Ermias Hailemicheal

Top Tools for Data Visualization in Web Development

A Performance first approach

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineering

Thesis

August 2016

Author(s) Title Number of Pages Date	Ermias Hailemicheal Top tools for data visualization for web development 40 pages November 16
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Digital Media, Mobile Programming
Instructor(s)	Ilkka Kylmäniemi, Senior Lecturer

The purpose of this final year project was to deliver a visualisation based solution for the game analytics company, Game Refinery. The company's platform presents a handful of reviews for games based on features with an additional chart that shows the rank of games in stores. Due to the slowness of the charts on the platform, multiple types of performance testing mechanisms were implemented to optimize the charts and the ways the system runs them.

The focal point of the thesis is testing the performance of different charting libraries based on their user friendliness, features, functionalities, customisability, architecture, scalability, compatibility and performance. The project also put emphasis on performance optimization in web development and proper implementation of data visualization in web applications.

Keywords	Data visualization, performance, charts, optimization

Contents

List of abbreviations

SVG	Scalable Vector Graphics
DataViz	Data Visualization
ms	Milliseconds
DOM	Document Object Model
HTML	Hyper Text Markup Language
CSS	Cascaded Style Sheet
XML	Extensible Markup Language
D3	Data Driven Documents
JS	JavaScript
kB	Kilobytes
IE	Internet Explorer
DPI	Dots Per Pixel
JIT	Just in time (compiler)

1 Introduction

Several studies show that presenting pieces of information in the form of visualization has set its root way before the graphs and pie charts came into play in the late 17th century. The popularity and practice of data visualization is evolving rapidly, due to the excessively growing big data industry. Visualization and data are two broad subject matters that can define and function well if they are used individually but they can be utilized to present a completely simplified and easy to read presentation of an abstract statistics with the power boost of great web performance. Visualization is what makes numbers from a data analysis come to life through the uses of shapes and colours. Although a good data visualization is a channel that presents more than an image, it is performance that brings fluidity to the visualization. Good performance leads to better user experience while a slow system does the opposite.

Data Visualization is a crucial tool to condense and present a large dataset which might take multiple pages of reading time with in just a visualization. Data Visualization also gives insights from different point of view about a particular area of study and helps viewers to comprehend the bigger picture. Moreover, it eases the flow of a comparison made within a particular data or multiple datasets and can be used as a link to fill the void between patterns.

The project is a test done within a game analytics company named Game Refinery. The company's platform loads 50 games per a web page and each game has a Sparkline chart attached with it in the thumbnail which it is presented on. The Sparkline shows the time series of the changes that the game is making based on multiple factors such as popularity, sell increase and decrease, special features, etc. The presentation is by the ranking the games have in ascending order. The overload of all those Sparkline charts in a single page slows the platform from loading fast. The formerly used data visualization tool, D3 is a powerful tool; however, it could not solve the current issue.

The literature review chapter goes through the basic theory of data visualization and discusses why it is important to visualize data. Also other topics such as JavaScript and performance optimization are discussed. The method section introduces the libraries and presented the outcome of the tests.

2 Literature Review

2.1 Introduction to Data Visualization

The human brain has the capacity to process visualizations efficiently and at a high speed compared with, for example, reading a text. Studies show that visuals are processed 60,000 times faster than text and almost ninety percent of the information received by our brain are visuals. It is a proven fact that people tend to respond positively for advertisements with visuals in them.

The American physician and author Michael Stephen Palmer, M.D. compares the importance of using data properly with oil, he states that if it is not preserved and refined well, it cannot be as valuable as it is intended to be. He concludes by emphasizing that it must taking down to pieces and scrutinized to get the best quality of it. The process of breaking down a data and analysing it, can be regarded similar to turning a raw material to food and cooking it to become the delicious meal that fulfils the desire and satisfaction of those who taste it. [1]

The big rise of the internet and its daily evolvement is certainly making its users less settled on reading a complex website with too many words on it. On the other hand, the infographic industry is bringing people a much easier approach even to the most impenetrable data collections.

If the execution process of data visualization is not started by a proper planning, it might lead to some difficulties as listed below:

- Ending up with too much of unneeded information which leads to a huge gap between the intended intended implementation of the data and its bloated and hard to follow version. Moreover, it makes the filtering process filled with difficulties. This approach might lead misunderstanding the storytelling of the data visualization as well.

- In the book Visualizing, Ben Fry states that there are 7 steps that should be taken to properly understand the collected data [2]
    - Acquire
        - Getting the data
    - Parse

        - Alter the form and keeping it in order
    - Filter
        - discharge unneeded information from being part of the visualization
    - Mine
        - Scrutinize the mathematical aspect of the data or get the proper calculation to get the visualization right
    - Represent
        - For a start use a simple chart to present the data
    - Refine
        - Enhance the visualization
    - Making it Interactive
- Data evolves through time. Constantly updating it is necessary because it never stays the same. [2]
- A data collected yesterday will become part of history, but visualization brings history back to life if presented well.

Although Data Visualization is all about visualizing data, there are some steps to be taken before starting to implement the visuals. Visualization has lifecycles which are similar to building a Software and it should not be defined by the tools or the libraries that it is going to be built with, but rather by the story that the data is going to tell and the analysis done on the data and the architecture designing of the system. If those steps are well executed, implementation would be simpler.

There are some factors to consider before deciding which tool to use, some of the crucial ones are listed in the next section.

2.2 Criteria to get the best tools in the market

As data visualization has been an important tool to manipulate and present data reports for almost all the business ventures including printing medias, a handful reasons can be given as a criterion for getting an efficient tool.

Who is it for?

Before answering all the questions that come with the data that is going to be presented, there must first of all be a clear direction to whom is the presentation for and secondly the user must be identified. Knowing the audience of the information can sort the prediction of picturing the data from so many different angles, rather than view it from the perspective of the insinuated user. Finally, we must know how much time would the user of the data is going to spend on visualizing it.

Data visualization is the presentation of data which can be understood and viewed by anyone, but there are methods or implementation on tools which are best suited for certain types of users based on the consumption of time which the user is willing to spend on viewing the data visualization tool; the main ones can be classified into three, [3]

I. **Infographics** - can be a better approach to the data presentation if the user is going to take a quick glance over the data. This is mostly for business executives who do not have much time for detailed information but for checking out changes and the result of those changes over some period of time. They only have time for the presentation of the data. The visualization is static. [3]

II. **Dashboards** - this is a good way to present data for users who have some time to play with the data to scrutinize it . These users need to understand the data and perform simple functionalities such as filtering, extents of the results through time, assessing the findings from different aspects, etc. with the options provided by the service. The users of Dashboards do not have to be well versed with the technical aspects of data or technologies that comes with the services but they just have to have the knowledge which is enough to get them around to interact with the data. [3]

III. **Analytical tools and applications** - Analysts, data experts, developers and designers who tend to have interests to dig deeper and come up with an in-

terpretation of their own are the primary users of this tools. As it is shown in figure 1, these users are expert in the field of data processing and their involvement is spread out from the starting point of designing up to the end product. They know their way around in manipulating and presenting the data and expect the IT team to provide clean data and ensure security. [3]

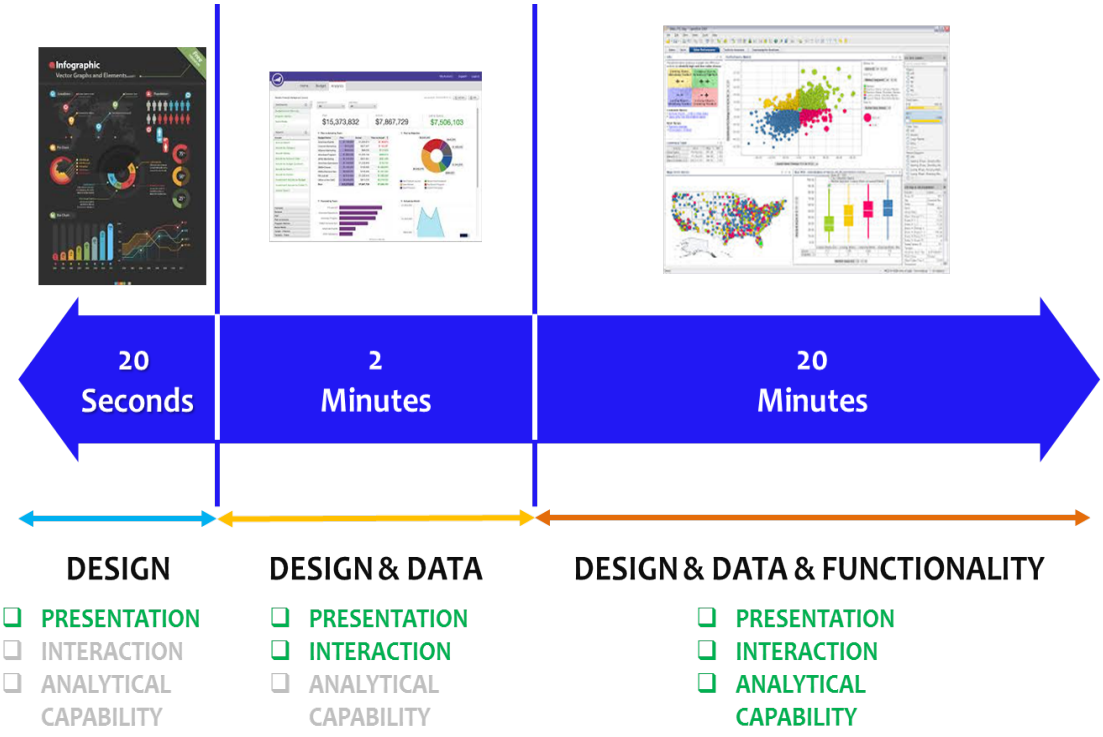

Figure 1. Data visualization tools and their user's exploitation time. [3]

Price

Another factor that plays major role in choosing the right visualization tool with the needed features is how much it costs. Almost all tool providers offer a free and full version of their product for a noncommercial use but some of the known ones come with pricing when it comes to a commercial use.

Architecture and Scalability

All visualizations have to be simple and understandable just by viewing and every seconds spent on the visualization must offer the viewer a much more detailed information about the data. [4]

Due to the rapidly growing mobile devices industry, testing data visualizations for mobile devices while developing is becoming a priority. The design needs to stay the same on all the devices with the use of responsive design.

The crucial point of any architecture should be choosing the right method with the message as a focal point, not how attractive or colorful the charts are going to be. That might end up leading to information overload or miscommunication between the data and its audience.

Customization

Customization refers to any tweaks made to an existing features of the library based on user preferences with several options of modifying it. Most libraries default settings have a good starting point for presenting a simple application, but that will not be enough for bigger projects. Customizability can be implemented by either using an API from another source or by letting others support the system by making the sources open for a public use; if not, the system or the application has to be used as it is. [4]

Functionality

All the different shapes of data visualization have their own qualities. It is important to know the priorities of the project; should the functionalities of the visualization lean more toward making it look prettier with animations and all sorts of decorations which might slow the page, or make it a little less shiny and present the data on its ugliest form; experts on the field suggest that the main emphasis should be on the end-user experience. The quality of a well prepared data visualization can be how well organized it is in one particular place, how eye-catching it is at the first sight and how much user friendly it is even for users with no experience with the technologies it is presented with.

The functionality is mainly the options the project takes when deciding which approaches must be taken while deciding what tools to use and how to use them; and it answers the core questions such as; [4]

- whether the chart is dynamic with interactions or remains unchanged
- Is it going to be a single chart or dashboard with multiple charts?

- easy to embed all the features
- Will the features be collaborative or shared?
- What is the the main purpose of the visualization? Is it observational, story-telling or demonstration?
- device support and browser compatibility

How fast is the platform or the environment of the system?

A performance based architecture is a crucial foundation for a better user experience. One of the primary criteria of search engines to respond for a search request is; the website's load time, which makes websites with slower load time to be less favourable to be on the top of the search results. The bigger the data, the less efficient the system and the more complicated the visualization.

What type of chart to use?

As data visualization is all about telling a story or a series of stories, the writer has to ponder upon how he or she wants to tell the story and which way of storytelling is best suited for the audience by making the data the main focus point of the presentation. The transition from the raw data into powerful visualization relies heavily on the choice of the right charting type. Though the decision is not going to be a hard one to make, but one has to figure out which charting type is a perfect fit for the data visualization because all can be used to present the data but might not tell the right story directly or drive the presentation into unintended narration. It is recommended to try out different types of chart or graphs before settling on one of them blindly. Here are the some of the mostly used types of charts and their purpose of usage.

Chart Type	Definition	When to Use
Bar and Column Chart	Displays data of different items horizontally and vertically	good for comparisons
Line Chart	Links points of data on a time sequence	To shows changes over time
Pie Chart	A circular chart with partitions between the amount of different data presented	Good to display simple datasets with proportions but us-

		age of pie charts for non static big projects is advised to be avoided
Map	Used to show geolocation based items with an area map as a background	Made for location based data
Sparkline	A single line chart without a grid and a y-axis value	Used for showing time series of certain changes. For example, can show the daily rank of a product
Bubble Chart	Leans towards putting emphasis on vaguely presented datasets on a map or a scatter plot charts	Used to focus on specific data based on axes
Histogram Chart	A vertically places rectangular shapes presenting data, not to be confused with bar chart where the bars have gaps between each other.	Used to show continuous data sets.
Waterfall Chart	Also known as the Mario chart due to the the visualizations of some data resembles to the block of bricks floating in the air from Super Mario the video game.	To show the positive or the increasing and the negative or the decreasing changes over time.
Radar or star charts	Presents values of different groups which begins from the center and goes till the outer part depends on the values	Not suitable for user with no data visualization experiences
Heat Map	Used to compare data of different groups with colors	To show differences and similarities of different aspects of points

Table 1 Types of charts

Other most used chart types which are worth mentioning are Candlestick chart, Bullet chart, Gantt chart, Highlight table and Box-and-whisker plot.

If using a chart is not a requirement to be implemented, using scorecards and gauges can be a good option depending on the need of the project. The disadvantage of this approach is that only the current status of the data is available with its minimum and maximum points.

2.3 Principles of Designing a visualization

It is self evident that differences in data sets and the project end goals shapes the look of a visualization, but the same core principles of design apply for almost all types of architectures of visualization. [5]

1. **Choose Informative over decorative** - improve the chart by removing grid lines, background colors, unnecessary animations
2. **Color matching** - the contrast between the colors used in the chart affects the readability of the chart in general
3. **Labels** - a good definition for the positioning of a label and a reasonably readable font size are crucial elements of readability of a chart
4. **Sorting** - filter to visualize only the important information so the audience would not be misled by the scrambled presentation
5. **Color alteration over different colors** - using different hue and grey, a chart can tell a perfectly crafted and painted story. Usage of different color for different data sets is more appealing than the one with alteration, but it is also hard to follow with the all the colors creating a beautiful mess right in front of the beholder.
6. **Smooth line must be avoided** - smooth lines that barely shows the data points and effects which have no direct relation to the data needs to be removed
7. **Avoid Repetition** - study the data well enough to avoid repetition or lean toward a method that minimizes as much repeated information as possible.

2.4 Performance

Through the few decades of internet's existence, so much has changed and the changes are happening on a pace that is highly demanding to catch up. Technology will keep on evolving by the seconds by giving birth to new trends which comes and goes by the seconds likewise. Nowadays speed of a website is one of the prior factors to consider utterly for a better user experience. How much of a time that a web page will take to load will determine the continuity of usage by the visitors, recommending it to the other and revisiting the site. Researches shows that, if a website is as slow as taking two seconds and more of loading time, more than half of its visitors tends to abandon it. [6] When excessive HTTP requests such as loading CSS style sheets, scripts, html, images and etc., are made to the server and the longer those elements take to render, the slower a website is going to be.

Data visualization must give a direct, accurate and quick information to its audience and performance should not be a drawback for the user to get any pieces of knowledge from the presented chart. The data needs to be a click away. The bigger the data points are, the longer it takes the page to load and render. The bottleneck of dealing with large web data is the rendering of DOM, specially with usage of Vanilla JavaScript will keep the DOM running slowly.

There are solutions that can be taken to resolve a performance problem depending on the type of visualization used; a few of them are listed below,

- By looking for a possible way to reduce data; for example, by using a search method to a specific area of an indexed data rather than looping through the entire data every time a particular data is needed.
- Tree Data Structure [7]
    - Quadtree for points
    - Rtree for rectangular search index
- Cross filtering the data sets with a library called crossfilter.js
- Polyline simplification is a method of decreasing the resolution of the polylines on the chart. [7]
    - simplify.js is a good library by Vladimir Agafonkin
- Applying hierarchical clustering, which is a way of categorizing or slicing up a large data set into smaller ones by the similarities they share.

- Whenever the data is load, the UI is going to freeze, because the everything happens in one thread. It is wise to load data with Web worker, because it is going to run the data on the background without distracting the UI.
    - Use the transferable objects method to make a freeze free pages

After the the page load is known, there are a few essential steps that need to be taken to speed up the web page's load time as stated below [8]

- compressing files by using GZIP compression
- scaling images,
- caching on the browser,
- presenting images in CSS sprites
    - CSS sprites is a way of combining many images together in a single file and access them by using their position
- using the "defer" attribute on the head of the HTML page to let the parsing of JavaScript happen after the page load
- minify all of the files used to create the web page.
- Use the local storage to store some assets for an easy access
- Prefetching
    - For caching
- make a use of Content Delivery Network or CDN
- Load only crucial assets for the load time then load the rest of the assets after the viewer has the visuals in front of himself; for example, tools such as Google analytics can load after the page is fully ready.
- Render blocking
- Parse blocking
- Using a pure function that can render fast

The first factor that determines the loading time of multiple charts is the speed of the internet of the user, but the development process must fix most of the performance issues. User connectivity speed is irrelevant to bringing the best user experience. There are multiple factors that directly affects the loading time of charts on the website which are the reasons for the slowness of the page and need to be eradicated before launching as listed below [9]

- **The size of the library**
  - the bigger the files are the longest it will take for the browser to load it.
  - Using the Unix C program JSMin, all JavaScript files can be condensed into a minified version of itself
  - avoid making multiple calls to load the library

- **Memory Issues** -
  - Not so many developers assess the way the are using memory, because of the informal way JavaScript has been built
  - Avoid redundant declaration of variables
  - Optimize the loop, for example by declaring variables outside of the loop
  - Decrease the size of the DOM and accessing it multiple times through JavaScript. [9]

- **Rendering and re rendering time**
- **Garbage collection**
  - Beware of how to use references, to not stack up the garbage collector with extra unneeded information.
- **Number of browsers repaint cycles**
- **Excessive amount of data points**
  - It is best to rethink and assess the methods used to create the chart in order to decrease the number of the points
  - Choose another mathematical or design solution to lessen the bloated presentation; which will make the chart difficult to read as well
  - Less is more! The smaller the points, the quicker that the audience can be allured by the looks of the chart

2.5 Performance testing

Performance testing refers to techniques that will be used to examine how consistent the system is before it is passed for use. The main purpose of performing tests repeatedly is to make the application go through all the possible defects in order to make the lifecycles of the web application flawless. The front end and the backend need to be tested thoroughly for the web application to perform efficiently. [11]

Ever since the booming era of the social media applications such as Facebook, Instagram, twitter, etc., contents has increased resulting in an obvious slowness of viewing assets on a web page. Earlier, web pages were simple and interactions were not as important as they are today. Human-Computer Interaction or HCI, solicits website owners to keep their sites below the page loading of time less than 2 seconds; if the web page takes a load time longer than that it starts to affect the conversion marketing. The leading technology oriented companies such as Google, Mozilla, Shopzilla and Netflix has discovered that performance is a vital feature of the web which determines the decline and growth of revenues.

An overall optimization of JavaScript and the data is necessary to bring the website up to speed, some methods are shown below:

- Adding ‘async or defer’ methods on the script tag will prevent JavaScript from blocking the Dom and CSSOM.
- Using an inline JavaScript will improve performance by reducing the fetching time of external JavaScript files.
- Divide a big data set into multiple pieces and display them one after another; this liberates the bloated visualization and boost the performance by letting the DOM take care of smaller ranges than one big chunk of data sets. A good example of this method is Cubism JS by Square Open Source. [7]
- diminish the dimensions of the data by altering a long ranges of presentation of a decade coverage of data into a year and real time visualizations into hourly or daily reports.

Crucial matters to remember are that it is not a good method to start stress testing before the main functionalities of the website are built. Stress testing a simple webpage with few operations in it is totally unnecessary and futile. A good start to stress testing can be using the already built-in JavaScript functions and the the browser’s developer tool as shown below with some sample codes:

- new Date().getTime() or new Data() function
    - this built in function helps a developer know how long that it takes to parse and execute a function by subtracting the end time from the starting time.

- It is an obsolete method to find the rendering time.

```
72
73  var startTime, endTime;
74
75  startTime = new Date().getTime();
76  function() {
77      //code to be executed goes here
78  }
79  endTime = new Date().getTime();
80
81  console.debug('Elapsed time:', (endTime - startTime));
82
83  //Output  Elapsed time: 568
84
85
```

- performance.now ()
  - it is more or less similar to the Date function in JavaScript but more accurate with microseconds included

```
85  var startTime, endTime;
86  startTime = performance.now();
87  function() {
88      //code to be measured goes here
89  }
90  endTime = performance.now();
91  console.debug('Elapsed time:', (endTime - startTime));
92
93  //Output Elapsed time : 467.46000000000004
94
95
```

- console.time()
  - by placing this function before the code is executed and by calling its operation successive function console.timeEnd() after the code is the executed; one can know how long that it takes for that function in particular to run.
  - It a timer and it is not a web standard yet and should not be used for a production level application
  - Works in all the major league browsers but it is only compatible with Firefox Mobile when it comes to working with mobile application development

```
93
94      console.time("timer number 1");
95
96      function () {
97          //code to be measured goes here
98      }
99      console.timeEnd("timer number 1");
100
101     //output  timer number 1: 467.460ms
102
103
```

- Using the navigation timing API
    - A more accurate function provided by the W3 group with plenty of options to assess the page's initial activities

```
71  })
72
73  var pTime = window.performance.timing;
74  var pageLoadTime = pTime.loadEventEnd - pTime.navigationStart
75  var requestTime = pTime.loadEventEnd - pTime.responseStart;
76  var responseTime = pTime.loadEventEnd - pTime.responseEnd;
77  var domLoadingTime = pTime.loadEventEnd - pTime.domLoading;
78
79  console.log("loading time of the page: " + pageLoadTime);
80  console.log("request time: " + requestTime);
81  console.log("Response time start: " + responseTime);
82  console.log("Dom loading time: " + domLoadingTime);
83
84  //output
85  // load time of the page: 272(ms)
86  //request time: 254(ms)
87  // Response time start: 252(ms)
88  //Dom loading time: 245(ms)
89
90
91
92  class ArticleComponent {
```

2.6 Built in developer tools from the browser

A web browser is responsible for taking the the program files sent to it and output the expected visuals. The browser which the web application runs on affects the performance and the functionalities of the web page. The different JavaScript Engine that

each of the browsers are running the website on, being the foremost reason how fast the page loads and runs. Another essential instrument that all browsers provide to a web developer is a built in developer tool which allows the developer to dig deeper into the core building blocks of how the code is coming together. This tool enriches the developer with vast options from inspecting a simple element in the page up to debugging the code. There are countless browsers, however the most favourable ones are Chrome, Firefox, Safari and Internet explorer. The presence and excessive use of data visualization is noticeable in every developer tools. [15]

Chrome

The Web browser Chrome has one of the best tools for developers to view how their code is transformed into visuals, how it is affecting the DOM tree and get deeper insights from it. The network and the timeline panels have a good presentation of all the necessary details about performance.

The network panel can record the network activities and display them on pictures with all the changes made through the time of its run; it also states how many http requests has been sent and how long it took to give the responses. The other two important events displayed as features on the network panel are DOM Content Loaded, which is; the initial time spent to load and parse the HTML without its assets such as images and style sheets and the other one is Load; which displays the time it has taken to load the page fully.

The timeline panel has graphical presentation of time spent for the loading, scripting, rendering and painting of the page. Chrome also has the most efficient checker for the memory usage to get the idle elements from the DOM tree and how it is passed around to the DOM nodes and the JavaScript objects through Heap snapshots. [16]

The figure below shows how google chrome displays its timeline with a data visualization.

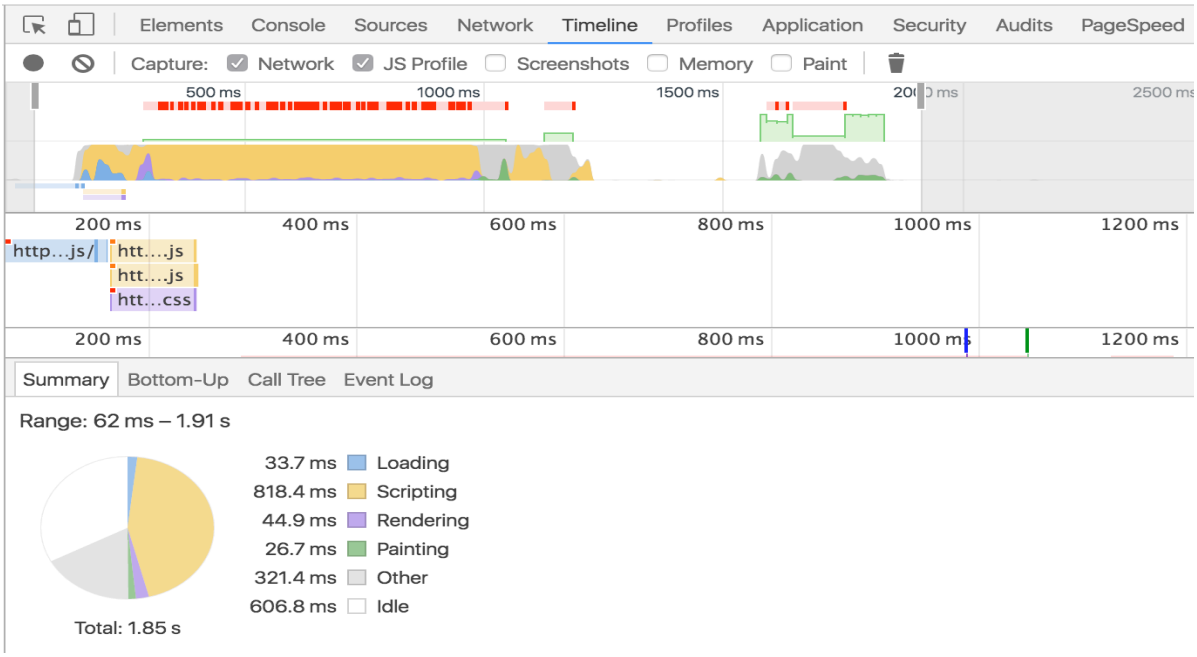

Figure 2, Google Chrome's Developer Tool view of the Timeline tab

Firefox

Firefox's developer tool has a network and a performance panel which are both used to assess the how the code is affecting the performance of the browser. In the performance panel presentations are given through the graphically painted timeline which has three different view types; Waterfall, Call Tree and Flame Chart. [10]

Safari

As all the other major browsers Safari also has a well built developer tool which has an efficient timelines and network tabs to test the performance and rendering settings of the page. The timeline tab graphically shows the activities happening during the page's existence. It records and colourfully presents the network requests, the events and rendering, with an option to record the entire activities after the page is fully loaded.

Microsoft Edge

The most outstanding new comer for the browsers war is the latest version of Windows 10's browser which outshined each and everyone of its contenders on a performance test done by the experts in the field. The lack of popularity or losing to grab the interest of a wider community support.

2.7 JavaScript Engines

The speed of JavaScript on the browser is the major factor that makes a massive difference on the performance of the browser's JavaScript engine. A test done by a web developer named Jacob Gube compares the performance of web browsers based on six core points, which are JavaScript speed, DOM selection speed, CSS rendering speed, CPU usage, Page load time and the browsers cache performance. The overall result of his test shows that Google Chrome is the leading browser. [10]

- Nitro Engine of Safari
- V8 of Google Chrome
- Trace Monkey of Mozilla Firefox
- Chakra or JScript engine of Internet Explorer 9

3 Method Section

3.1 Introduction to charting libraries

As Steve Souders stated in his book, High Performance web sites, most of the response time are spent on the front end part of the web site while only less than 20% of it is used to get the document from the server, so, a good tackling of a performance problem or improvement must start from the frontend. [11]

The initial starting point of this thesis was to perform a stress testing of different charting tools for web development and use the applicable and most powerful one for the company project. The first test was made to test the current popular libraries by using a line chart with three similar data sets, as shown in Figure 3 ; after the first test has been

done, the second phase continued with increasing the number of charts in a page to fifty charts.

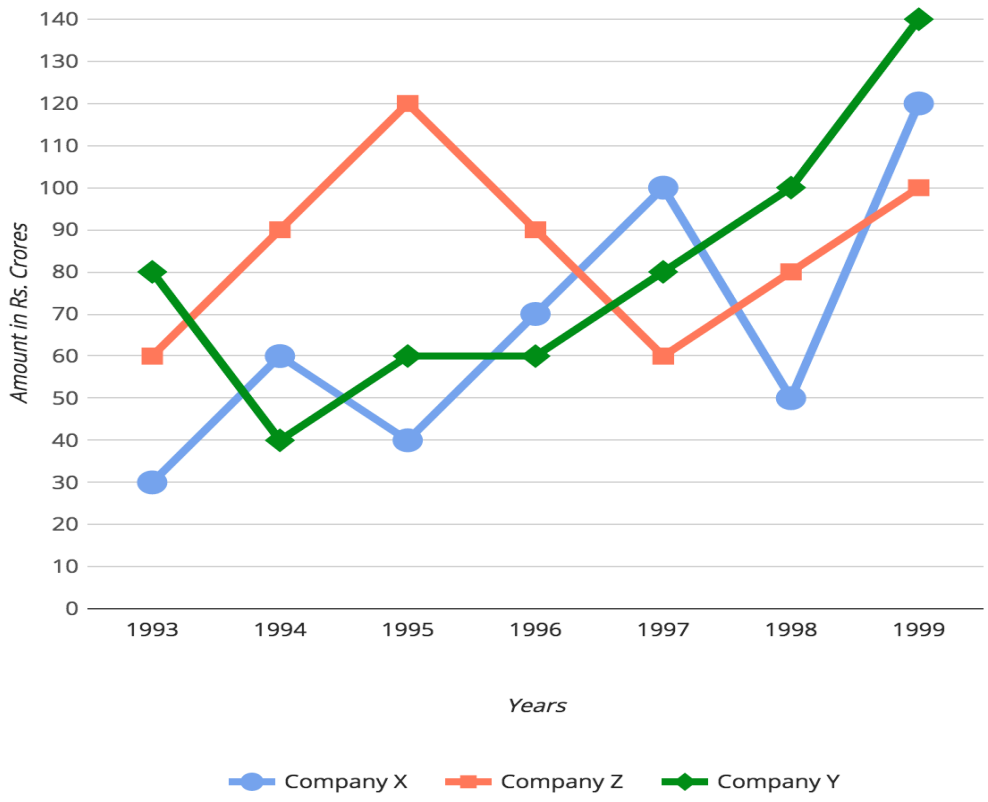

Figure 3 a Line chart made with google chart for the first round test

The stress testing section of each libraries assess the libraries load time and rendering time. The load time is the amount of time spent by the browser to download all the JavaScript and CSS files, images and any used objects; and the rendering time refers to the processing time. The *DOM content loaded* event launches the main html page without waiting for extra files such as images and style sheets.

3.2 Canvas vs SVG

The decision of selecting between these two HTML technology standards must come first and defines what the project tends to value. If the project is going to involve a large dataset, the interactivity, the built in animation support, best support of event handlers and the DOM API control, the resolution independence in different devices and the accessibility advantages from being an XML based graphics of SVG might not come in handy, because of how the page starts to crumble from trying to handle all the node points created for each and every element. Canvas is pixel based and performance

won't be affected after rendering unless the resolution of the graphic is increased. [12] Other good advantages are that if the canvas has repeated elements, it is possible to copy and use it after it is drawn once, it is also good to boost the performance; and its performance does not show an enormous variation when displayed in different browsers as much as SVG. [13] The downside of using Canvas is its non modularity when it comes to interactivity. Few points to take on to optimize the performance of a canvas are

- Draw and redraw partially
- Draw once and copy
- Use a minimalistic approach to the styling of the canvas, such as the coloring fill and the stroke
- Using typed arrays for pixel manipulation in Web Worker

Depending on the project, using both SVG and canvas together is another was to get the best qualities of both technologies. Another standard for making graphics is WebGL, although mainly used for 3D effects purpose, but its 2D version is much faster.

DOM (Document Object Model) manipulation is what charting is all about, if a JavaScript framework such as AngularJS is in use, directives are a good approach to it.

3.3 Tests

The tests done with the different charting libraries had three different stages. The first stage involved searching for the mostly used libraries and filtering out the efficient ones related to the project goal; this was done by choosing twelve different JavaScript visualization libraries and write a small application by using same datasets for all of them. This stage led to finding out which ones were more user friendly, easily customizable, fast to load and render the charts drawn by using them. The first stage's aim was to make a line chart of three lines with only 7 data points shown on it, to assess how long does it take for a library to load and render a single chart. The second stage was used to test how powerful the libraries are to draw 50 charts and increasing the size of the data to assess how the library will handle more data points. This method helped to sort

out which libraries are more capable of handling on the capacity run them without performance drawbacks.

The DOM Content loaded function refers to the time it took for the DOM and CSSOM to load and become fully ready.

3.4 Charting Libraries

Chart.js

Chart JS is one of the leading JavaScript libraries for data visualization which uses the HTML5 canvas element to draw and render charts. The library is a perfect fit for small projects or projects with a limited usage of charts that has a bloated dataset. Chart JS is a new and upcoming library, so it only uses six of the mainly uses chart types which are line, bar, radar, pie and donut, polar area and bubble charts. It is free and open source. It has a fairly dedicated community which had the second version of it up and running within a few months.

Technical Details

- It is easy to get started with a project using Chart JS, for the reasons of its flexibility to work on options such as scaling the axes, tooltips, labels, colors, custom actions, and much more.
- It has faster interactivity response.
- The legend functionality comes with the default options but the library only generates the HTML so one has to add it to the innerHTML property in order to set it up
- Configuration of a functionality can be given for all the charts all together or the specifically for a chart existing on the web page
- The canvas element needs a bit of workaround in order for it to work well when the window is resized
- No stable new versions and there is a big difference and changes with the previous version and the newly added which makes almost the older version charts unusable with the update
- Major issues with mostly used components such as legends, title, axis controls, etc.
- It does not have support for JSON file format by default, which makes it a bit time consuming to work around it in order to get a dynamic data being displayed

```
'use strict';

$(document).ready(function () {
    $.getJSON("data.json", function (result) {

        var labels = [], data = [];
        for (var index in result) {
            var res = result[index];
            for (var r in res) {
                labels.push(new Date(res[r].x * 1000));
                data.push(res[r].y)
            }
        }

        var tempData = {
            labels: labels,
            datasets: [{
                fillColor: "rgba(172,194,132,0.4)",
                strokeColor: "#ACC26D",
                pointColor: "#fff",
                pointStrokeColor: "#9DB86D",
                data: data
            }]
        };
        var options = {
            animation: false,
            scaleLineColor: "rgba(0,0,0,0)",
            scaleShowLabels: false,
            scaleShowGridLines: false,
            pointDot: true,
            datasetFill: false,
            showTooltips: true,
            showLines: false,
            scaleFontSize: 1,
            skipLabels: 13,
            responsive: true,
            scaleFontColor: "rgba(0,0,0,0)"
        };
        var i = 1;
        for (i = 1; i < 50; i++) {
            var canvases = document.createElement("canvas");
            document.body.appendChild(canvases);
            var div = document.getElementById('gallery');
            div.appendChild(canvases);
            canvases.width = 600;
            canvases.height = 150;
            new Chart(canvases.getContext("2d")).Line(tempData, options);
        }
    });

```

Stress Testing Results

Browser	Time taken for the server response	Time taken till inter-activity	Time for Assets to be downloaded	Total load time
Chrome	4	157	46	236
Safari	5	110	24	147

Firefox	0(?)	107	30	176

Table 2. Chart JS test results

NVD3.js

NVD3 is a front runner of JavaScript libraries based on D3 or Data-Driven Documents, which is the most widely used data visualization tool. D3, built on top of the the common web components, HTML, CSS, the DOM (Document Object Model) and SVG or Scalable Vector Graphics, is a highly powerful data visualization library. D3's lack of user friendliness makes it not feasible to developers with little experience to start with and it also does not come with any charts by default which leads to hard coding the visuals. Here is where NVD3 comes to save the renowned status of D3. D3 has been a foundation for NVD3 and so many other plugins and libraries such as Graffeine, Mermaid, Epoch, Parallel coordinates, D3GH, insights.js, Raw, Tributary, JSNetworkX, Dimple, DC.js, DVL, Graphene and many more. [14]

As the official website for the library states, NVD3 is built on top of D3 to make the usage of re-usable charts and elements of it to empower its parent library.

Technical Details

- The library is inspired by the groundbreaking reimplementation of its parent library, D3, by Mike Bostock with a support of its committed community and the platform Novus.
- User friendly and easy to create a simple and good looking charts to begin a project with
- Open Source and free
- Responsive and dynamic with the the CSS file connecting the dots of styling the charts
- The D3 JavaScript file needs to be included before all the other JavaScript libraries, so that all components run well.
- Works well with D3 version 3.5.3 and below but not yet compatible with D3 version 4.0 and above.
- Compatible with web kit based browser such as the latest versions of Safari, Chrome and Firefox, Internet Explorer 10 and above and Opera Mini.

Stress Testing

Browser	Time taken for the server response	Time taken till inter-activity	Time for Assets to be downloaded	Total load time
Chrome	19	28	112	189
Safari	21	17	134	228
Firefox	26	32	125	295

Table 3. NVD3 test results

Flot

- Flot is one of the oldest plotting JavaScript libraries
- It is a jQuery plugin
- It works with all browsers including IE6
- Full in hand use of animation, presentation and interaction
- the library doesn't get the zero point so the zero point of the Vertical or the y-axis and Horizontal the x-axis must be specified, by default it will take the minimum value as a zero point
- Need to include explore canvas for the support of older browsers
- Setting the simplest part of a chart is easy, but one must work around to make a chart interactive
- User interactions like panning, zooming, resizing, switching a data series and more…
- Had a wide variety of other user-created plugins
- Open Source and free

Technical Details

- After creating the div element, the width and the height of the graph has to be set, so that the library knows how to scale it.
- In order to hide the additional invisible element, it forces the developer to use a tweak to get the element and delete it.

Metrics Graphics

- It is a library optimized for visualizing and laying out time-series data.
- The library is based on D3
- Compared to other libraries, it is light weighted with the minified file shipping only with 80kbs and the library is easy to get started with.
- It supports line charts, scatter plots, histograms, bar charts and data tables
- It got a simple to use list of options for axes, data, layout, graphics and some more options based on the chart type

Chartist.js

- It is the lightest JavaScript library. The documentation states that the library weighs only 10kb but the minified file of it is 38kb when downloaded.
- It does not have dependencies.
- It includes few of the most used charts which are responsive, animated and rendered beautifully.
- The setting is built with a simple handling while using convention over configuration.
- It comes with a great flexibility while using clear separation of concerns
- It is an SVG based library which is fully built and customized with SASS
- Before it has been terminated SMIL (Synchronized Multimedia Integration Language) was a good way to do animations as well
    - with chartist SVG animation API
- It comes with a responsive configuration with media queries
- As it is stated on Table 3, the library works with all browsers (IE9 and and above)
- It has wrapper libraries with the well known and commonly used frameworks such as Node JS, Angular JS, React JS and more
- animation with CSS

Technical Details

- It does not have interactivity by default
- Although the documentation provides a workaround to get a tooltip, but for projects with little time the customizing to the projects need might be challenging

- there aren’t a lot of plugins to choose from to use effects such as tooltips which are easy to use in other visualization tools
- easy to use functionalities such as spacing for Y axis and grid control are still existing issues in Chartist.

Stress Testing

Browser	Time taken for the server response	Time taken till interactivity	Time for Assets to be downloaded	Total load time
Chrome	3	104	159	294
Safari	13	79	223	329
Firefox	0 (?)	31	113	187

Table 3. Chartist JS test results

Code Sample

```
/**
 * Created by boochoo on 15/08/16.
 */

var demo = angular.module('demo', []);

demo.controller('DemoCtrl', function ($scope, $http) {

    var startTime = new Date().getTime();

    $http.get('data.json').then(function (data) {


        $scope.data = data.data[0].values;

        $scope.xAxis = [];
        $scope.yAxis = [];


        angular.forEach($scope.data, function (v, k) {

            $scope.xAxis.push(new Date($scope.data[k].x * 1000));
            $scope.yAxis.push($scope.data[k].y);

        });

        $scope.lineData = {
            labels: $scope.xAxis,
            series: [
                $scope.yAxis
            ]
        };

        $scope.lineOpts = {
            showPoint: false,
            fullWidth: false,
            chartPadding: {top: 5, right: 5, bottom: 5, left: 5},
            axisX: {showGrid: false, showLabel: false, offset: 0},
            axisY: {showGrid: false, showLabel: false, offset: 0}
        };
        var j = 1;
        for (i = 1; i < 100; i++) {
            var chart = document.createElement('div');
            document.body.appendChild(chart);
            var div = document.getElementById('gallery');
            div.appendChild(chart);

            new Chartist.Line(chart, $scope.lineData, $scope.lineOpts);

        }
```

The above code shows how the first test was done with the library chartist.

Dygraph JS

- The library is easier to work with CSV data sets, although a work around the provided file format is still possible for the user application

- It was build to maneuver large data sets without a breaking
- Has a minimalistic design
- Developed by Google
- Built in range selector, zooming functionalities
- Customize point shapes with a library named shapes.js
- Interactive and Responsive out of box
- comes with powerful handy functionalities such as pinch, pan, mouse over and zoom which makes it favorable to fit developing for mobile devices as well
- Open Source and free
- Works with almost all browsers (backward compatibility with IE8)
- Options and custom callbacks make the library highly configurable

Stress Testing (all the test results are in milliseconds)

Browser	Time taken for the server response	Time taken till interactivity	Time for Assets to be downloaded	Total load time
Chrome	3	104	159	294
Safari	13	79	223	329
Firefox	0 (?)	31	113	187

Table 4 Dygraph test results

Highcharts

- A free version of the library is available for development but payments with different ownership and access are provided for commercial apps
- Easy to set up same as google chart
- Well built with several pre designed
- The well designed charts make customizing much more easier
- A bit exceeded loading time
- Responsive
- Uses SVG for modern browsers and Vector Markup Language (VML) for older ones
- Even though, it is still in progress; with the addition of the module boost.js an application can run up to a million data points without a problem.
- It has a feature for downloading an image format of the current visuals

Google Charts

- Well documented with a commented code
- Easy to set up and have a go with it
- Good support from its community
- Easy to customize components such as point shapes for multiple line charts
- Free but not open source
    - Can’t be good for sensitive data

Stress Testing

- Load time = 260 ms
- Rendering time =
- DOM content loaded = 256 ms

N3-charts

- A simple chart library for Angular JS applications
- Built on top D3.js
- Simple and interactive
- Doesn’t have a well defined documentation
- Visibility of a single line can be achieved by setting up unique ids for the series
- No premade title for X and Y

Canvas JS

- The people behind it claims it to be the fastest charting library
- It can handle large data within a short amount of milliseconds
- It is allowed for a free development use but has a paid version for a commercial depletion
- The library is responsive out of box

Stress Testing (all times are in milliseconds)

Browser	Time taken for the server response	Time taken till inter-activity	Time for Assets to be downloaded	Total load time
Chrome	94	2	191	880

Safari	4	5	58	612
Firefox	12	74	199	516

Table 5 Canvas JS test results

Other important charting libraries which deserve to be mentioned are PlotlyJS, am-Charts, zingCharts, eCharts, C3JS,uvCharts,SciChart and FusionChart and Datawrapper.

3.5 Stress testing on mobile devices

The testing for all the libraries was done both on a personal computer and a mobile device. All the presented testing results are from the testing done on a personal computer and the performance speed of handling charts of most libraries tested differs from the test results taken by the mobile devices; so does the output or the visuals. Another difference is the interactivity in some libraries does not perform well or not at all in mobile devices. Some of the browser characters from the tests done on a mobile device are stated below;

- On safari browser
  - While testing ChartJS, when 50 charts are drawn with the library, the page starts to break and struggles to start before a repetitive reloads.
  - Tests done with Dygraph JS showed performance breaks between the charts drawn
  - Google charts work well.
  -
- On chrome mobile browser
  - Google chart works well.
  - Chartist, it starts working properly after a couple of reloads.
  - ChartJS outputs similar and runs well

4 Discussion

On the search for the lightest, the fastest to load, easy to control and strong sustainable library has been done by testing more than 12 current and popular libraries. Even though, the project goal and the data provided defines which library is the right library to use, one can save a project budget and time by using a charting library.

Due to the lack of a proper testing environment for performance testing, the stress testing done on the project shows inconsistency when the page is reloaded. The developer tool methods used to get the execution time differs from the previous test results. Browser based tools such as Yahoo's YSlow and Google's Page Speed show methods to keep the code clean by pointing out what is making the website to run slow and general guidelines of what needs to done. Although, applying those methods can make the website run faster, but it does not support the developer to get the precise time of how long it takes for the functions used to be executed.

The charts performance on a mobile device is another problem which invites additional difficulties for the decision making process of choosing the right library. Even charts such as Dygraph; which is created for handling larger datasets and data points performs weak when used on a mobile phone. With how much of information customers get from their mobile phones, it will be a big loss if the library can not deliver the data visualization with a performance fluidity.

5 Conclusion

The main purpose of this project was to test libraries used for data visualizations for a web based application then optimize the performance of the platform based on the outcome of the tests. As JavaScript is the language of the web and used by the libraries tested, a deeper understanding of how it operates and how the JIT interpreters compile the DOM which converts the computer readable code into the human readable HTML file, CSS interpreter which is responsible for stylizes the HTML page to make it more

appealing and the rest of the compiling done depending on the browser's JavaScript Engine.

The evolvement of data demands urgent methods of control and preservation of it. Data visualization is one of the methods and presenting a complex analytics is not as difficult as it used to be with all the right tools being around us. Although, it is essential to test the performance in every steps of the development process, the significance of stress testing excels in bigger projects. Further more, the necessity for a tool while developing an application locally is compulsory while there are many available for production level applications. Most of the libraries used for the stress testing were user friendly to start a project with and become challenging to control when the project gets bigger. Most promising open sourced libraries are neither well documented nor have a community that support and update the bug fixes requested by the chart's users.

References

1 Michael Palmer (Nov 3 2006). Data is the new oil[online]

http://ana.blogs.com/maestros/2006/11/data_is_the_new.html

Accessed August 25, 2016

2 Ben Fry (2008). Visualizing Data. O'Reilly Media, Inc., 1005 Gravenstein High-way North, Sebastopol, CA 95472. Page 5

3 Kaptain(2015). Infographics vs dashboard vs application[online]

http://wisdomschema.com/2015/03/infographic-vs-dashboard-vs-application/

Accessed Aug 25, 2016

4 John Griffin (29 May 2016). The best web-based data visualization tools[online] http://atchai.com/blog/2016-03-29-the-best-web-based-data-visualisation-tools/ accessed in August 2016.

5 Mico Yuk and Stephanie Diamond (2014). Data Visualization for dummies. John Wiley & Sons, Inc., Hoboken, New Jersey. Page 175

6 Shaun Anderson (15 April 2016). How fast should a website load? [Online].

http://www.hobo-web.co.uk/your-website-design-should-load-in-4-seconds/

Accessed in August 25, 2016

7 Vladimir Agafonkin (2013). High Performance Data Visualizations[online]

https://vimeo.com/68252990

Accessed September 29, 2016

8 Steve Halverson (Feb 21 2013). How to speed up your website load times[online]

http://www.webdesignerdepot.com/2013/02/how-to-speed-up-your-website-load-times/

Accessed in September 4, 2016

9 Coach Wei (November 21 2011). Seven mistakes that make websites slow[Online]

https://www.sitepoint.com/seven-mistakes-that-make-websites-slow/

Accessed September 2, 2016

10 Jacob Gube (October 9, 2009). Performance of Web Browsers [Online]

http://sixrevisions.com/infographs/browser-performance/

Accessed in November 6, 2016

11 Steve Souders(2007). High Performance web sites. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Page 1

12 Paul Rouget (2011). Faster canvas pixel manipulation with typed arrays[online]

https://hacks.mozilla.org/2011/12/faster-canvas-pixel-manipulation-with-typed-arrays/

Accessed September 29, 2016

13 Maria Antoinetta Perna (March 23, 2016). Canvas Vs. SVG: Choosing the right tool for the job[Online]

https://www.sitepoint.com/canvas-vs-svg-choosing-the-right-tool-for-the-job/

Accessed in September 1, 2016

14 Mike McDearmon (2016). Charting libraries using D3[online]

http://mikemcdearmon.com/portfolio/techposts/charting-libraries-using-d3

Accessed Aug 30, 2016

15 Craig Buckler (January 12, 2016). Browser trends January 2016: 12 Month Review. [Online]

https://www.sitepoint.com/browser-trends-january-2016-12-month-review/

Accessed November 7, 2016

16 Kayce Basques (September 21, 2016). How to use the timeline tool? [Online]

https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/timeline-tool

Accessed September 21, 2016