Performance Evaluation of Web Browsers for Vector Graphics Based Applications

Kazuaki Maeda
College of Business Administration and Information Science, Chubu University
Kasugai, Japan
E-Mail: kaz@isc.chubu.ac.jp

Abstract—This paper describes performance evaluation of six web browsers for SVG based applications. SVG is widely used to draw vector graphics image on web pages. It’s designed as an application of XML so that it’s possible to express various graphical representations with animations using JavaScript programs. Based on the author’s experiences about development of SVG based applications, experiments were designed to measure the performance of web browsers. As a result, two web browsers show poor performance and three browsers show similar performance.

Index Terms—Web browser, SVG, Vector Graphics, Web application

I. INTRODUCTION

There are two different ways in which graphical data are stored and manipulated. They are called raster graphics and vector graphics. Oxford Dictionary of Computer Science [1] has the following definitions:

- **Raster graphics** is the system used for scanned images and for photographs produced by a digital camera. It is suitable for complex pictures with a large number of different colors and shades. The image is stored as chunks of data, each representing the properties of a small element of the picture.
- **Vector graphics** is stored and manipulated as mathematical formulae producing lines, geometrical figures, and curves. It is used for diagrams, graphs, flow charts, engineering drawings, et al.

The web browsers on early days displayed only textual information. After that, `<img>` tag was developed to embed raster graphics in the web pages [2]. Most of latest web browsers support SVG [3], which stands for Scalable Vector Graphics, using lines of code to define paths, shapes, fills, colors, et al. Moreover, we can embed scripts in JavaScript to create special effects. The web sites ([4] and [5]) show all modern web browsers support SVG.

Usage share of web browsers has changed in the last decade. Fig. 1 is a graph downloaded from wikipedia.org¹, and modified two paths by the author using a text editor. The figure shows the usage share of web browsers between 2009 and 2015. There is no doubt about the fact that Internet Explorer lost the share. On the other hand, the usage share of Chrome has grown significantly.


The author joined in a software development project for commercial applications, and now the applications with vector graphics are under construction. The experiences indicate that different web browsers have different characteristics to draw the vector graphics. We tried to find research papers and articles on the Web about the issues, but there are no research papers to clarify the difference of web browsers to the best of my knowledge. Therefore, major six web browsers were chosen, benchmark programs in JavaScript were developed, and experiments to measure the performance were done.

In section 2, we will overview features of SVG, and will discuss experiments and the performance evaluation in section 3. Finally, the paper will be summarized.

Fig. 1. Usage share of web browsers

```xml
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink" width="...
     <path style="stroke-width:5;stroke-dasharray:4 8; stroke:rgb(0%,75.294118%,0%);stroke-opacity:1.../>
```

Fig. 2. A part of SVG description in Fig. 1

In section 2, we will overview features of SVG, and will discuss experiments and the performance evaluation in section 3. Finally, the paper will be summarized.
II. OVERVIEW OF SVG

SVG is an application of XML, and W3C released the proposal as a standard SVG1.0 in 2011 [6]. After that, W3C released next version of the proposals as SVG1.1 in 2011 [3].

The graph in Fig. 1 is a vector graphics image written in SVG. It is a plain text file so that two paths could be modified as shown in Fig. 2 by the author. A simple text editor, e.g. vim, was used to emphasize the changes of Internet Explorer and Chrome. One of the modifications is to change the value of the “stroke-width” attribute to change the width of the path to 5 pixels, and the other is to add a “stroke-dasharray” attribute to change the path to dotted one. It was not a difficult task due to the XML based text representation.

Fig. 3 shows another example of SVG. When the web browser reads SVG description on the lefthand side of the figure, it displays a circle, a line and a text string shown in the righthand side. SVG is not only for showing on the web browser, but also graphic tools read and edit the data. Recently it plays a standard role to exchange graphics data between applications.

From the author’s experience, there are three ways to apply SVG:

1) to draw static images,
2) to play animations, and
3) to draw images and update them dynamically.

“To draw static images” means we draw the images using graphic tools such as Illustrator, and store the images to the file using SVG.

For example, Fig. 4 shows a part of railway map around Tokyo, Japan². This map was released under public domain, so the author had a plan to use as a sample of SVG applications, and to show how to handle SVG images using JavaScript. But the SVG file generated by Illustrator includes very complicated elements in many lines of code (about 9,000 lines of code). It is not easy to handle them using JavaScript programs.

One of examples “to play animation” is to use an animateTransform element. Using the animateTransform element, we can move and rotate a specified element in SVG. Fig. 5 shows a newton’s cradle. In this animation, I created a plain text file written in only SVG. The leftmost circle with a line is rotated clockwise by 30 degrees. After that, the rightmost circle with a line is rotated counterclockwise by 30 degrees.

An example “to draw images and update them” is a clock, developed by the author shown in Fig. 6. It updates all hands every second to measure the performance. The other example is a bar chart to update every second shown in Fig. 7.

III. EXPERIMENT FOR PERFORMANCE COMPARISON

The experience of SVG based application development indicates that different web browsers have different performance to draw the user interface. We tried to search research papers and articles on the web about the performance issues, but there are no research papers to clarify the difference of web browsers to the best of my knowledge. Therefore, major six web browsers, those are Chrome, Firefox, Vivaldi, Opera, Internet Explorer (hereinafter, called IE) and Edge, were chosen, and some benchmark programs in JavaScript were developed by the author. After that, some experiments to measure the performance were done.

Three types of laptop PCs were prepared to examine the performance of web browsers to display SVG images. The PCs were referred as PC2017, PC2016 and PC2014 because they were bought on year 2017, 2016 and 2014. An experiment was designed using the following hardware and software:

²https://note.openvista.jp/2014/svg-rail-map
Firstly, HDD and SDD were initialized, and Windows 10 was installed on each PC, but anti-virus software was not installed because the software sometimes have an effect on the performance. Moreover, the following six web browsers were installed on each PC.

- Google Chrome, Version 70.0.3538.110
- Mozilla Firefox, Version 63.0.3
- Vivaldi, Version 2.1.1337.51
- Opera, Version 56.0.3051.116
- IE, Version 11.1418.15063.0
- Microsoft Edge, Version 40.15063.674.0

Web pages including JavaScript programs were displayed, and the execution time was calculated.

Three JavaScript programs were created for performance comparison.

- Clock based on SVG and clock based on D3.js [7] shown in Fig. 6.

The experiment was designed as follows:

- Nine clocks were placed on one web page. For each clock, the hour hand and the minute hand are rotated using two generated random numbers.
- 100 iterations were executed for measuring.
- The execution time was calculated as shown in Fig. 8.
- The web page was updated on each 3 seconds, totally updated for 100 times, and the average of the execution time was calculated.

To check the difference of implementation, two types of programs, which are JavaScript native version (called Clock-SVG) and D3.js API based version (called Clock-D3), were prepared.

- Bar chart
  The experiment was designed as follows:

  var date1 = new Date();
  for(var cnt=0; cnt < 100; cnt++) {
      for (var i in svgClockList) {
          updateClock(svgClockList[i]);
      }
  }
  var date2 = new Date();
  var t = date2.getTime() - date1.getTime();

Fig. 8. Snippet of JavaScript program to calculate the execution time

- Bar charts shown in Fig. 7 were updated using generated 50 random numbers
- Each random number was inserted in each bar chart from left side.
- Old data were deleted from right side to keep 100 data on each chart.
- Only 3 bar charts were placed on one web page because of the heavy workloads.
- 50 iterations were executed for measuring.
- The execution time was measured shown in Fig. 8.
- The web page was updated on each 3 seconds, totally updated for 100 times, and the average of the execution time was calculated.

The average execution time are given in TABLE I. From the result, IE and Edge show poor performance. Moreover, Chrome, Vivaldi and Opera show very similar value. One reason of this is that they include same implementation code. I downloaded and checked source code of Vivaldi so that the code includes a part of Chromium [9]. Moreover Wikipedia.org writes “Opera is a Chromium-based browser” [8].

Before the experiments shown in Table I, multiple SVG images were placed on one web page, and updated every second. But IE and Edge could not update the images accurately so that the experiments were performed to update every 3 seconds.

The result of the experiment is almost same as my experience when I developed vector graphics based web applications.

<table>
<thead>
<tr>
<th>PC 2017</th>
<th>Clock-SVG</th>
<th>Clock-D3</th>
<th>Bar chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>17.77</td>
<td>29.79</td>
<td>20.09</td>
</tr>
<tr>
<td>Firefox</td>
<td>22.03</td>
<td>31.13</td>
<td>21.88</td>
</tr>
<tr>
<td></td>
<td>108.86</td>
<td>233.64</td>
<td>110.56</td>
</tr>
<tr>
<td></td>
<td>97.88</td>
<td>433.12</td>
<td>360.06</td>
</tr>
<tr>
<td>Opera</td>
<td>13.78</td>
<td>16.24</td>
<td>15.46</td>
</tr>
<tr>
<td></td>
<td>15.20</td>
<td>16.07</td>
<td>15.29</td>
</tr>
<tr>
<td></td>
<td>108.08</td>
<td>197.7</td>
<td>111.30</td>
</tr>
<tr>
<td></td>
<td>103.08</td>
<td>103.42</td>
<td>103.42</td>
</tr>
<tr>
<td>Edge</td>
<td>28.31</td>
<td>37.40</td>
<td>33.80</td>
</tr>
<tr>
<td></td>
<td>28.58</td>
<td>41.46</td>
<td>35.94</td>
</tr>
<tr>
<td></td>
<td>171.42</td>
<td>315.64</td>
<td>163.10</td>
</tr>
<tr>
<td></td>
<td>164.80</td>
<td>510.70</td>
<td>586.94</td>
</tr>
</tbody>
</table>
IV. SUMMARY

This paper described performance evaluation of six web browsers for SVG based applications. Based on the author’s experiences about development of SVG based applications, experiments were designed to measure the performance of web browsers. As a result, two web browsers show poor performance and three browsers show similar performance.

Corporate Vice President in Microsoft announced that next version of Edge will change drastically [10]. It will become a Chromium based browser. The reason is that Edge should observe web standards and improve the performance. This research should continue because this year is a good timing to evaluate enhancements of web browsers. The results will be published in a future paper.

Acknowledgement

I gratefully acknowledge the financial support provided by The Hibi Science Foundation.

REFERENCES