

ベクター画像活用のための Web ブラウザの検討

A Study on Web Browsers for Vector Graphics Based Applications

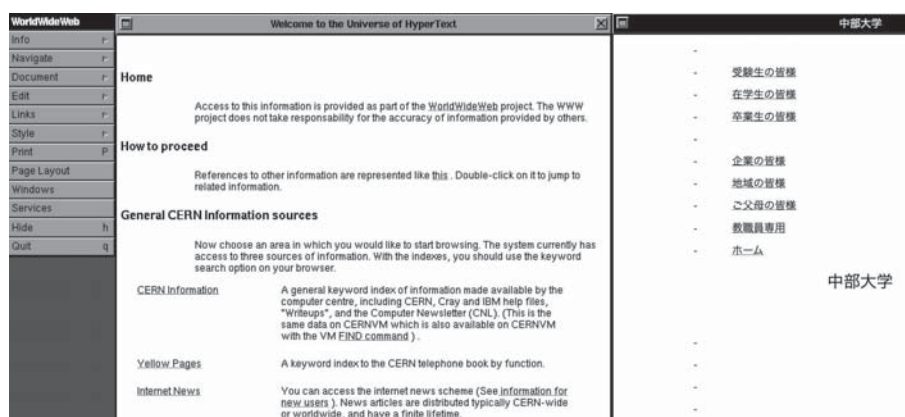
前 田 和 昭*

Kazuaki MAEDA

1 はじめに

インターネットの爆発的な普及は、Web ページを記述するための言語 HTML とそれを表示する Web ブラウザがきっかけとなって始まった。世界初の Web ブラウザ WorldWideWeb は、1990 年に CERN (European Organization for Nuclear Research) で開発された [1]。この WorldWideWeb は、Tim Berners-Lee が書いた提案書 [2] にもとづいて、彼自身が実装したアプリケーションである。それから 30 年経ったことを記念して、当初と同じ見た目・使い勝手を実装するプロジェクトが立ち上がり、CERN 2019 WorldWideWeb Rebuild として公開されている [3]。

図 1 CERN 2019 WorldWideWeb Rebuild の実行画面の一部



* 経営総合学科 — ソフトウェア工学

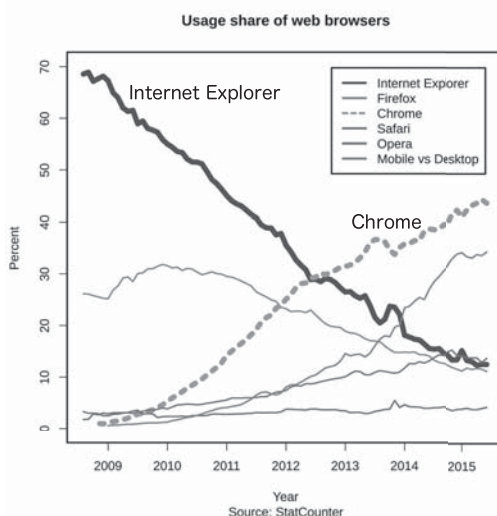
Department of Management Synthesis — Software Engineering

WorldWideWeb を起動させてから、中部大学トップページを開いたときのスクリーンショットを図 1 に示す。中部大学のトップページ部分には文字しか表示されていないことから分かるように、当時は Web ページに画像を埋め込むことができなかった。最近の Web ブラウザでは、画像だけでなく音声や動画を扱うことができる。さらには、動的なコンテンツを提供するためにプログラムを埋め込む技術として Flash や Java アプレットなどが使えるように機能拡張が行われてきた。その結果、初期の WorldWideWeb に比べて最新の Web ブラウザのサイズは膨れ上がり、巨大なアプリケーションになった。それとともに、この 30 年間で各 Web ブラウザの市場シェアは大きく変わっていった。

2009 年から 2015 年までの市場シェア変動グラフを Wikipedia から抜粋し¹⁾、加工して図 2 に示す。白黒印刷で比較できるように、元画像ではカラー表示だったグラフ内の Internet Explorer の線を太く、Chrome の線を太い破線になるように、テキストエディタを使って直接データを書き換えた。2009 年頃には 70% 近くを占めていた Internet Explorer が 2015 年頃には 10% ぐらいまで減少し、その一方 Google Chrome が急速に普及したことが分かる。

2019 年 1 月の Web ブラウザ市場シェアを、StatCounter 社が提供するデータ [4] から抜粋して表 1 に示す。この表では、日本・世界・米国・欧州の地域ごとに Web ブラウザ市場シェアの違いが分かるようにした。日本では Internet Explorer の割合が他と比べて高めであり、また、欧州では Firefox と Opera が他と比べて高めになっ

図 2 2009 年以降の Web ブラウザ市場シェア



1) [https://ja.wikipedia.org/wiki/ブラウザ戦争#/media/File:Usage_share_of_web_browsers_\(Source_StatCounter\).svg](https://ja.wikipedia.org/wiki/ブラウザ戦争#/media/File:Usage_share_of_web_browsers_(Source_StatCounter).svg)

表 1 デスクトップ PC での Web ブラウザ市場シェア

(2019 年 1 月 各地域での割合：単位%)

Web ブラウザ	日本	世界	米国	欧州
Google Chrome	58.46	70.88	63.19	63.17
Internet Explorer	15.19	5.74	10.1	6.08
Mozilla Firefox	9.47	9.5	8.43	13.58
Safari	8.3	5.15	8.31	6.52
Microsoft Edge	6.69	4.14	8.6	5.4
Opera	0.84	2.37	0.82	3.49

図 3 Usage_share_of_web_browsers_(Source_StatCounter).svg の内容

```

<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
width="504pt" height="504pt" viewBox="0 0 504 504" version="1.1">
<defs>
<g>
<symbol overflow="visible" id="glyph0-0">
<path style="stroke:none;" d="M 1.203125 -8.25 L 7.796875 -8.25 L 7.796875 0 L 1
.203125 0 Z M 1.640625 -7.8125 L 1.640625 -0.453125 L 7.34375 -0.453125 L 7.3437
5 -7.8125 Z M 1.640625 -7.8125 "/>
</symbol>
<symbol overflow="visible" id="glyph0-1">
<path style="stroke:none;" d="M 0.609375 0 L 0.609375 -0.75 C 0.804688 -1.207031
1.046875 -1.609375 1.328125 -1.953125 C 1.617188 -2.304688 1.921875 -2.625 2.23
Usage_share_of_web_browsers_(Source_StatCounter).svg

```

ている。地域によって細かな違いがあるものの、Chrome は 50 % 以上を占め、Internet Explore は 10 % 前後であることが分かる。

最近の Web ブラウザには、SVG (Scalable Vector Graphics) による画像を扱う機能が実装され、2 次元グラフィックスを Web ページ上に描くために広く使われている。図 3 に示すように、図 2 のグラフは SVG で提供されている。そのため、テキストエディタを使ってデータを書き換えることが可能であった。SVG は、ベクター画像を XML で記述するように設計され、プログラムから操作することで多様な表現が可能となる。本稿では、SVG を使ってユーザインタフェースを構築してきた経験を踏まえ、SVG の概要を説明し、品質の良いユーザインタフェースを上げるために行った性能評価実験について述べる²⁾。

2) 本稿は、発表済みの論文 [5] の内容に解説を追加して作り直した研究ノートである。

2 XML と画像

WWW の普及に貢献した HTML は、文書中に印刷命令を表現するタグを埋め込むために設計された。これに対して XML は、タグを使ってデータに意味を付記できるようにしたマークアップ言語である。コンピュータでの処理やデータベースに格納するなど、ビジネスで広く活用できるように工夫がなされている。例えば、文献データや注文伝票などを記述したものがそれにあたり、アプリケーション間でデータを交換したり、またデータベースからデータを取り出して他の製品で活用する場面など多くの応用分野がある。

XML データを処理する場合、データの容量が膨れあがることに注意が必要である。例えば、コンピュータ科学の論文文献データを提供する DBLP [6] の中から筆者執筆による論文データを抜粋して図 4 に示す。XML は、人間が読むことができ、また、プラットフォーム非依存にするため、タグによって構造化された文字データを使って表現されている。そのため、バイナリ形式や簡易テキスト形式に比べて、3 倍から 20 倍もの容量が必要だといわれている [7]。DBLP を XML データとして書き出したところ約 2.5GB の容量となった (2019 年 2 月 24 日現在)。

図 4 DBLP 論文データの一部

```
<inproceedings mdate="2017-05-21" key="conf/bionetics/Maeda14">
<author>Kazuaki Maeda</author>
<title>Parser Development with an Internal Domain-Specific Language and
an Aspect Weaver.</title>
<year>2014</year>
<booktitle>BICT</booktitle>
<ee>https://doi.org/10.4108/icst.bict.2014.257977</ee>
<crossref>conf/bionetics/2014</crossref>
<url>db/conf/bionetics/bict2014.html#Maeda14</url>
</inproceedings>
```

コンピュータ科学辞典 [8] には、画像について以下のように説明されている。

(以下、Oxford Dictionary of Computer Science[8] より抜粋)

Computer graphics Computer graphics is the creation of, manipulation of, analysis of, and interaction with pictorial representations of objects and data using computers. There are two quite different ways in which the information is stored and manipulated, usually referred to as raster graphics and vector graphics.

Raster graphics This 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 (e.g. color, tint, transparency) of a small element of the picture.

Vector graphics This system is used for diagrams, graphs, flow charts, engineering drawings, etc. It is suitable for relatively simple line drawings. The image is stored and manipulated as mathematical formulae producing lines geometrical figures, and curves (e.g. Bezier curves).

本稿では、Raster graphics のことを「ラスター画像」、Vector graphics のことを「ベクター画像」と記すことにする。図 5 左側にラスター画像 (PNG 形式) を、右側にベクター画像 (AI 形式) を表示したときの様子を示す。これらは、同じ絵を描いてから各画像を 10 倍に拡大したときの表示結果を並べたものである。左側のラスター画像はギザギザが目立つのに対して、右側のベクター画像は滑らかな曲線になっている。これらの差は、ピクセルでデータを持つか、図形命令でデータを持つかの違いに由来する。

図 5 ラスター画像(左)とベクター画像(右)



図 6 円・線分・テキストを SVG で記述した例

```
<svg width='400' height='200'>
  <circle cx='200' cy='100' r='80' fill='none'></circle>
  <line x1='200' y1='20' x2='200' y2='180'></line>
  <text x='40' y='160'>晴れ</text>
</svg>
```

図 7 Web ブラウザでの表示結果



最新の Web ブラウザでベクター画像を扱うには SVG を使うことが多い。SVG は、ベクター画像を表現するための XML データ形式であり、W3C により仕様が作成されている。SVG 1.0 が 2001 年に、SVG 1.1 が 2011 年に勧告として公開された [9, 10]。開発者らによって保守されている Web サイト [11, 12] によれば、代表的な Web ブラウザには SVG 1.1 を満たす機能が実装済みとなっている。

SVG を説明するために、SVG による簡易な記述を図 6 に示す。この記述を Web ブラウザで読み込むと、図 7 に示す円・線分・テキストの組み合わせが表示される。SVG は、Web ブラウザで表示されるためだけではなく、グラフィックスツールに読み込んで加工し保存できる。最近では、グラフィックスツール間でデータを交換するためにも使われている。

3 SVG の 3 通りの使い方

筆者のこれまでの経験から、SVG は

1. 静的に使う
2. アニメーションに使う
3. 動的に使う

の 3 通りの場合があると考える。「静的に使う」ときは、Adobe 社 Illustrator などのグラフィックスツールを使って絵を描き、SVG で保存する場合である。例えば、図 8 に、橋本氏が描いた東京近辺の鉄道路線図の一部を示す [13]。この路線図はパブリックドメインで公開されているため、SVG を説明するためのデモや、JavaScript プログラムで操作する素材として使うことを検討した。しかし、Adobe 社 Illustrator を使って丹念に生成されたベクター画像を SVG で保存したファイルは、ファイル 1 つが約 9,000 行の記述からなっていた。アプリケーションが生成する SVG ファイルは複雑になることが多く、これを JavaScript プログラムから操作することは困難と判断した。

「アニメーションに使う」ときは、例えば `animateTransform` 要素を使い、SVG で記述された特定の要素に対して、移動や回転、変化の始点・終点、変化の開始・終了

図 8 鉄道路線図の一部



図 9 アニメーションの例

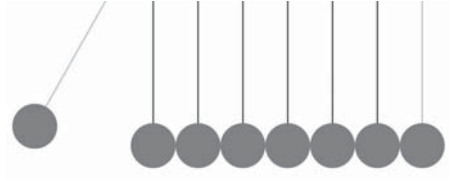
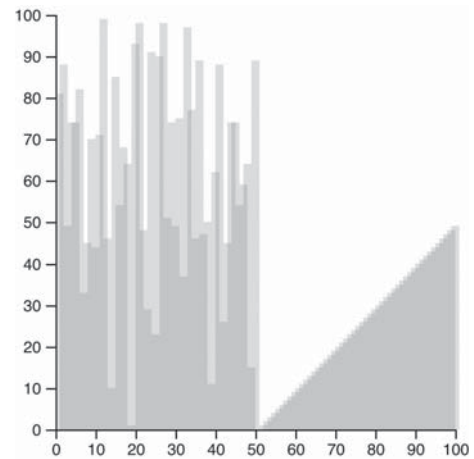


図 10 毎秒ごとに針の表示を更新する時計



図 11 毎秒データを更新する棒グラフ



タイミングを指定して、動きを表現する場合である。図 9 に、書籍 [14] で紹介されているニュートンのゆりかごの例を示す³⁾。このアニメーションでは、SVG で規定された XML のみで動きを記述し、左端の円とそれに接続する線分と、右端の円とそれに接続する線分を左右に 30 度ずつタイミングを合わせて交互に回転させている。

3 つめの「動的に使う」ときは、定期的（例えば毎秒ごと、または、ユーザーの要求時）に表示を変化させる場合である。性能評価のために筆者が作成した、毎秒ごとに針の表示を更新する時計の例を図 10 に、毎秒データを更新する棒グラフの例を図 11 に示す⁴⁾。

次節では、3 つめの「動的に使う」ときのみを対象とし、SVG で作成されたベクター画像を、決められたタイミングで更新していく JavaScript プログラムを使って性能評

3) 説明のために筆者が修正し、付録で示す Web サイトに保管した。Web ブラウザで表示すれば動きを確認できる。

4) 時計の例と棒グラフの例ともに、付録で示す Web サイトを表示すれば動きを確認できる。

価を行った結果について報告する。

4 Web ブラウザの性能評価

これまで SVG を使ってユーザインタフェースを描いてきた経験では、Web ブラウザを変えると動作速度が異なることを感じていた。この動作速度について調査をしたものの、SVG で作られたベクター画像を表示する場面で、Web ブラウザによる違いを示した研究論文または解説記事が見つからなかった。そこで、各 Web ブラウザでの実行時間を計測するための JavaScript プログラムを作成し、代表的な Web ブラウザを使って性能評価実験を実施することにした。

4.1 SVG を活用する JavaScript プログラムの Web ブラウザ毎の性能評価

実験を行うハードウェアとして、以下の仕様のノートブック PC を 3 台準備した。これらの購入時期が 2017 年、2016 年、2014 年であることから、それぞれ PC 2017、PC 2016、PC 2014 と名付けることにする。まずは、全ての PC に内蔵された HDD または SSD を初期化した後、Windows 10 をインストールし、その後ウィルス対策ソフトウェアなど実験に関係ないソフトウェアは一切インストールしないことにした。

- ・ PC 2017 - Intel Core i7 (7820HQ) 2.90GHz, 主記憶 : 16GB, SSD : 2TB, Windows 10 Pro, Version 1703
- ・ PC 2016 - Intel Core i5 (6200U) 2.30GHz, 主記憶 : 8GB, HDD : 500GB, Windows 10 Home, Version 1803
- ・ PC 2014 - Intel Core i5 (4308U) 2.80GHz, 主記憶 : 16GB, HDD : 1TB, Windows 10 Pro, Version 1803

さらに、以下に示す代表的な 6 つの Web ブラウザをインストールして、それらの上で筆者が作成した SVG データを含む Web ページを表示し JavaScript プログラムを動かして、実行時間を測定した。

- ・ 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
- ・ Internet Explorer (IE と略記する) , Version 11.1418.15063.0
- ・ Microsoft Edge, Version 40.15063.674.0

性能評価のために作成した JavaScript プログラムは以下の 3 種類である。

- ・時計 (SVG を主) と時計 (D3 を主)

2つの乱数を生成し、その値にしたがって図 10 で示す時計の長針・短針を回転させる。この時計を同じ Web ページ上に 9 個配置して、長針・短針の回転処理を 100 回行い、プログラム実行開始前と終了後の時間の差から実行時間を計算する。さらに、3 秒に 1 回 Web ページ全体を更新する処理を 100 回実行して平均値を求める。実装の違いによる性能を評価したかったので、ベクター画像の大部分を SVG で描くことで JavaScript のプログラム量を最小限にしたもの（時計 (SVG を主) と記す) と、SVG による記述を最小限にして大部分を D3.js [15] の API を使って JavaScript で記述したもの（時計 (D3 を主) と記す) の 2 つを準備した。

- ・棒グラフ

乱数 50 個を生成し、図 11 で示す棒グラフを更新する。各データを左側（棒グラフの原点）から順番に挿入し、表示されているデータを 100 個に限定するため、右側（棒グラフの x 座標値 100）から削除している。この棒グラフを同じ Web ページ上に 3 個配置して、データの更新処理を 50 回行い、プログラム実行開始前と終了後の時間の差から実行時間を計算する。さらに、3 秒に 1 回 Web ページ全体を更新する処理を 100 回実行して平均値を求める。

4.2 性能評価の結果

性能評価実験で求めることができた実行時間を表 2 に示す。この結果から、IE と Edge のときが他に比べて値が大きいことが分かる。また、Chrome, Vivaldi, Opera

表 2 各 Web ブラウザ毎の JavaScript プログラム実行時間

(単位：ミリ秒)

		Chrome	Firefox	Vivaldi	Opera	IE	Edge
PC 2017	時計 (SVG を主)	17.77	29.79	18.54	20.09	102.19	126.10
	時計 (D3 を主)	22.03	31.13	21.88	29.76	108.05	147.66
	棒グラフ	108.86	233.64	110.56	97.88	433.12	360.06
PC 2016	時計 (SVG を主)	13.78	16.24	15.46	15.13	90.91	103.42
	時計 (D3 を主)	15.20	16.07	15.29	15.85	84.74	100.49
	棒グラフ	108.08	197.7	111.30	103.08	447.88	517.08
PC 2014	時計 (SVG を主)	28.31	37.40	33.80	33.23	97.73	126.10
	時計 (D3 を主)	28.58	41.46	35.94	40.60	100.12	132.06
	棒グラフ	171.42	315.64	163.10	164.80	510.70	586.94

の実行時間が似た値を示していることは、Web ブラウザでの表示機能を実装しているレンダリングエンジンが同じであるという事実を示しているものと思われる。表 2 で示した実験結果を求める前段階で、同じ Web ページに複数の SVG データを置いてから、1 秒に 1 回の頻度で表示を更新していた。しかし、Web ブラウザによっては更新のタイミングに追いつくことができず異常な値を示していたため、3 秒に 1 回の頻度で実験を行うことになった。これらから得られた Web ブラウザ間での優劣は、筆者が経験したことに近い感触を得ている。

5 おわりに

本稿では、SVG を使ってユーザインタフェースを構築してきた経験を踏まえ、SVG の概要を説明し、SVG に基づくベクター画像を表示するときの各 Web ブラウザ毎の性能評価について述べた。現段階で示することができる性能評価実験の結果は、各 Web ブラウザの限定した特徴を示しているにすぎない。今後さらなる改良と実験を続けていく予定である。

参考文献

- [1] James Gillies and Robert Cailliau, How the Web Was Born, Oxford University Press (2000).
- [2] Tim Berners-Lee, Information Management: A Proposal (1989), <https://www.w3.org/History/1989/proposal.html> (2019 年 2 月 26 日参照).
- [3] CERN 2019 WorldWideWeb Rebuild, <https://worldwideweb.cern.ch/> (2019 年 2 月 26 日参照).
- [4] StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share, <http://gs.statcounter.com/> (2019 年 2 月 26 日参照).
- [5] 前田和昭, SVG によるユーザインタフェース構築の検討, 第 14 回情報システム学会全国大会・研究発表大会 (2018).
- [6] dblp: computer science bibliography, <https://dblp.uni-trier.de/> (2019 年 2 月 26 日参照).
- [7] Zap Think, The “Pros and Cons” of XML, Zap Think Research Report (2001).
- [8] Oxford Dictionary of Computer Science 7th edition, Oxford University Press (2016).
- [9] Scalable Vector Graphics (SVG) 1.0 Specification, <https://www.w3.org/TR/2001/REC-SVG-20010904/> (2019 年 2 月 26 日参照).
- [10] Scalable Vector Graphics (SVG) 1.1 (Second Edition), <https://www.w3.org/TR/SVG11/> (2019 年 2 月 26 日参照).
- [11] Can I use... Support tables for HTML5, CSS3, etc, <https://caniuse.com/#search=SVG> (2019 年 2 月 26 日参照).
- [12] HTML svg tag, https://www.w3schools.com/tags/tag_svg.asp (2019 年 2 月 26 日参照).
- [13] 東京の鉄道路線図 SVG を作りました&パブリックドメインで配布します - Liner Note, <https://>

note.openvista.jp/2014/svg-rail-map (2019 年 2 月 26 日参照).

[14] Damian Dawber, *Learning Raphael JS Vector Graphics*, Packt Publishing (2013).

[15] D3.js - Data-Driven Documents, <https://d3js.org/> (2019 年 2 月 26 日参照).

付録：本稿で紹介した SVG 記述例の保管場所

- ・ 図 9：ニュートンのゆりかご：<http://jsfiddle.net/kmaeda/c5uqb9xt/4/>
- ・ 図 10：乱数で長針・短針を更新する時計：<http://jsfiddle.net/kmaeda/2d7a0kqs/3/>
- ・ 図 11：乱数データを挿入していく棒グラフ：<http://jsfiddle.net/kmaeda/bjr23twk/4/>
- ・ 大部分を SVG で書いた時計の文字盤：<http://jsfiddle.net/kmaeda/ksq1oc6v/6/>
- ・ 大部分を D3 で描画する時計の文字盤：<http://jsfiddle.net/kmaeda/v5utg6rq/6/>

付録 Impress.js を使ったスライド記述の一部

本論文は、クリエイティブ・コモンズ 表示—継承ライセンスの下で利用可能である。