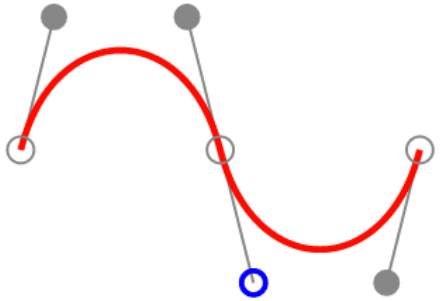
M600,200 C675,100 975,100 900,200

M100,500 C25,400 475,400 400,500



M600,500 C600,350 900,650 900,500

M100,800 C175,700 325,700 400,800



M600,800 C625,700 725,700 750,800
S875,900 900,800

Path “Shortcuts”

<rect x=".." y=".." width=".." height=".."/>

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

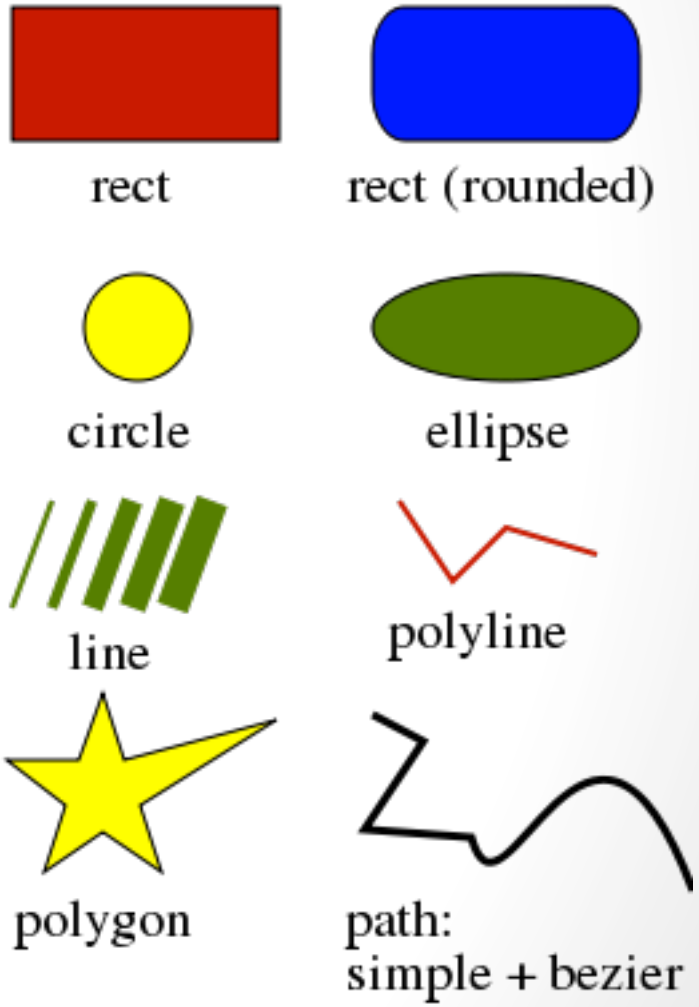
<ellipse cx=".." cy=".." rx=".." ry=".."/>

<polyline points=".."/>

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

<polygon points=".."/>

<path d=".."/>



live DEMO

Stroke attributes

Stroke/Fill default

For stroke:

- stroke: none (except for line, which is black)
- stroke-width: 1
- stroke-linecap: butt
- stroke-linejoin: miter
- stroke-dasharray: none
- stroke-opacity: 1

For fill:

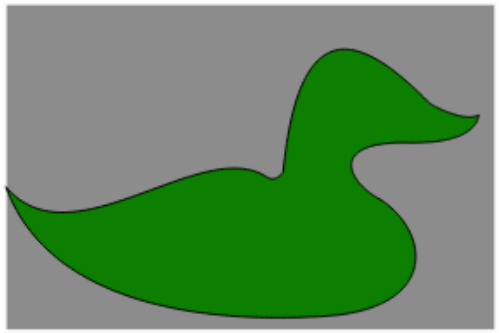
- fill: black
- fill-rule: nonzero
- fill-opacity: 1

<circle cx="50" cy="50" r="50" />

Transformations

Transformations can be defined on all elements

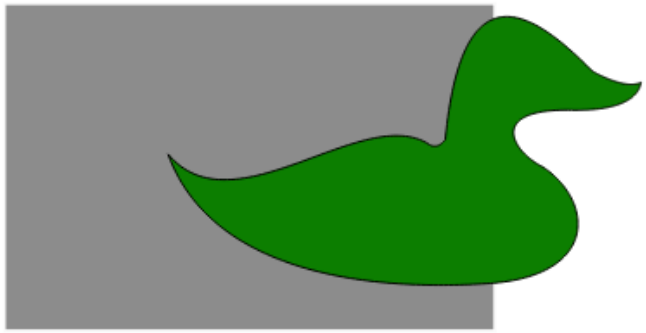
- translate, scale (both in X and Y), skewX and skewY, rotate
- concatenation of all these



Translate: 100 in x, -20 in y

<path transform="translate(100, -20)"

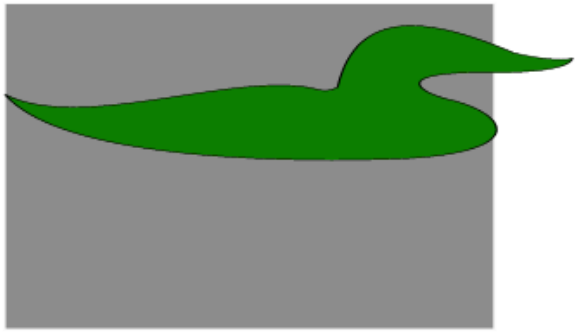
d="M 0 112c40 48 120-32 160-6 ...z"/>



Scale 1.2 in x and 0.5 in y

<path transform="scale(1.2, 0.5)"

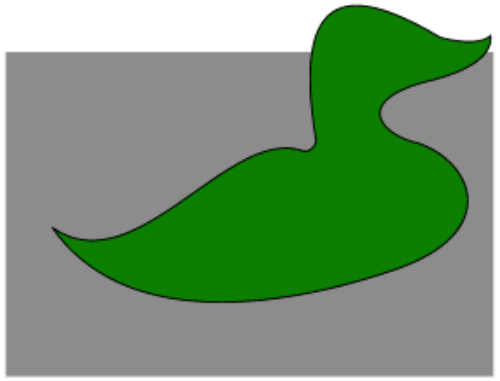
d="M 0 112c40 48 120-32 160-6 ...z"/>



Rotate: -15 degrees about the origin

<path transform="rotate(-15)"

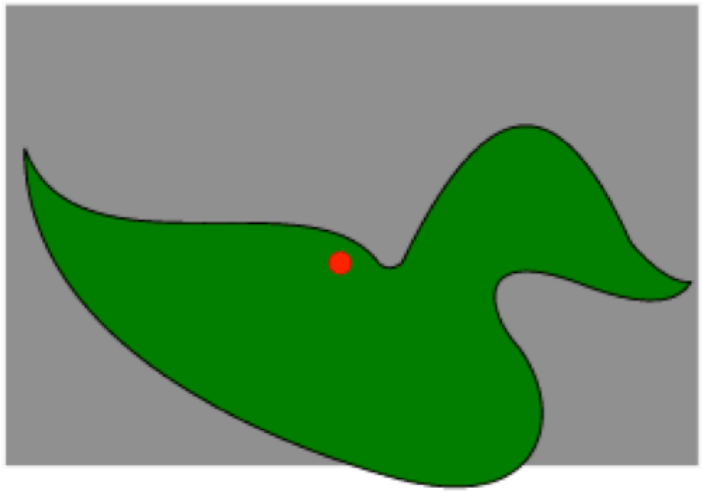
d="M 0 112c40 48 120-32 160-6 ...z"/>



Rotation

Rotation about the point (145, 112)

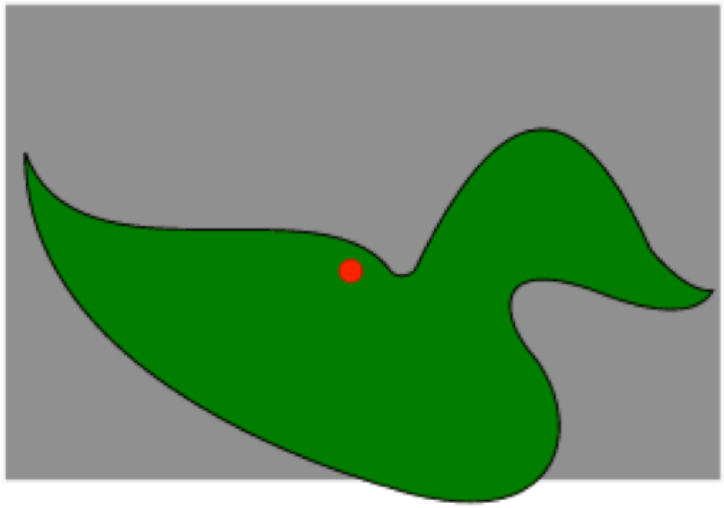
transform="translate(145, 112) rotate(20) translate(-145, -112)"



Transformation DEMO

Rotation about the point (145, 112)

transform="rotate(20,145,112)"

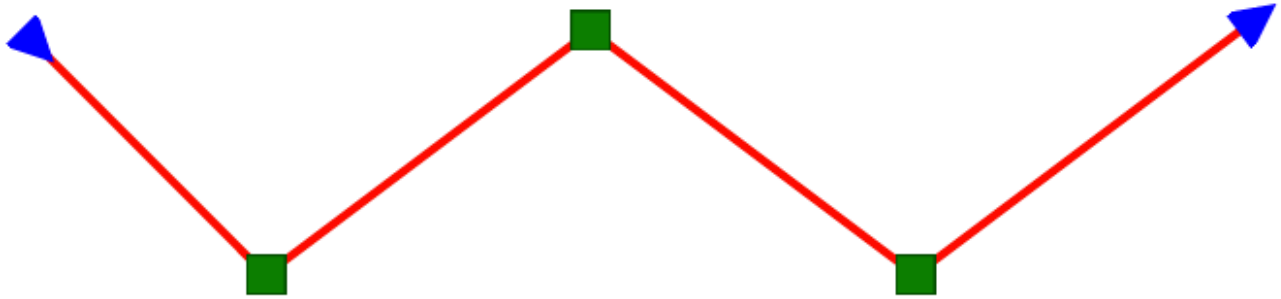


Grouping

- Elements can be “grouped” in a g element
- Well-known concept in graphics packages
- Groups
  - can be nested
  - can set transformations, attributes, etc,
  - inherited by their constituents ○can be named, can be referred to in, eg, URL-s
  - can make the content more accessible for specialized browsers
  - eg, browsers for visually impaired ○make SVG code more readable
- Typical usage of a group:

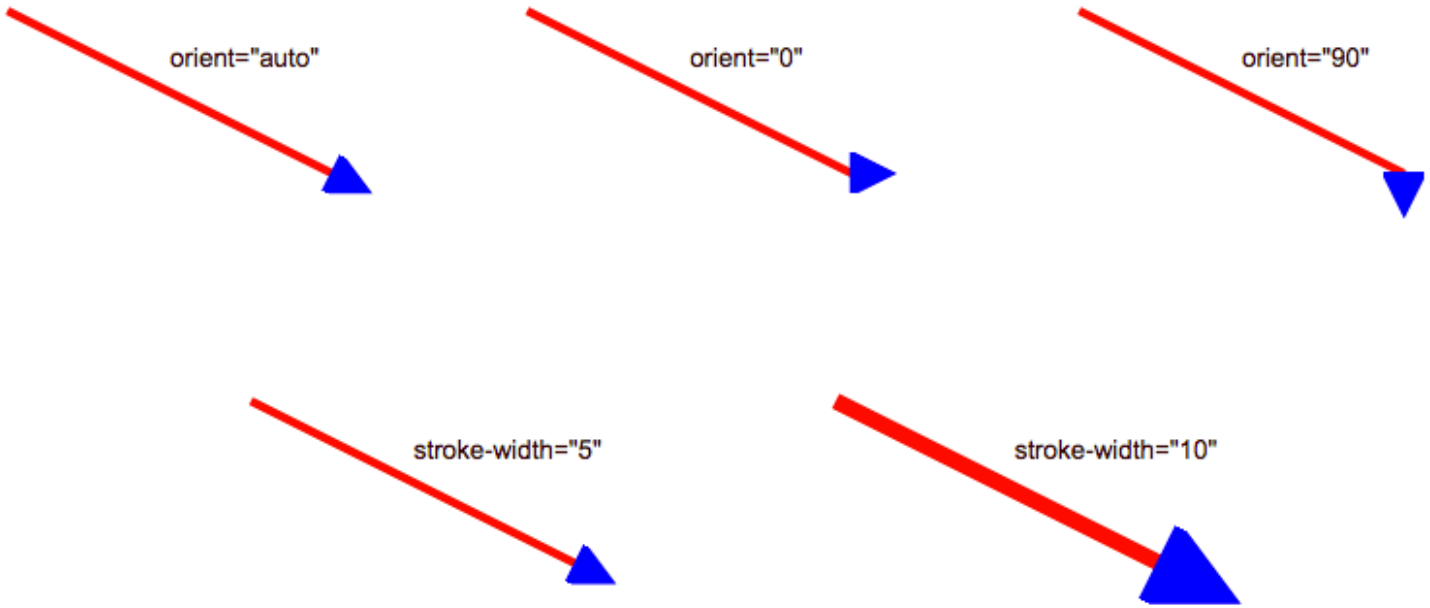
```
<g id="duck" transform="translate(100,345)">
        <desc>This is the duck></desc>
        <path d="...."/>
</g>
```

Marker Symbols



live DEMO

```
<marker id="Tr" viewBox="..." markerUnits="strokeWidth" orient="auto"><path .../>...
<marker id="Pt" viewBox="..." markerUnits="strokeWidth" orient="0"><path .../>...
<polyline marker-start="url(#Tr)" marker-mid="url(#Pt)" marker-end="url(#Tr)" ... />
```



Text

Is a separate element

<text x=".." y="..">Blablabla</text>

- Has tons of attributes (mostly from CSS):
  - font, weight, font style, size
  - color (both outline and inside a character)
  - writing mode, text anchor
- Text can be positioned on a curve:

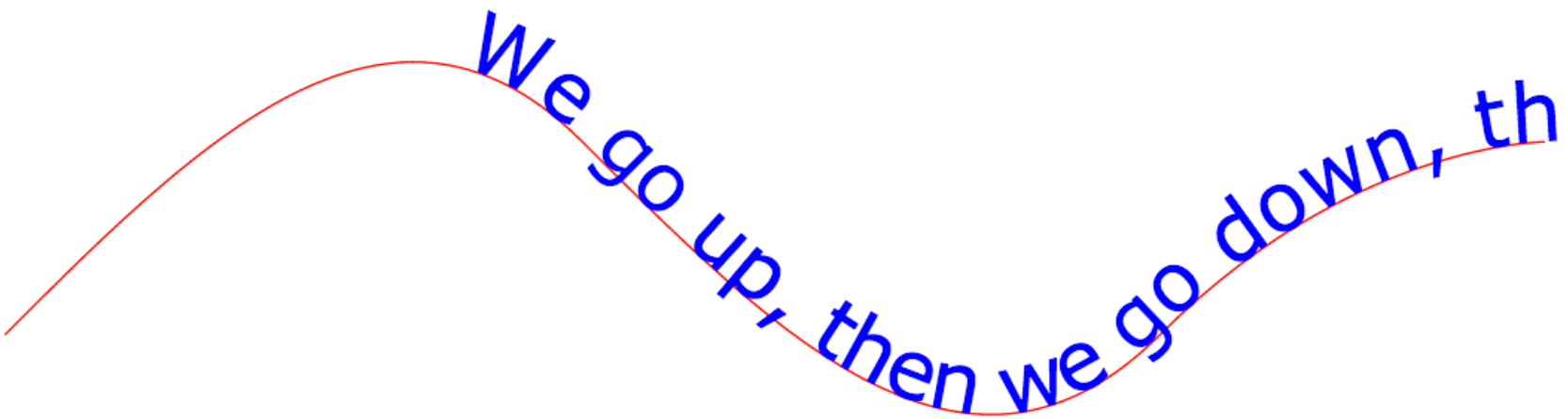
<path id="path" d="……." />

<text><textPath xlink:href="#path"> ...</textPath></text>

Text on path + offset

```
<text font-family="Verdana" font-size="42.5" fill="blue" >
  <textPath xlink:href="#MyPath" startOffset="30%">
    We go up, then we go down, then up again
  </textPath>
```

live DEMO



Image

```
<image x="20" y="20" width="100" height="50"
xlink:href="cat.png" />
```

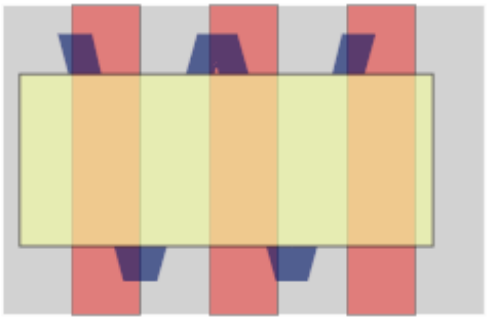
Rendering Model

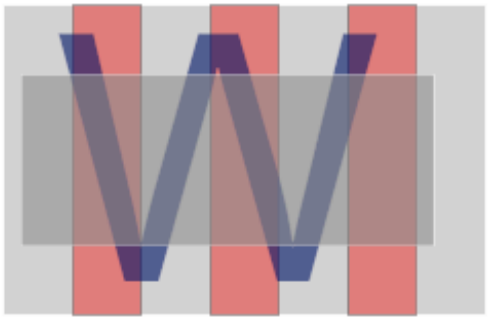
- SVG uses the painter's model for rendering:
  - each operation "paints" on the output device successively
  - rendering order is implicit to the XML document
  - groups are rendered separately, then mapped on the output

Opacity on Groups

```
<g style="opacity:0.5">
    <text x="30" y="160">W</text>
    <rect width="240" height="100" />
</g>
```

```
<text style="opacity:0.5" ... >W</text>
<rect style="opacity:0.5" ... />
```





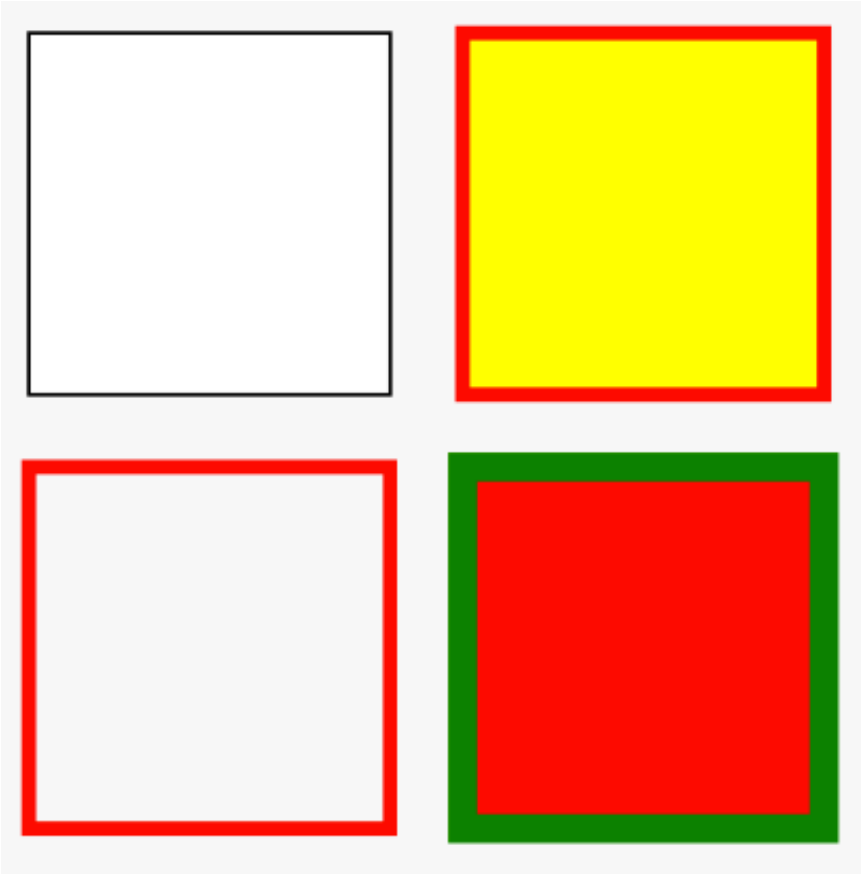
Clipping

```
<clipPath id="myClip">
        <circle cx="200" cy="10" r="50"/>
</clipPath>
<image clip-path="url(#myClip) " />
```

(the unclipped image...)

SVG styling with CSS

```
<rect x="20" y="20" width="100" height="100" />
<rect class="different" x="20" y="140" width="100" height="100" />
<rect class="different again" x="140" y="20" width="100" height="100" />
<rect id="else" x="140" y="140" width="100" height="100" />
```



CSS

```
rect {
  stroke:black;
  fill:white
}

rect.different {
  stroke:red;
  stroke-width:4;
  fill:none
}

rect.again {
  fill:yellow
}

rect[id ~= "else"] {
  stroke:green;
  stroke-width:8;
  fill:red
}
```

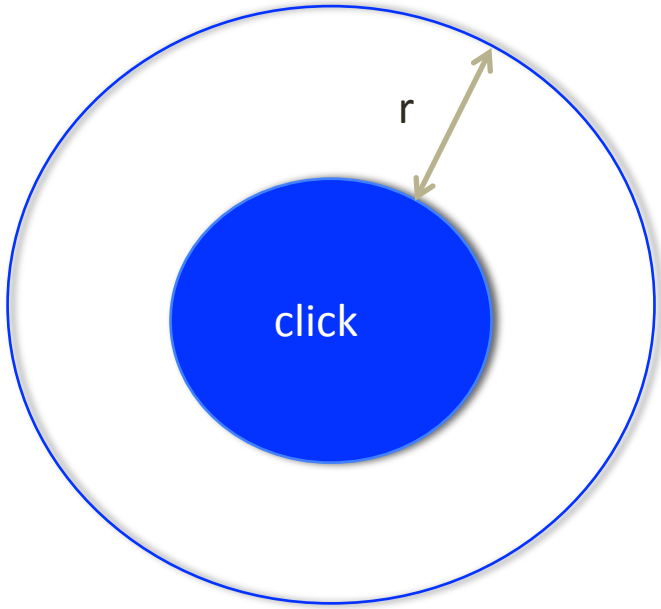
SVG Animation

SVG Animation = change object attributes dynamically

- Animation is controlled through a set of animation objects
  - reusing parts of W3C's SMIL2.0 Specification (Synchronized Multimedia Integration Language Version 2.0)
- Declarative syntax, no real increase in file size
- Animation is performed on client side
- Practically all attributes can be changed
  - XML: coordinates, path expression, filter characteristics, …
  - CSS: colour, visibility, opacity, …

Animation Examples

```
<circle id="c1" cx="200" cy="200" r="50" style="fill:blue; cursor:pointer">
    <animate id="an1" attributeName="r" from="50" to="100" dur="3s" fill="freeze"
    begin="c1.click"/>
    <animate attributeName="r" from="100" to="50" dur="3s" fill="freeze"
    begin="an1.end"/>
</circle>
```



examples from W3C

SVG & Javascript

- jQuery SVG
  - A jQuery plugin that lets you interact with an SVG canvas.
  - http://keith-wood.name/svg.html
- Snap.svg
  - is designed for modern browsers and therefore supports the newest SVG features like masking, clipping, patterns, full gradients, groups, and more.
  - http://snapsvg.io/
- SVG.JS
  - A lightweight library for manipulating and animating SVG.
  - http://svgjs.com/

SVG