

# Dynamic SVG generation under Firefox 1.5 using JavaScript, XML and XSLT

*Thomas Meinike*

Merseburg University of Applied Sciences / Department of Computer Science and  
Communication Systems

[thomas.meinike@hs-merseburg.de](mailto:thomas.meinike@hs-merseburg.de)

## Introduction

The first specification of Scalable Vector Graphics (SVG) was released by the W3C organization in 2001. SVG is a vocabulary which describes two-dimensional graphics in XML syntax. In 2003 a modularized version with number 1.1 including the mobile profiles SVG Tiny and SVG Basic came out. SVG provides vector shapes like circles, rectangles, lines or paths for more complex graphical objects. There are also special techniques available: gradients, filters, animations and transformations. Formatting is done by using presentation attributes or CSS properties [*W3C01*].

SVG is attractive for designers and programmers. It supports other technologies such as ECMAScript (the standardized form of JavaScript) or XSLT in the field of interactive and dynamic content creation. In the first few years users had to install a plug-in for their web browsers. In most cases this was the Adobe SVG Viewer software. Recently more and more native SVG implementations were

launched on the browser market. This presentation covers some SVG capabilities of the popular Firefox browser in version 1.5 and upgrades.

## SVG in 5 minutes

SVG graphic documents are pure XML files consisting of elements, attributes, entities, CSS rules and textual content. The whole usable syntax is defined in document type definitions (DTDs) for the current final versions 1.0 and 1.1 (full specifications and modularized parts).

Drawings consist of basic shapes: circles, ellipses, lines, polylines, polygons, rectangles, paths as well as text. These are the defined XML elements:

```
<circle cx="..." cy="..." r="..." />
<ellipse cx="..." cy="..." rx="..." ry="..." />
<line x1="..." y1="..." x2="..." y2="..." />
<polyline points="x1,y1,...,xn,yn" />
<polygon points="x1,y1,...,xn,yn" />
<path d="..." />
<rect x="..." y="..." width="..." height="..." />
<text x="..." y="...">text content</text>
```

Coordinates and presentational information like fill color, opacity value or stroke width are specified as attributes. CSS properties are also available, but not for SVG Tiny documents. As an example, let's create a red circle. A center point with the x- and y-coordinates (cx, cy) and radius r are needed:

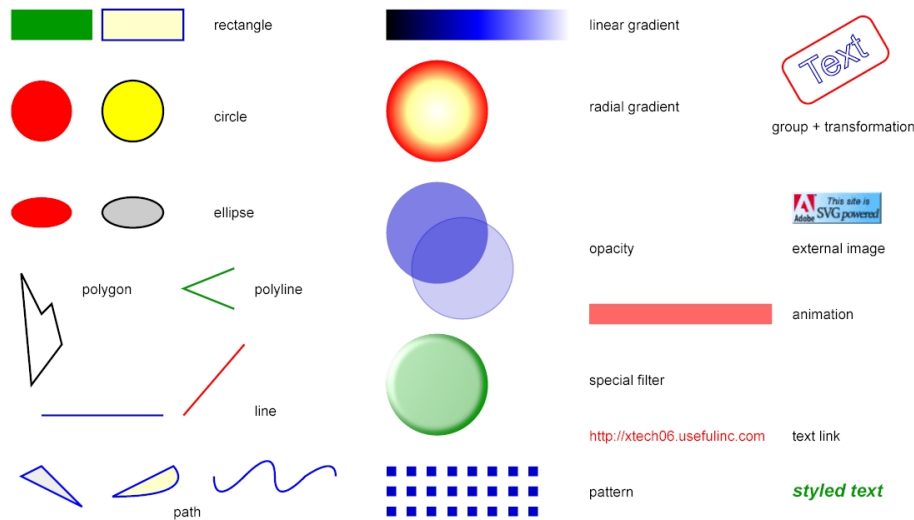
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <circle cx="100" cy="200" r="30" fill="#FF0000" />
</svg>
```

The fill color refers to the hexadecimal codes known from HTML. Other formats are also allowed (short hexadecimal form #F00, color word red and the CSS-like RGB declarations rgb(255, 0, 0) or rgb(100%, 0%, 0%). Numeric values without an extension refer to user coordinates (i. e. pixels). The coordinate

system starts in the topleft-hand corner (point 0 , 0) of the initial viewport. More detailed coordinate handling is possible with the `viewPort` attribute located at the root element `svg`.

There are more special techniques: gradients, filters, animations, transformations and scripting enhancements. Figure 1 shows the main objects and effects.

#### SVG – objects and effects



[svg\\_objects\\_effects.png](#)

Figure 1. Objects and effects provided by SVG.

## Practical SVG with Firefox

The final version of Mozilla Firefox 1.5 provides a subset of the SVG 1.1 specification [MDC05 / MEI06\_1]. Some interesting parts such as SMIL animations are not included, but there are enough possibilities to compensate for the absent features with script programming based on the Document Object Model (DOM).

XML related aspects of Firefox 1.5 and upgrades are also innovative. Client-sided XSL Transformations and the introduction of ECMAScript for XML (E4X) expand and therefore improve practical SVG handling. The popular AJAX methods allow communication with server-side resources like web services out of a SVG document. By using the concept of namespaces, SVG document fragments can now be directly included in XHTML code.

In the following, some techniques are demonstrated using data collected from the author's daughter within the framework of a high school project. In a social studies project she asked 50 persons about organ transplantation. One question was related to the most frequently transplanted organ with these results:

<b>organ(s)</b>	<b>answers in %</b>
kidneys	82
liver	12
heart	4
lungs	2

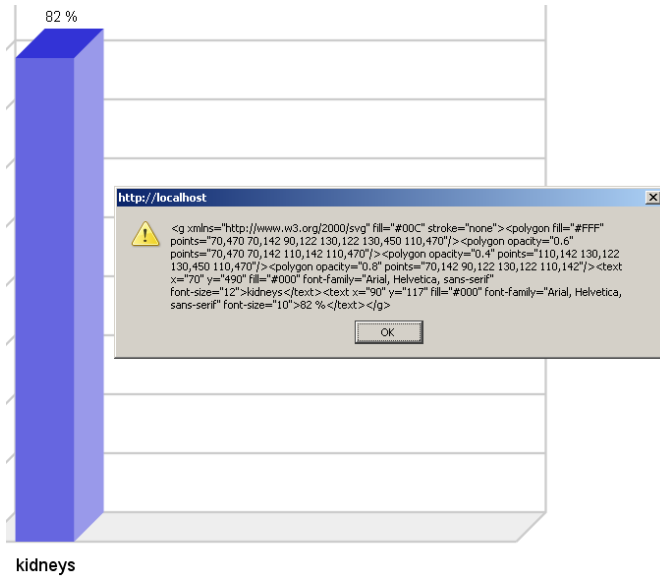
*Table 1. Data collected for visualization purposes.*

The data source from table 1 is organized in a simple XML structure based on a separate DTD with an additional information text and color codes for the graphical representation. Here only the first data set has been selected:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE data SYSTEM "data.dtd">
<data infotext="The most frequently transplanted
organ(s)?">
  <set>
    <value>82</value>
    <descr>kidneys</descr>
    <color>#00C</color>
  </set>
  <!-- more set elements ... -->
</data>
```

The data visualization was performed with a 3D bar chart. SVG describes objects in two dimensions. Drawing individual objects in perspective results in a pseudo 3D look. In this project each bar consists of three polygons placed side by side. These polygons are formatted with one solid color and different opacity values (0.4, 0.6 and 0.8). An ECMAScript implementation of the bars requires some W3C-DOM methods: `createElementNS()`, `setAttributeNS()`, `createTextNode()` and `appendChild()` [W3Coo]. They are intended for the

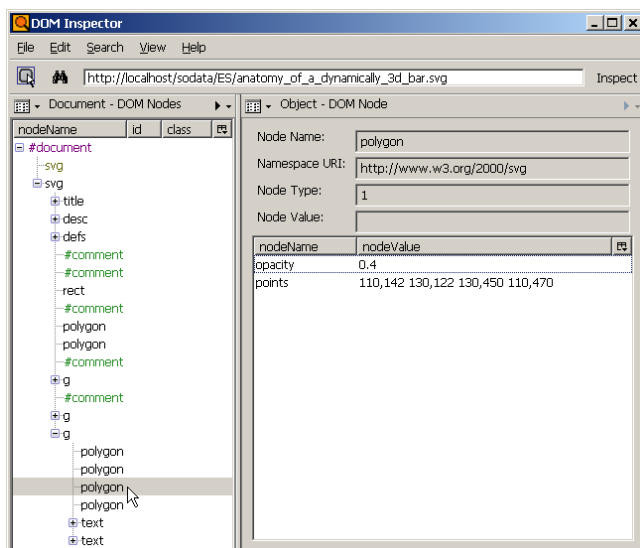
creation of elements (grouped polygons), attributes (point coordinates, fill colors, opacity values, font properties) and descriptive text pieces. Figure 2 is related to a single bar and the code created for it.



anatomy\_of\_a\_bar.png

Figure 2. Anatomy of a single bar with the created code fragment.

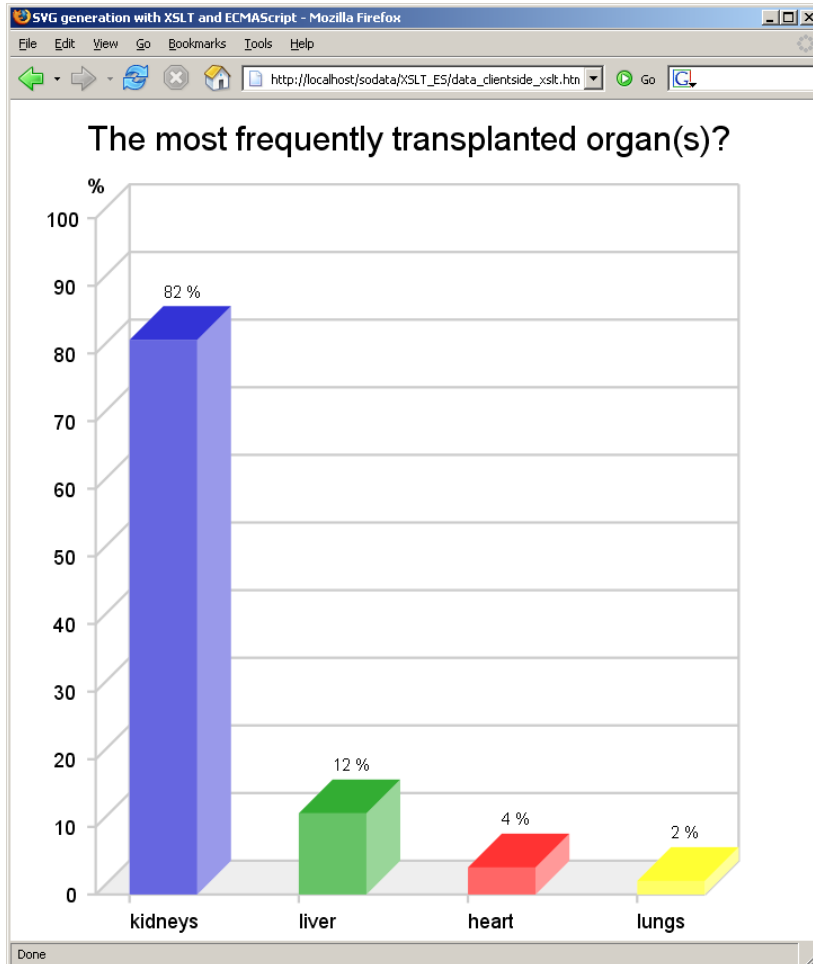
Developers should use the integrated DOM Inspector to control static and dynamic code parts (see figure 3).



ff\_dom\_inspector.png

Figure 3. Code analysis using the Firefox tool DOM Inspector.

A function called `create3DBars()` does the main job four times according to the data from table 1. This function source code is too long for printing, but is also included in the accompanying code package [MEI06\_2]. A view of the complete diagram within the Firefox 1.5 browser is shown in figure 4.



[survey\\_final\\_chart.png](#)

Figure 4. 3D bar chart generated with SVG under Firefox 1.5.

The presentation covers different approaches to the same view of the diagram. Values are hardcoded in the script section or read on demand from the XML source file using the `XMLHttpRequest()` implementation. Other client-side techniques such as `DOMParser()`, `XMLSerializer()`, `XPathEvaluator()`, `XSLTProcessor()` and `E4X` calls are also involved, see [MOZO6]. SVG code inclusion in XHTML documents with scripting access to both is also possible. A PHP-based server-side XSL transformation of the XML data into SVG completes the list of the dynamic SVG generation procedures discussed.

# Conclusion

Although not yet complete, the Firefox 1.5 SVG viewer is certainly a step in the right direction. The use of DOM scripting is the key to more dynamic and interactive graphics. However, to sum up, in combination with XML data sources, AJAX and further advanced technologies, SVG can achieve the status of an application format.

# Bibliography

- [MDCo5] Mozilla Developer Center: SVG in Firefox 1.5, [http://developer.mozilla.org/en/docs/SVG\\_in\\_Firefox\\_1.5](http://developer.mozilla.org/en/docs/SVG_in_Firefox_1.5) (2005).
- [MEIo6\_1] T. Meinike, Fuchsschlaue Vektorgrafiken – Zur SVG-Entwicklung mit Firefox 1.5, Entwickler Magazin 2.2006, p. 132-135 (2006).
- [MEIo6\_2] T. Meinike, XTech 2006 conference material, <http://svglbc.datenverdrahten.de/xtech06/> (2006).
- [MOZo6] Mozilla Developer Center, <http://developer.mozilla.org> and MozillaZine Knowledge Base, [http://kb.mozillazine.org/Knowledge\\_Base](http://kb.mozillazine.org/Knowledge_Base) (2006).
- [W3Coo] W3C, Document Object Model (DOM) Level 2 Core Specification, <http://www.w3.org/TR/DOM-Level-2-Core/> (2000).
- [W3Co1] W3C, Scalable Vector Graphics (SVG) – XML Graphics for the Web, <http://www.w3.org/Graphics/SVG/> (2001).