Examensarbete

HÖGSKOLAN I SKÖVDE

SCALABILITY OF JAVASCRIPT LIBRARIES FOR DATA VISUALIZATION

Bachelor Degree Project in Information Technology Basic level 30 ECTS
Spring term 2021

Jonna Persson

Supervisor: András Márki
Examiner: Yacine Atif

Abstract

Visualization is an important tool for making data understandable. Visualization can be used for many different purposes, such as charts on the web used to visualize a dataset. Render time is important for websites since slow response times can cause users to leave the site. When creating a website with fast render times in mind, the selection of JavaScript library may be crucial. This work aims to determine if dataset size and/or chart type affects the render time of charts created by different JavaScript libraries. The comparison is done by a literature search to identify suitable libraries, followed by an experiment in which 40 websites are created to compare the performance of selected JavaScript libraries for rendering selected chart types. The results show that while both dataset size and chart type affect the render time in most cases, the libraries scale differently depending on the dataset size.

Keywords: Visualization, JavaScript, Charts, Scalability, Render time

Contents

1 Introduction

Render time refers to the time it takes for a website to load visually and become interactive. Render time is something crucial for web developers, since a slow website will cause users to leave more rapidly (Solarwinds Pingdom, 2018). Websites with a long render time could also have fewer visitors due to Google Search ranking web pages based on their load time (Google Search, n.d.).

JavaScript is used on over 97% of websites today (W3Techs, 2021). A JavaScript library is a collection of code that can be used to simplify the creation of websites and is a way to re-use code. Multiple libraries exist with several purposes, including creating charts to visualize data. Four JavaScript libraries are used in this work, namely ApexCharts, Frappe Charts, Google Charts, and TeeChartJS.

Visualization of data has gotten more and more popular during recent years (Rau et al., 2018). It can be used in many ways, as demonstrated by Huang et al. (2019) who used visualization to convey a user's emotion back to them and Li et al. (2014) who used visualization to present information collected from tweets in different forms. Visualization is important to explore, confirm and present data (Rau et al., 2018), as well as to identify trends in data (Börner and Polley 2014, as cited in Huang et al. 2019). The results reached in this work can aid in the search for a JavaScript library for visualizing data depending on what criteria are present.

Choosing a JavaScript library can be difficult because there are many different factors to consider. Depending on the scenario, some abilities may be more important than others. For one website it may be more important that render time is fast for small datasets than the ability to render large datasets. This work aims to help in the decision of choosing a library when the render time is an important factor. The comparison of libraries is done similarly to Carlström (2018), Ehrling (2019), ElTayeby et al. (2013), and Smatt et al. (2020), however, there are differences between the libraries and datasets used as well as some differences in the methods used.

This work aims to find if dataset size and/or chart type affects the render time of a chart created using a certain library. To reach this aim, a literature search was performed to compile a list of suitable libraries (ApexCharts, Frappe Charts, Google Charts, and TeeChartJS) and chart types (bar- and line charts) on which an experiment was conducted with five datasets of different sizes (between 100 and 27 thousand datapoints). 40 websites are created, and each website's render time is measured using the Lighthouse tool. Lastly, the results are analyzed to answer the research questions.

2 Background

This chapter explains terminology that is relevant to this work to provide context and understanding for the rest of this work.

2.1 Visualization

According to Rau et al. (2018), there are three main reasons to visualize data: exploration, confirmation, and presentation. Exploration is done to find out details about the data easily, confirmation is using the visualization to prove or disprove some theory or hypothesis, and presentation is using data visualization to present findings to someone else. Another reason to visualize data is to make it possible to identify trends in data (Börner and Polley 2014, as cited in Huang et al. 2019). While data visualization dates back to ancient times, the popularity of visualization has become increasingly significant during the last two decades (Rau et al., 2018). According to Rau et al. (2018) this increase in popularity of visualization was due to recent development in hardware, software, and connectivity, allowing more data to be processed more quickly and in higher quality.

Data can be visualized in many different ways depending on the purpose. Visualization can be performed in real-time, as demonstrated by Huang et al. (2019) who used visualization to convey information about the users' emotions back to them by collecting data from a webcamera as well as text input that was then summarized in charts for the user to view in real-time. It can also be used with static data, for example by presenting information collected from tweets in the form of maps, pie- and bar charts as well as a word cloud (Li et al., 2014).

2.2 Charts

A chart is a graphical description of data and is a simple way to show information (Cambridge Dictionary chart, n.d.). A chart can be used for example to show patterns or trends in data that may be hard to see in something like a table. Many websites use charts to display their data. For example, Google Trends uses charts to visualize the trends in Google Search (Google Trends, n.d.). There are several types of charts, including, but not limited to, bar charts, pie charts, line charts and scatter plots.

2.2.1 Bar and column charts

Bar and column charts are typically used when comparing data across categories or to show how a variable changes over time (Szoka, 1982). Bar charts (see Figure 1 and Figure 2) are typically horizontal but can also be vertical while a column chart (see Figure 1) is always vertical.

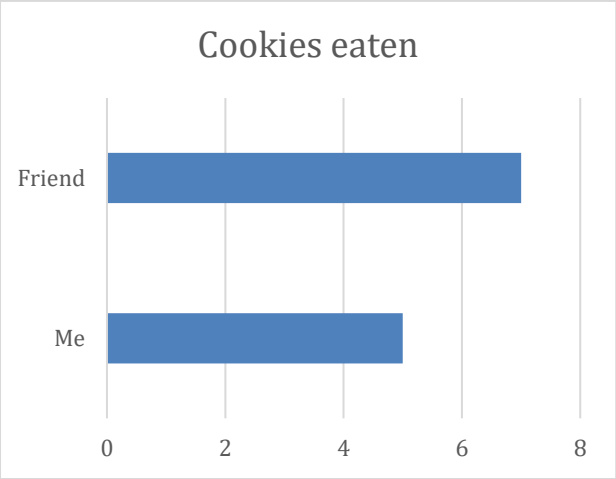

Figure 1 Example of a bar/column chart.

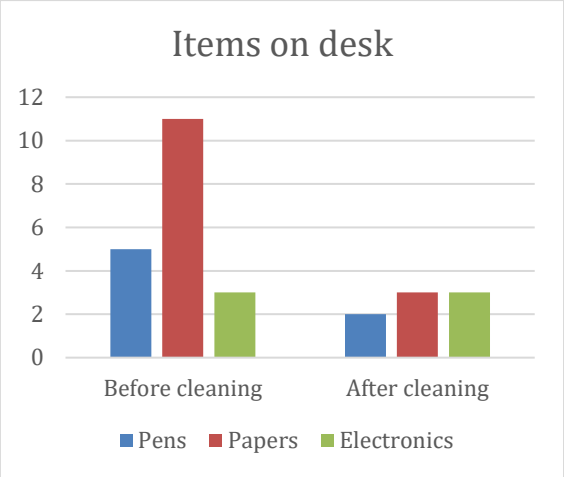

Figure 2 Example of a bar chart.

2.2.2 Pie charts

Pie charts are used when percentages of a whole are presented (Szoka, 1982). Pie charts work well for data that is divided into 6 or fewer categories, if more categories are used the chart can get cluttered. Figure 3 shows an example of a pie chart.

Figure 3 Example of a pie chart.

2.2.3 Line charts

Line charts, also called curve charts, are typically used for showing how data changes over time (Szoka, 1982). Figure 4 shows an example of a line chart.

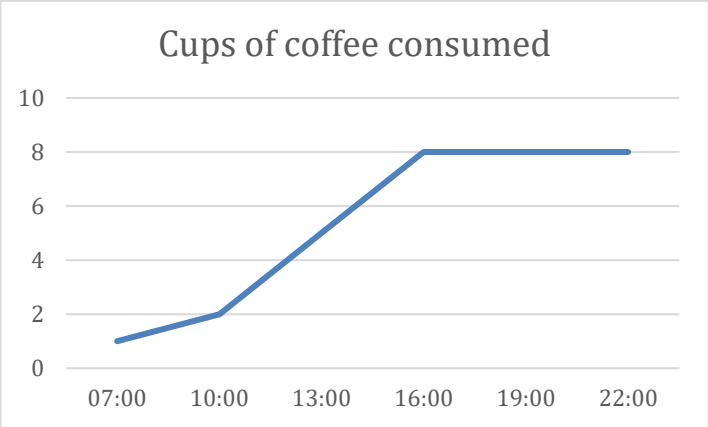

Figure 4 Example of a line chart.

2.2.4 Other charts

Other chart types include, but are not limited to, bubble charts, scatter plots, box plots, and area charts (see Figure 5).

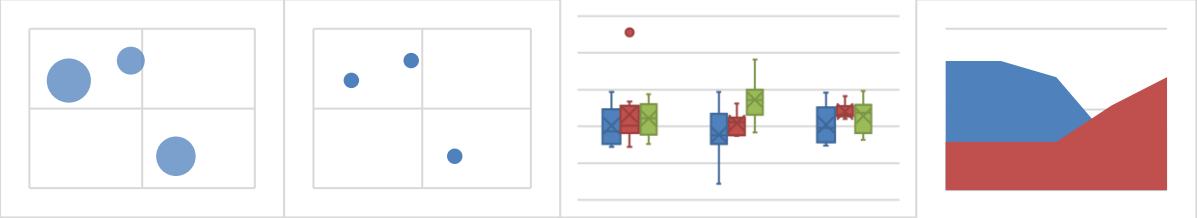

Figure 5 From left to right: Bubble chart, scatter plot, box plot, area chart.

2.3 SMHI

SMHI stands for the Swedish Meteorological and Hydrological Institute. Their tasks include collecting data about the weather in Sweden and providing weather forecasts (SMHI, 2021). On their website, one can download many different datasets, such as the amount of rain that has fallen in a certain period. The dataset used in this project consists of the amount of rain fallen in Skövde, Västra Götaland between 1945 and late 2020 which is then trimmed to the desired size to create five datasets of different sizes.

2.4 JavaScript, HTML

HTML and JavaScript are the programming languages used in this project.

2.4.1 JavaScript

JavaScript is a programming language that is used mainly for scripts on web pages (Mozilla JavaScript, 2021). It is one of the most common programming languages and is used on over 97% of all websites (W3Techs, 2021).

2.4.2 HTML

Hyper Text Markup Language (HTML) is the standard language used when creating websites, invented by Tim Berners-Lee in 1991. (W3Schools HTML, 2021). The latest version is HTML5, which came with features such as support for video and audio to remove the need for additional plugins (HTML, n.d.). One such plugin was Flash, which is not supported since December 2020 (Adobe, 2021).

2.5 JavaScript Libraries

JavaScript libraries are used to re-use existing code (Khan Academy, 2021). Some libraries are free and/or open-source while some require payment to use. Open-source means the source code is available for free and that it is possible to make changes to that code (Cambridge Dictionary open-source, n.d.). When using a JavaScript library, the developer can choose to either download the files and use them locally or get them from a remote group of servers called a Content Delivery Network (CDN). This puts less strain on the server hosting the website and can reduce latency (Mozilla CDN, 2020).

ApexCharts, Frappe Charts, Google Charts, and TeeChartJS are examples of libraries that can be used to make charts to use on a website.

2.5.1 ApexCharts

ApexCharts is open source and licensed under MIT. This means anyone can “use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software” (Open Source Initiative, n.d.). The code is available on GitHub (GitHub apexcharts.js, 2021). According to themselves, their charts are responsive, interactive, dynamic, and offers high performance (ApexCharts, 2021).

2.5.2 Frappe Charts

Frappe Charts is, like ApexCharts, open-source and licensed under MIT. Frappe is a remote software products company that also develops other software products (Frappe, n.d.). The Frappe Charts source code is available on GitHub (GitHub frappe/charts, 2021).

2.5.3 Google Charts

Google Charts is a JavaScript library developed by Google. It is also what Google themselves use when creating charts. Unlike ApexCharts and Frappe Charts, not anyone can contribute to Google Charts. It is maintained by a developer team (Google Charts Support, 2015). The source code is licensed under Apache 2.0, which means distribution is more controlled than with a MIT license. Users can still distribute, modify, and otherwise use the software, but they must comply with the license terms (Apache, 2019). The source code is available on GitHub (GitHub google-chart, 2021).

2.5.4 TeeChartJS

TeeChartJS is another library licensed under MIT. It was originally written by David Berneda in 2012 but is now open source. The source code is available on GitHub (GitHub TeeChartJS, 2020). Even though the library is open source, Steema Software, which maintains the library, offers paid support subscriptions.

2.5.5 jQuery

jQuery is a JavaScript library that simplifies working with JavaScript in several ways (jQuery, 2021). In this project the function jQuery.getJSON is used to get the data from the JSON files where the datasets are stored.

2.6 Render time

Render time refers to the time it takes for a webpage to load visually and be usable and interactable. It is an important factor to consider when developing websites, for several reasons. A study shows that page load time directly impacts the rate at which users leave the site (also called bounce rate) (Solarwinds Pingdom, 2018). The recommendation is to try and keep web page load time under 2 seconds, and the study performed by SolarWinds Pingdom (2018) confirms that this is a good idea. After 2 seconds 6% of users have left the site, increasing to 11% after 3 seconds and 38% after 5 seconds (Solarwinds Pingdom, 2018). Google's search engine also ranks pages based on load time (Google Search, n.d.). To conclude, a fast website is important for several reasons.

2.6.1 Lighthouse

Lighthouse is a tool that is used to measure and improve the quality of a web page. Lighthouse can be run from the command line, as a Node module, or from Chrome DevTools which is done in this project. Lighthouse generates a report on the webpage with many different metrics as well as giving the website a score to measure the performance (see Figure 6) as well as giving pointers on how to improve the web page (Google Developers Lighthouse, 2021). The metric that will be studied for this work is "Time to interactive" since this reflects the render time, which is the time it takes until a user can interact with a website. Other metrics, such as First Contentful Paint, Speed Index among others are not discussed in this work.

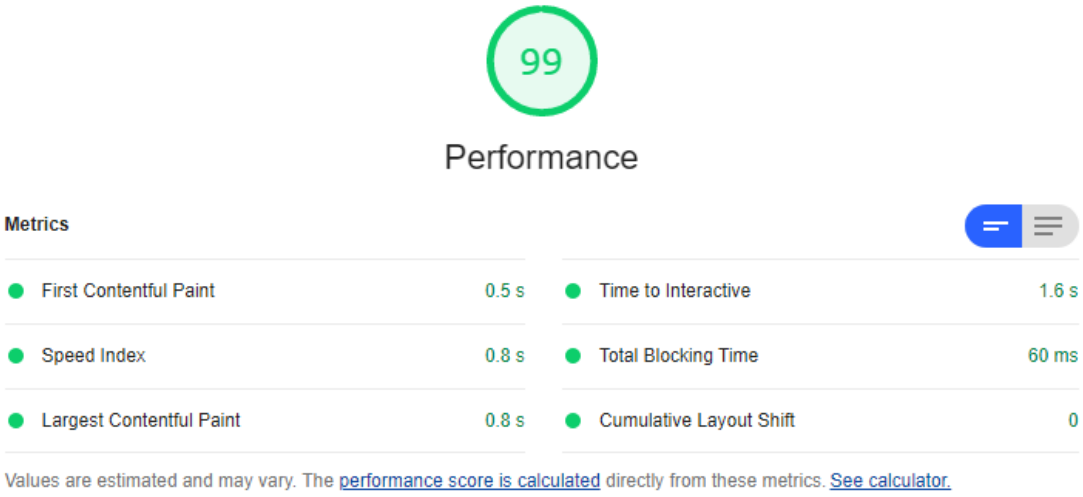

Figure 6 Example of output from Lighthouse.

3 Problem

To avoid creating a new software solution to display a chart on a website, which can be time-consuming and complicated, an existing solution can be found with a JavaScript library. There are a lot of libraries to choose from and deciding between the sometimes seemingly similar libraries can be difficult. Render time is an important aspect to consider when building a website, and overall render times can depend on the render times of the charts present of the websites. Thus, choosing the correct library can be crucial to ensure the website loads quickly so it can gain as well as keep users.

This study aims to determine if the size of the dataset and/or the type or chart affects the render time when comparing a set of JavaScript libraries for visualization as well as determining if there is a difference in render time scaling between the libraries if the dataset size affects the render time. This work only considers static data that does not change, therefore the time to convert the data is not accounted for.

3.1 Research questions and hypotheses

Research questions:

RQ1: When comparing a set of JavaScript libraries for visualization, does the size of the dataset affect the render time for any library? If so, is there a difference between the libraries in render time scaling depending on the dataset size?

RQ2: When comparing a set of JavaScript libraries for visualization, does the chart type affect the render time for any library?

Hypotheses:

H1.1: The dataset size will affect the render time for all libraries.

H1.2: There will be no difference in render time scaling between the libraries depending on the dataset size.

H2: Chart type will not affect render time for any of the libraries.

3.2 Objectives

Objective 1 includes searching for and selecting JavaScript libraries as well as chart types to include in the experiment.

Objective 2 includes creating websites for each library, chart type, and dataset size.

Objective 3 includes measuring the render time of each of the websites several times and calculating an average render time for each website.

Objective 4 includes analyzing the results gathered from objective 3 to answer the research questions.

4 Related work

Several other studies are comparing the performance of JavaScript libraries in similar ways to this work. There are also examples of how visualization can be used both with and without JavaScript libraries.

Carlström (2018) performed an experiment comparing the performance and scalability of three JavaScript libraries (Google Charts, Chart.js, and Plotly.js). The author aimed to answer what library had the fastest render time as well as if the dataset size impacted the render time. This was done by creating several websites with different libraries whose render times were measured. Chart types that were included were scatter plots, bar- and line charts. The datasets used were between 30 points and 73 thousand points. The conclusion was that Chart.js rendered the fastest while Google Charts rendered the slowest and that an increase in dataset size resulted in longer render times. It was not considered how or if the chart types affected the render times.

Ehrling (2019) performed an experiment comparing the JavaScript libraries D3 and Google Charts to explore which was more suited for visualizing log data from webservers along with which library had the fastest render time. This was done, similarly to Carlström (2018), by creating websites, but Ehrling (2019) measured render times with the TamperMonkey tool. The conclusion that was reached was that D3 was the fastest on all accounts and thus the most suited for visualizing log data. It was also concluded that the size of the dataset, chart type, number of categories as well as interactivity all have an impact on the render time.

ElTayeby et al. (2013) performed a case study comparing the JS libraries D3 and Highcharts for use in a certain application where data needs to be visualized in real-time. The focus was mainly on the different features of the libraries but render time was also briefly mentioned. D3 and Highcharts showed to be useful in different ways, however, Highcharts was slower when rendering a large number of datapoints. The work was not finished when this article was written, and they mention that they will continue to work to improve the speed of Highcharts when using large datasets.

Lee et al. (2014) wished to visualize a large dataset and thus performed a study comparing a set of tools for data visualization, including Google Charts and D3.js. Lee et al. (2014) evaluated tool complexity, performance, and memory use among other attributes. They concluded that each tool worked well in different settings. They concluded that D3 worked well with large amounts of data and was still memory efficient, but it had a steep learning curve. Additionally, Google Charts was easier to work with, but limited in speed because of communication overhead with the server.

Smatt et al. (2020) performed a study on 29 participants to see if Google Charts or D3.JS provided the most user-friendly interface as well as which was more effective and efficient. A data visualization tool was deemed effective if it is the right tool used in the right way for a certain task. Google Charts was deemed more effective since the participants performed fewer errors when working with Google Charts and participants reported Google Charts was easier to use than D3.JS. Efficiency was measured by looking at how fast participants completed a task using the libraries. D3.JS was deemed more efficient than Google Charts, however, Google Charts was perceived as more efficient by the participants because of its ease of use and

aesthetically pleasing interface. Smatt et al. (2020) did not consider the render time of each libraries' visualizations in their work.

Huang et al. (2019) developed a software prototype called Sens-e-motion that captures a user's emotion. This is done through facial expression recognition, text analysis as well as eye-tracking. The data gathered is then visualized for the user in real-time using ApexCharts. The motivation behind choosing ApexCharts over other libraries is however not discussed by Huang et al. (2019). Nevertheless, ApexCharts was suitable for real-time visualization of dynamic data as shown by Huang et al. (2019).

Li et al. (2014) developed the Social Media Workbench, an open-source tool for visualizing data from social media such as Twitter. The motivation behind this is that they found JavaScript libraries too difficult to work within their scenario. In their paper, they describe two use cases showing how this workbench can be used. First, they present the tool visualizing geo-tags of tweets on a map and pie charts. They also present the workbench being used to collect complaints about heavy rain in Rotterdam, Netherlands. The data is presented in a bar chart, a word cloud as well as a data table. Li et al. (2014) describe that this workbench is under development and that they intend to expand it in the future. They wish to "reduce the need for code replication throughout the research community" by providing this tool.

5 Method and Approach

The method and approach of this work are described in this chapter.

5.1 Method

The method and alternative methods are described in this section.

5.1.1 Literature search

An informal literature search is done to find libraries to include in the experiment. Libraries are found through blogs, GitHub, and articles, but mainly not through scientific articles. To ensure as many libraries as possible are included in the initial list, all sources are considered. An informal search was preferred since few actual research articles exist on this topic. The easiest way to find JavaScript libraries is through internet searches leading to blog posts and the like. The quick emerging of new libraries hinders scientific articles from being a good source of information on this topic because of the time it takes to publish such an article. JavaScript libraries are also often rapidly updated, which can make cause information in an article to be outdated while the information available on sites like GitHub contains the latest information.

5.1.2 Experiment

An experiment was chosen for the method of this project. According to Wohlin et al. (2012), experiments are used when control is an important factor, which is the case for this project. The independent variables (chart type, library, and dataset size) need to be controlled to ensure that the dependent variables (render time) really depend on the independent variables and not some other factor. Wohlin et al. (2012) also describe that an experiment is about evaluating a hypothesis, which is precisely what this project also aims to do.

5.1.3 Alternative methods

Alternative methods are a case study and a survey.

A case study was not selected because there is no case to study. There would also be less control over variables which is not ideal for this project. There is a possibility to study existing websites to answer the research questions, but there would be too many variables out of control to be able to draw any conclusions from the results.

A survey would not yield results useful to answer the research questions. A questionnaire or an interview could be conducted asking experts on the topic, however, that would not yield the desired quantitative data that is needed to ensure that conclusions can be drawn. Surveys are also used to understand a population (Wohlin et al., 2012), which is not the case here.

5.2 Approach

The approach is described this section.

5.2.1 Literature search

An initial, informal literature search is performed to create a list of libraries from which a few can be selected to be used in the experiment.

Libraries are selected from the list using certain criteria:

- Free and open source.
- Easy to use.
- No dependencies on other libraries.

The criteria “Easy to use” is discussed in more detail under “Discussion”.

Libraries may then be filtered further depending on how many libraries fit these criteria since the time frame of this project only allows for a small number of libraries to be compared.

To select what chart types to use, research is done to find out what types of charts are supported by each of the selected libraries. These chart types will then be included in the experiment, if the number is not too large in which case further filtering will need to be done. In the end, less than 5 libraries are selected for the experiment.

5.2.2 Experiment

Websites are created for each library, chart type, and data sample size containing only a single chart. As few settings as possible are configured to make the comparison as fair as possible.

The dataset from SMHI is in CSV format, which gets converted into different JSON files depending on what the library supports. The time to do this conversion is not included in the render time since this work focuses on static, not streaming, data. The data is added from JSON files with the getJSON function available in the jQuery library.

The render time is measured using Lighthouse. The metric “Time to interactive” is focused on. The measurement is done 5 times per website and the mean value is calculated for the results.

5.2.3 Analysis

The mean render time value of each website is analyzed and compared to the other websites to draw conclusions and answer the research questions.

6 Results

The results are presented in this chapter.

6.1 Selecting libraries and chart types

To complete objective 1, Libraries were found through internet searches, blog posts, and similar sources. The initial list contained 67 libraries (see Appendix A). After removing libraries that did not fit the criteria (free and open-source, easy to use, no dependencies on other libraries), 4 libraries remained that fulfilled all the criteria, represented by the green cells in Table 1.

Table 1 Final selection of libraries.

	Free (full version)	Open source	Easy enough	Enough Documentation (at first glance)	No dependencies
Google Charts	Yes	Yes	Yes	Yes	Yes
ApexCharts	Yes	Yes	Yes	Yes	Yes
Frappe charts	Yes	Yes	Yes	Yes	Yes
TeeChartJS	Yes	Yes	Yes	Yes	Yes

A quick test run was done on these four libraries to ensure they work well. This test only consisted of copying the respective libraries' example graphs that were provided in the documentation and ensuring it rendered properly on a local webserver using Python.

To choose what charts to include in the experiment a search was done to find what chart types were supported by the selected libraries. This was done by inspecting the documentation of the libraries.

Bar/Column-, line-, and pie charts were the chart types that were supported by all the libraries (see Table 2). A decision was made to exclude pie charts, so bar- and line charts were selected

for the experiment. Pie charts were excluded because they are used for visualizing other types of data and using the same dataset for all graphs was deemed more important for this study.

The dataset from SMHI was chosen because it provided data in a format that was easy to work with for free.

Table 2 Supported chart types.

	Pie	Bar/ Column	Line	Bubble	Scatter
Google Charts		Supported	Supported	Supported	Supported
ApexCharts		Supported	Supported	Supported	Supported
Frappe Charts		Supported	Supported	Not Supported	Line w/ only dots
TeeChart JS		Supported	Supported	Supported	Supported

6.2 Creating the websites

In this section it is presented how the websites were created in preparation for the experiment to complete objective 2.

6.2.1 Pilot

The first set of websites was created to test the basic functionality of the libraries and to familiarize with the available settings and so on. Four websites were created with bar charts and a dataset of about 100 datapoints. These were run on a local webserver using Python. All websites rendered a chart successfully.

6.2.2 Preparation of websites

40 websites were created in total. Each website contained only the chart rendered by the library. There was one website for each combination of one out of five datasets, one out of four libraries, and one out of two chart types. The source code for these websites is available on GitHub (GitHub BachelorDegreeProjectIT2021, 2021).

All libraries allowed for some customization of the created charts. These settings were not changed unless necessary (such as changing the width of the chart if not all the data could fit). Animation was disabled for the libraries that supported it, and if possible, the x-axis labels were rotated and placed in such a way that they were readable. The code was kept as similar as possible for all websites using the same library.

6.2.3 Preparing the data

The dataset from SMHI is in a CSV file with 27 thousand datapoints. To make the smaller datasets, this file was simply trimmed to the desired size (100, 5 thousand, 10 thousand, 20 thousand, and roughly 27 thousand datapoints). The CSV format is not directly supported by all libraries, and because of this, the CSV file was converted to JSON files using an online converter[1]. The different libraries required different formats of JSON files, all of which could

[1] https://www.convertcsv.com/csv-to-json.htm

be acquired from this converter. These JSON files as well as the CSV files are available on GitHub (GitHub BachelorDegreeProjectIT2021, 2021). The JSON files are then imported with the getJSON function available in the jQuery library.

6.3 Measuring the render times

In this chapter it is presented how the render times of the websites were measured to complete objective 3.

6.3.1 Pilot

To decide what tool to use for measuring the render time, a small pilot test was done. The tools that were considered were the function console.time, the performance tab in Google Chromes DevTools (it was later discovered that this tool does not measure render time, only runtime performance) and the Lighthouse tool. This resulted in Lighthouse being selected as tool since it provided the wanted results. The function console.time did not consider any rendering that might happen after the code is done running and thus did not provide an accurate measurement for when the page is interactive. Lighthouse takes this into account and measures the time until a user can interact with the page.

Another pilot test was done to ensure Lighthouse provided the desired measurements, become familiar with the tool, and to gather some preliminary results. The results of this pilot test can be seen in Figure 7. The confidence interval is not included in this chart since each measurement was only done once, making confidence intervals irrelevant for the pilot.

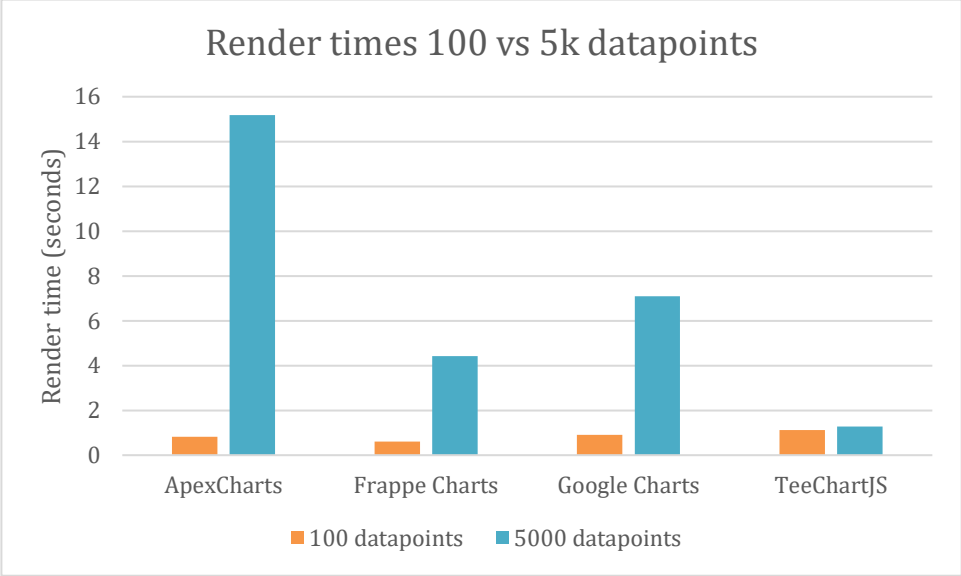

Figure 7 Results of pilot test.

6.3.2 Data collection

The render time measurements were done on a separate machine that was only running the local webserver and an Incognito Chrome window (to avoid any data being saved to the cache which could alter the results). Lighthouse was run five times on each website and the mean was calculated. The overall result of these measurements is shown in Figure 8 and Figure 9 and the raw measurements can be found on GitHub (GitHub BachelorDegreeProjectIT2021, 2021). The measurements for each library are discussed in further detail under “Analysis”.

Striped bars in Figure 9 and Figure 8 represent cases where the chart did not render for some reason. This is discussed further under “Discussion”.

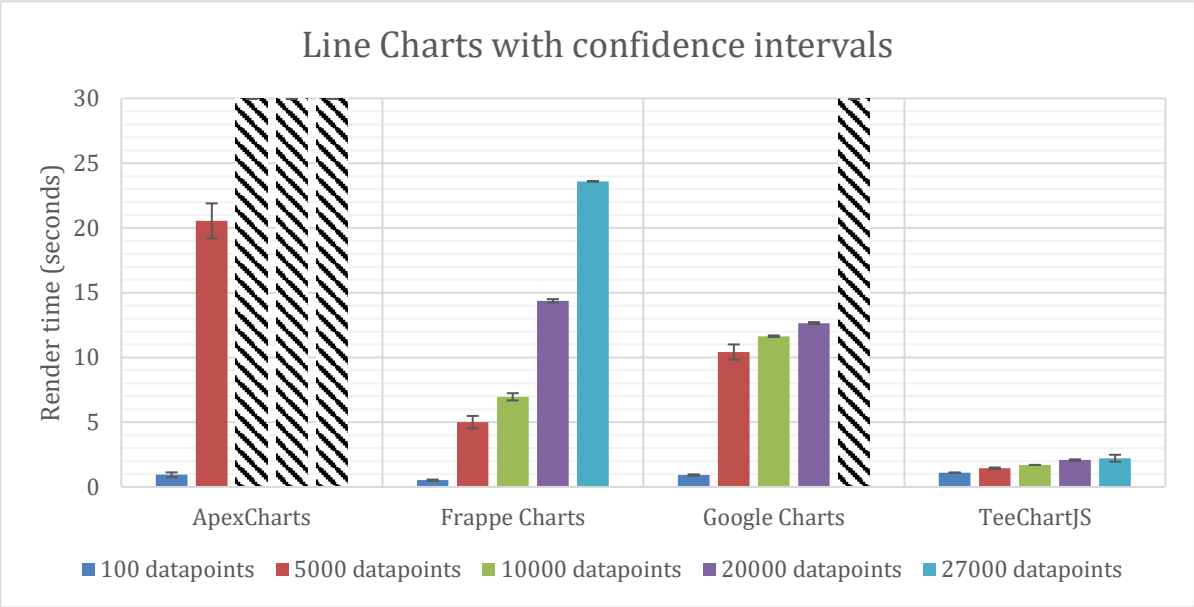

Figure 8 All line chart measurements.

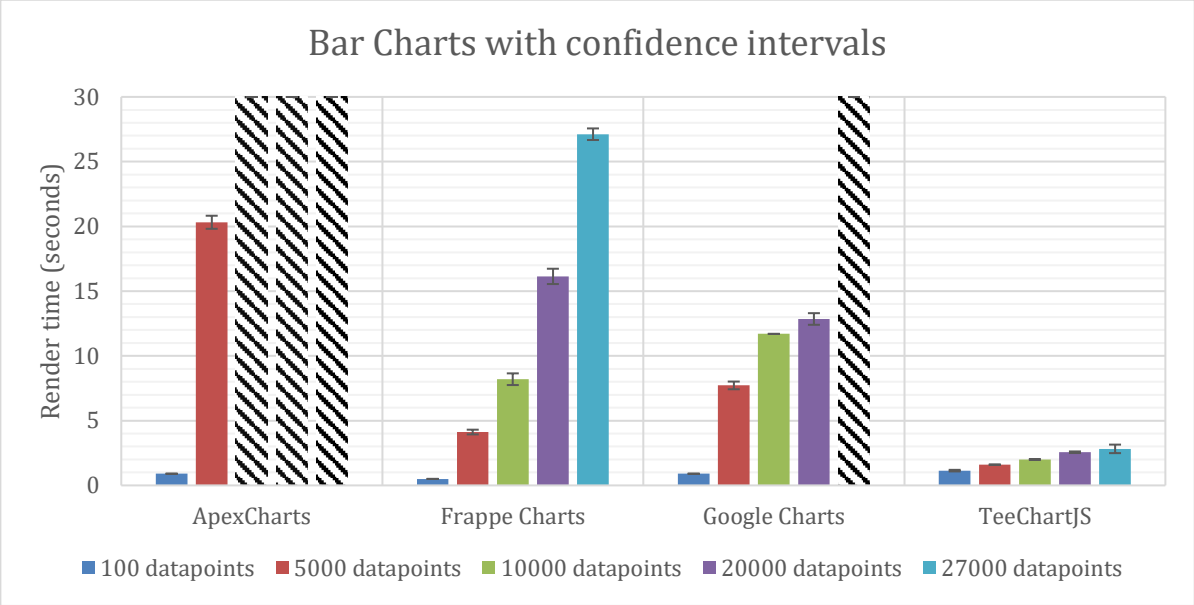

Figure 9 All bar chart measurements.

6.4 Analysis

To answer the research questions and complete objective 4, the results were analyzed. To ensure the difference in render time was significant, the confidence intervals were analyzed, and ANOVA-tests were performed where necessary. All measurements used in the calculations as well as confidence intervals and more detailed data is available on GitHub (GitHub BachelorDegreeProjectIT2021, 2021).

6.4.1 ApexCharts

Table 3 shows the result of the render time measurements for ApexCharts. For line charts with 5 thousand datapoints, the chart only rendered two out of five times. The other three times Lighthouse reported that the page stopped responding. This could mean it took too long to load or that something else went wrong. ApexCharts was unable to render pages containing charts with 10 thousand datapoints and above. For 10 thousand datapoints, Lighthouse reported that the page did not paint any content and thus the measurement was terminated. When the dataset was 20 thousand and 27 thousand datapoints it was not possible to start Lighthouse at all because the page did not respond and/or crashed.

Figure 10 shows the result of the measurements with ApexCharts. For ApexCharts the dataset size clearly affected the render time. The confidence intervals do not overlap; thus, the difference is significant. However, when it comes to the comparison of chart types the render time does not seem to be affected. An ANOVA-test gives the p-value of 0,61 with 95% confidence which means there is no significant difference in render time between line- and bar charts when the dataset is 100 datapoints. The result is similar for the 5-thousand-point dataset, where the p-value is 0,54 with 95% confidence and thus there is no significant difference in render time between line-and bar charts when the dataset is 5 thousand points.

Table 3 ApexCharts render time measurements, in seconds.

Line					**Bar**				
100	**5000**	**10000**	**20000**	**27000**	**100**	**5000**	**10000**	**20000**	**27000**
0,8	20,4	-	-	-	0,9	20,7	-	-	-
0,8	-	-	-	-	0,9	20,2	-	-	-
0,9	-	-	-	-	0,9	19,6	-	-	-
1	-	-	-	-	0,9	20,4	-	-	-
1,2	20,7	-	-	-	0,9	20,7	-	-	-

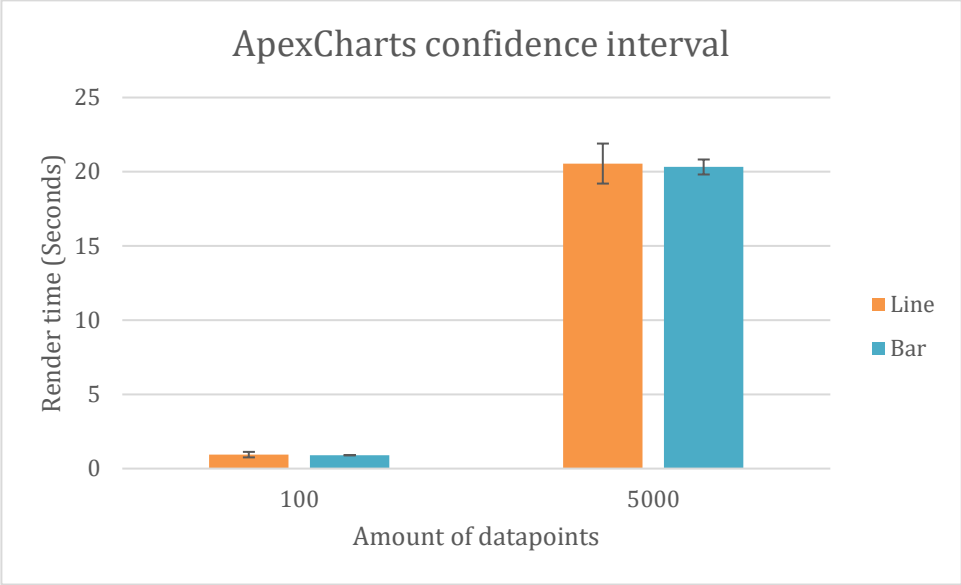

Figure 10 Results of measurements with ApexCharts.

6.4.2 Frappe Charts

Table 4 shows the result of the render time measurements for Frappe Charts. Line charts with 20 thousand datapoints failed to render once, and line charts with 27 thousand datapoints failed to render four out of five times. This was due to Lighthouse reporting PROTOCOL_TIMEOUT, which means Lighthouse has been waiting for Chrome to respond to a command for more than 30 seconds.

Figure 11 shows the result of the measurements with Frappe Charts. The size of the dataset affects the render time and the larger the dataset the longer the render time. The confidence intervals do not overlap; thus, the difference is statistically significant. When comparing the chart types, there is a difference in render time for all dataset sizes except the 100-point dataset. An ANOVA-test of that category gives a p-value of 0,35 with 95% confidence which proves there is no statistically significant difference in render time between chart types for such a small dataset. Line charts render faster for the datasets 10 thousand and larger, but bar charts rendered faster with the 5-thousand-point dataset. This is discussed further under “Discussion”.

Table 4 Frappe Charts render time measurements, in seconds.

Line					Bar				
100	**5000**	**10000**	**20000**	**27000**	**100**	**5000**	**10000**	**20000**	**27000**
0,5	4,5	6,6	14,5	-	0,5	3,9	7,8	15,7	27,7
0,5	4,6	6,8	14,3	-	0,5	4	8,1	16,7	27,3
0,6	5,1	7,1	14,3	-	0,5	4,2	7,9	15,6	26,8
0,5	5,3	7,2	14,4	23,6	0,5	4,2	8,8	16	26,7
0,5	5,5	7,1	-	-	0,5	4,3	8,4	16,7	27,1

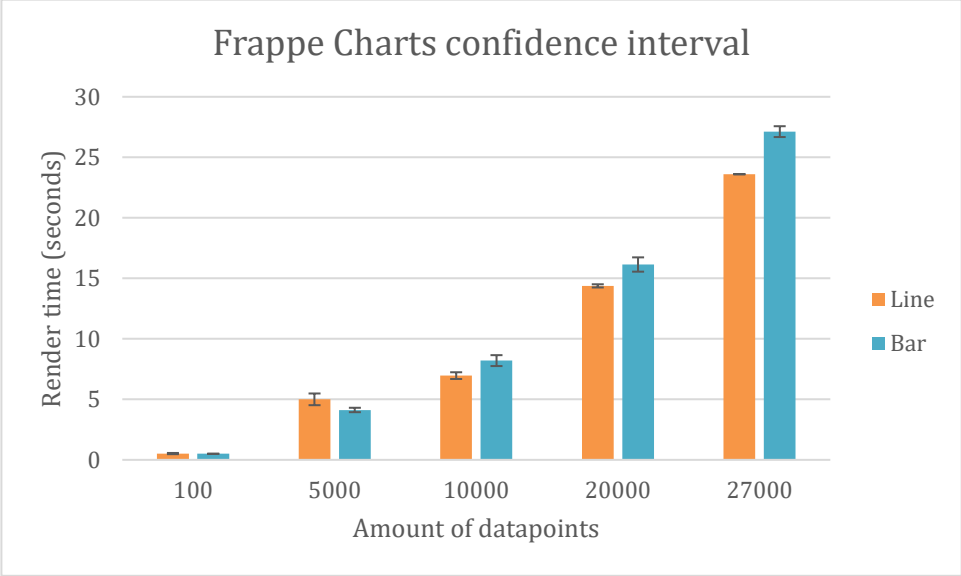

Figure 11 Results of measurements with Frappe Charts.

6.4.3 Google Charts

Table 5 shows the result of the render time measurements for Google Charts. Google Charts failed to render once with line chart with 20 thousand datapoints and three times out of five with bar chart with 20 thousand datapoints. No chart rendered with 27 thousand datapoints. Lighthouse reported this was due to the page not responding.

Figure 12 shows the result of the measurements with Google Charts. The confidence intervals do not overlap between the dataset sizes; thus, it can be concluded that dataset size affects the render time for Google Charts. For the 5-thousand-point dataset, bar charts rendered faster and the chart type did affect the render time. With a dataset of 100 points, an ANOVA-test gave a p-value of 0,35 with 95% confidence which means there is not a statistically significant difference in render time between the different chart types with 100 datapoints. For the 10-thousand-point dataset, an ANOVA-test gave a p-value of 0,040 with 95% confidence which proves there is a statistically significant difference in render time between the chart types with 10 thousand datapoints. Similarly, with the 20-thousand-point dataset, an ANOVA-test gave a p-value of 0,020 with 95% confidence which proves there is a statistically significant difference in render time between the chart types with 20 thousand datapoints.

Table 5 Google Charts render time measurements, in seconds.

Line					Bar				
100	**5000**	**10000**	**20000**	**27000**	**100**	**5000**	**10000**	**20000**	**27000**
1	10,3	11,6	12,6	-	0,9	7,9	11,7	12,9	-
0,9	9,8	11,6	-	-	0,9	8,1	11,7	12,8	-
0,9	10,1	11,6	12,7	-	0,9	7,5	11,7	-	-
0,9	11,1	11,7	12,6	-	0,9	7,5	11,7	-	-
0,9	10,8	11,7	12,7	-	0,9	7,6	11,7	-	-

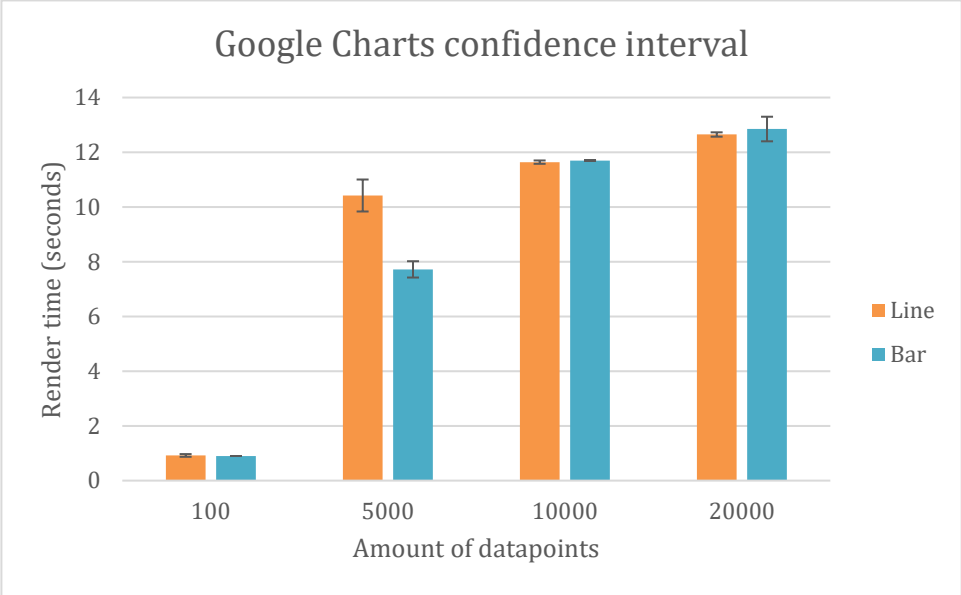

Figure 12 Results of measurements with Google Charts.

6.4.4 TeeChartJS

Table 6 shows the result of the render time measurements for TeeChartJS. Figure 13 shows the result of the measurements with TeeChartJS. TeeChartJS was the only library that rendered a chart every single time.

For the 100, 5k, and 10k datasets there is no overlap of confidence intervals and thus there is a difference in render time based on dataset size.

However, the confidence intervals overlap for line charts of 20k and 27k and bar charts of 20k and 27k, respectively. An ANOVA-test of the bar chart measurements gave a p-value of 0,090 with 95% confidence which means there is not a statistically significant difference in render time for bar charts with datasets of 20- and 27 thousand datapoints. An ANOVA-test of the line chart measurements gave a similar result with a p-value of 0,23 with 95% confidence which means there is again no statistically significant difference in render time for line charts with datasets of 20- and 27 thousand datapoints. To conclude, there is no statistically significant difference in render time between the 20k line charts and 27k line charts, and there is no statistically significant difference in render time between the 20k bar charts and 27k bar

charts. There is however a difference in render time between the 20k bar and line charts as well as the 27k bar and line charts, respectively.

There is also an overlap between the 27-thousand-point line chart and the 10 thousand datapoint bar chart. An ANOVA-test gave a p-value of 0,14 with 95% confidence which means there is yet again no statistically significant difference in render time between these two categories.

Finally, when comparing the two chart types with a 100-point dataset, an ANOVA-test gave a p-value of 0,14 with 95% confidence which means there is no statistically significant difference in render time between chart types with 100 datapoints.

Table 6 TeeChartJS render time measurements, in seconds.

Line					Bar				
100	**5000**	**10000**	**20000**	**27000**	**100**	**5000**	**10000**	**20000**	**27000**
1,1	1,4	1,7	2,1	1,8	1,2	1,6	2	2,6	2,3
1,1	1,4	1,7	2,1	2,3	1,1	1,6	2	2,5	2,9
1,1	1,5	1,7	2,1	2,3	1,2	1,6	2	2,5	2,9
1,1	1,4	1,7	2	2,4	1,1	1,6	2	2,6	3
1,1	1,5	1,7	2,1	2,3	1,1	1,6	2	2,6	3

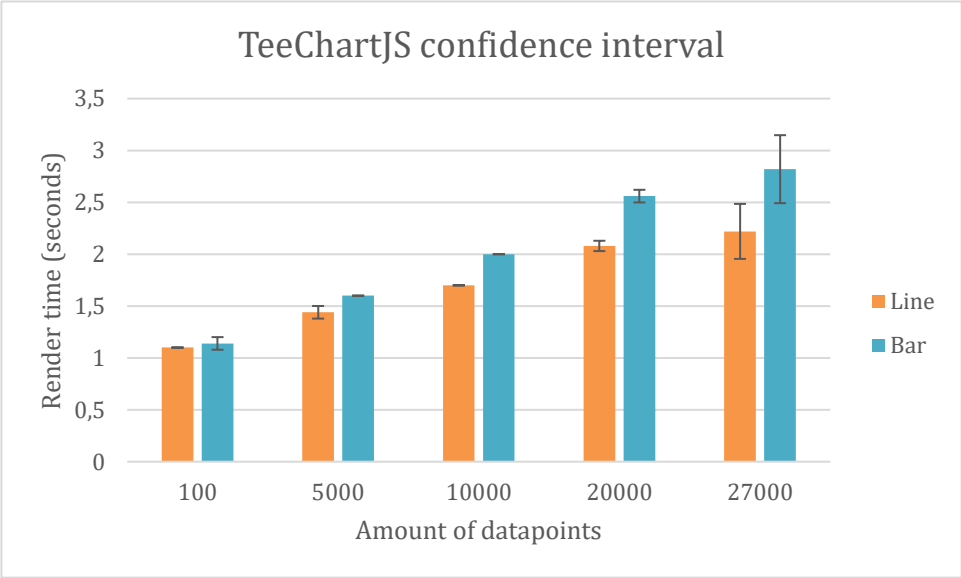

Figure 13 Results of measurements with TeeChartJS.

6.5 Conclusion

RQ1: *When comparing a set of JavaScript libraries for visualization, does the size of the dataset affect the render time for any library? If so, is there a difference between the libraries in render time scaling depending on the dataset size?*

From these results, it can be concluded that in most, but not all cases the dataset size does affect the render time. The exception is the library TeeChartJS, where dataset size did not always affect the render time. The results also show that all libraries scale differently. Thus, both H1.1 and H1.2 are rejected.

RQ2: *When comparing a set of JavaScript libraries for visualization, does the chart type affect the render time for any library?*

From these results, the conclusion can be drawn that the chart type does affect the render time in most, but not all, cases. The exceptions were for the library ApexCharts with the 100- and 5 thousand-point datasets as well as Frappe Charts and Google Charts when using the 100-point dataset. It was however not consistent which chart type was fastest, this requires further study. This means H2 was rejected.

7 Discussion

The results of this study, ethics as well as threats to validity are discussed in this chapter.

7.1 Results and discussion

The results of the experiment are discussed in this section.

7.1.1 Selecting libraries and chart types

The criterion “Easy to use” was evaluated by looking at the library’s documentation and in some cases a small pilot test. If the code seemed too complicated to understand in about an hour, it was ruled out. This decision was made due to the short time frame for this project. This was the case for the D3 library (D3.js, 2020). It was deemed that it would be too time-consuming to try and learn to use D3 in the short amount of time that was available.

Some libraries were removed even though they technically fit all the criteria. The reason for this was that some other factor made that library unfit for this experiment. This could be for example that the library was “abandoned”, that the website was unreachable, or that no installation instructions were found. Since there was a limited time frame, no time could be wasted on libraries that required extra effort to be functional. More specific information on which libraries were excluded and why can be found on GitHub (GitHub BachelorDegreeProjectIT2021, 2021).

The decision to only include libraries that could be used with a CDN and did not require a download was done as the list of libraries that fit the criteria was too large (around 14 libraries) for the time frame and needed to be trimmed down even more. This decision also simplified the work since using a CDN only requires adding a line of code, while downloading each separate library would require more effort and thus would have taken time away from the actual experiment, which is the focus of this work. For a real-life application, using a CDN may make usage of a library easier and can improve the user experience since the user may be located closer to the CDN than the webserver hosting the website. It does however introduce an added uncertainty since it is possible for a CDN to be unavailable which could then render certain features or the whole website unusable. There are also concerns about security if dealing with sensitive user information, in which case a CDN should not be used since the JavaScript library could be modified to collect that data without alerting the users of the CDN.

The reason pie-, bar-, line-, bubble-, and scatter charts were the only chart types included in the selection is that these were the ones most supported by the libraries. They are also simple and somewhat easy to interpret and work well with large datasets (apart from pie charts, which did not suit this experiment). Pie charts were eventually excluded from the experiment, this was due to it not being a good fit with the dataset that was selected. It was deemed more important to consistently use the same dataset than include pie charts. This decision also saved some time that could be put towards the actual measurements instead of website creation.

Since JavaScript libraries, as well as chart types were chosen, Objective 1 was completed.

7.1.2 Creating the websites

The decision to use the smallest dataset of 100 datapoints in the pilot was to ensure it was possible to check that all datapoints were rendered as they should. The smaller dataset was also faster to convert to the correct JSON file, which saved some time.

As few changes as possible were made to the settings on each website. Settings could include chart size, font, colors, and so on. This was to ensure that changes to the settings altered the render time as little as possible. Animation was disabled for the libraries that supported it to ensure the animation did not make the render time longer than necessary. This decision was made after noticing that ApexCharts charts render time was a few seconds longer than necessary due to the animation taking several seconds, and Lighthouse included that time in its measurements.

The reason that the largest dataset is roughly 27 thousand datapoints is that that is the size of the dataset from SMHI in its original state.

Completion of the websites consequently completed Objective 2.

7.1.3 Measuring the render times and analysis

All websites were run on Python's SimpleHTTPServer. It is simple to install and use, which means more time could be allocated towards the actual measurements.

To choose a tool for measuring the render times, a small test was done. The function console.time was deemed unfit because it seemed to only measure the time it took to run the code, which was not reflecting render time.

Since the data is prepared before measuring the render times of the websites, that time is not included. The results of this work are aimed at those who want a chart for existing data that will not change often in the future. It is not aimed at those looking for a chart that updates in real-time or uses streaming data in any way. This means that the work needed to convert the data to the correct format was tolerated since it was assumed that it would only have to be done once. For this reason, this time was also not included in the render time measurements.

The pilot test to measure render times was done informally. This means that it was performed without ensuring other factors did not skew the results. It was done in a Chrome incognito window, but other processes were running on the same machine at the time. The render time was also only measured once. Regardless, the pilot test provided some preliminary results and helped refine the measurements process in preparation for the real measurements.

These initial results seem to show that there is a difference in render time when the dataset size changes for at least one of the libraries. This result was promising and a good motivation to continue the experiment.

For the actual measurements, the websites were run on a local webserver using Python on a separate machine that had no other applications running. Still, there is no guarantee that no processes were running in the background that went unnoticed. Measurements were done five times to ensure the result represented the normal case and not an outlier.

When the dataset got larger, the measurements could not be completed for several different reasons. The paged stopped responding before Lighthouse could even be opened, Lighthouse reported that the page stopped responding, Lighthouse gave the error protocol timeout

(meaning Chrome did not respond within 30 seconds), or Lighthouse reported that the page did not paint any content (which seems to happen if there is no activity for about 30 seconds). This of course means that any website that took longer than about 30 seconds to render was deemed a failure. There is reason to believe that with a different tool, one could measure these longer render times as well, however, there might not be a reason to. Since 38% of users leave the site after 5 seconds of load time (Solarwinds Pingdom, 2018), one can only imagine how few would wait over 30 seconds.

ApexCharts offers a demo on their website where they claim that rendering 25 thousand datapoints can be done in less than a second. That is wildly different from the results of this experiment. Perhaps it could be because the library was not downloaded and used locally but through a CDN. Another reason could be that the demo on the website uses hard-coded datapoints or some other way of retrieving the data rather than getting them from a JSON file which is done in this experiment. Other reasons could be some difference in hardware or settings (this probably would not cause such a big difference though).

From the results of the experiment, one can conclude that in this case, ApexCharts is not a good fit for larger datasets. Even with 5 thousand points, the chart took over 20 seconds to load. For the 100-point dataset, however, render time was consistently under 2 seconds which is Google's recommendation for page load time.

Regarding **Frappe Charts**, one interesting result from this experiment was that bar charts consistently rendered slower than line charts except for 100 datapoints (where it was about the same), and for 5 thousand datapoints, were bar charts rendered quicker than line charts. What caused this is hard to say and could be a topic of future work.

Frappe Charts was the library with the shortest render time for the 100-point dataset with both line- and bar charts. However, it did not scale very well since render time quickly increased with larger datasets. From these results, one can conclude that Frappe Charts may be a good choice of a library if one wishes to visualize a dataset that is smaller than 5 thousand datapoints since that would keep render time reasonably long (if using the same methods as this experiment for data preparation, settings and so on).

When using **Google Charts**, as the dataset grew larger not all the data could fit on the chart, and it was cut short. This was solved by increasing the width, which strangely enough also moved the chart on the page. However, according to the documentation, the chart is supposed to adjust itself as to not require this. There was not enough time left to research this problem further, otherwise, it might have gotten solved. Additionally, similarly to Frappe Charts, bar charts are rendered faster than line charts only with the 5-thousand-point dataset. Once again, further research is needed to conclude why this happens.

The render times did not wary as much for Google Charts as it did for ApexCharts and Frappe Charts. When one went from 100 to 5 thousand datapoints it grew (from about 1 second to around 9 seconds), but between 5 thousand and 20 thousand it only increased by around 4 seconds. This alone could suggest that Google Charts may be a good fit for larger datasets, however, the opposite is proved when the 27-thousand-point dataset is observed since that did not render a single time. To conclude, Google Charts is a good fit for a small dataset (around 100 datapoints) but larger datasets (5 thousand to 20 thousand) are slower and with datasets around 27 thousand it does not render at all.

TeeChartJS was the library with the shortest render times for all but the 100-point dataset, where it was the slowest (keeping under 2 seconds still).

From these results, one can conclude that in this case, TeeChartJS is the best choice when visualizing a large dataset, since it did not fail to render once, and all render times were under 3 seconds.

Since the render times for every website were measured as well as analyzed, objectives 3 and 4 are complete.

7.1.4 Conclusion

Every library performed well for the 100-point dataset, keeping render time below 2 seconds (see Figure 14). Someone looking for a library to use on a website to visualize a dataset around 100 points could choose either library. Frappe Charts was however the fastest out of all the libraries with 100 datapoints for both line- and bar charts.

For the larger datasets (5 thousand to 27 thousand datapoints), TeeChartJS performed the best out of all the libraries. This library would be the best fit for someone looking to visualize a larger dataset using methods similar to those used in this work.

The completion of every objective as well as the answering of the research questions means the aim of this work was fulfilled.

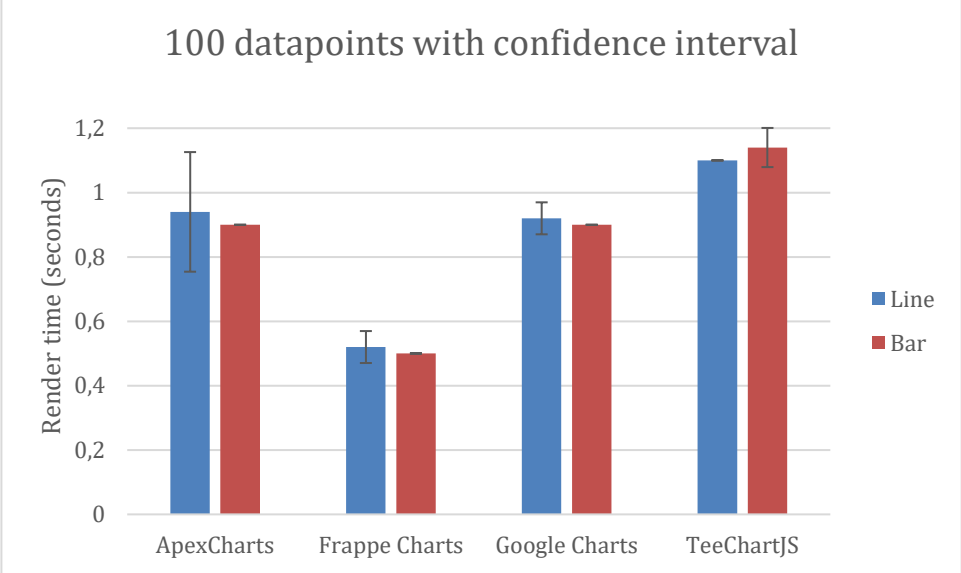

Figure 14 Results of measurements with the 100-point dataset.

7.2 Value to society

Visualization is something important for the average person when using a webpage. If data were only displayed in tables, it would be much harder if not impossible to get a good overview and see patterns in the data. Charts have many different uses, and many websites have at least one chart presenting something. Apart from making data easier to read and take in, it can also serve as a tool to convince users a product is superior to another, as well as be used to present results in research like this work does. On the flip side, it is also possible to manipulate charts to convey a certain message that is favorable to the creator of the chart. By manipulating things like axis labels, bar width, colors, or even using the wrong type of chart, one can make two

charts using the same data look vastly different and thus conveying different, sometimes misleading, messages. This may be relatively harmless in some cases but can also be harmful in cases like political elections or other hotly debated topics.

Selecting the right library can have an impact on the website's user experience. A chart that loads very slowly may be frustrating and the user might even leave the site, resulting in negative consequences for the developer. While a chart that renders fast will convey information to the user fast, which in turn results in a better user experience.

7.3 Related work

Carlström (2018) only considered the variables library and render time in the work. Even though both line- and bar charts were created for the study, they were not compared against each other. The results Carlström (2018) reached are slightly different from those reached by this work. Google Charts had a faster render time overall in Carlström's (2018) work, even when the dataset was larger. A possible cause is that Carlström (2018) only measured the time to draw the chart, not including the time to prepare the data. Other factors that could contribute to different results are that Carlström (2018) used XML files instead of JSON files for the data, which may have contributed to the difference. There could also be some difference in source code, and it is also possible that the library itself has been updated since Carlström's (2018) work. It should also be noted that Carlström (2018) measured the render time using the function performance.now which may have given a different result.

The experiment performed by Ehrling (2019) was similar in that the same variables were looked at (library, dataset size, and chart type), there were however several differences in how the experiment was performed compared to this work. To start, the render times were measured using TamperMonkey while this work uses Lighthouse. Ehrling (2019) did not look at the time until the website was interactive but instead chose to look at initial render time as well as render time while interacting with the charts. Second, while JSON-files were used similarly to this project, the conversion from the raw format to JSON was done by a converter created by the author, and the conversion time was included in the render time. This difference is because different datasets are used in this and Ehrling's (2019) work. The conclusion was reached that library, dataset size, chart type among other factors do affect render time. These results are similar to those of this work which concluded that render time is affected by dataset size and chart type in most cases. Another difference between this work and the work by Ehrling (2019) is the render times measured. The render times measured in this work are significantly longer than those of Ehrling (2019). This could once again be the result of several different factors.

ElTayeby et al.'s (2013) work was much more focused on the actual features of the libraries rather than their performance, even though that was also considered. Their goal was also to find a library that suited their needs for a certain application, while this experiment's goal was broader (to simply compare the libraries and see if dataset size or chart type affects render time). Perhaps a project like that of ElTayeby et al. (2013) could find the results of this experiment useful when searching for a JavaScript library to use for a certain task. Since the work by ElTayeby et al. (2013) is not finished at the time of writing it is unclear whether HighCharts or D3 best fit their application and why. At this time both libraries have advantages as well as challenges in ElTayeby et al.'s (2013) case.

Lee et al. (2014) focused on large datasets. They also looked more closely at user-friendliness, something that this project did not consider, as well as other visualization tools that are not JavaScript libraries. It is possible to recreate this work and extend it to include both user-friendliness and other tools, given more time. Lee et al. (2014) concluded that different tools were fitting for different applications, and a similar conclusion was drawn in this work. Depending on what size dataset is used, some libraries are better suited than others.

Smatt et al. (2020) focused only on user-friendliness and did not consider render time or performance in any way. They also did a task survey rather than an experiment, which was more suited for their project. This work did not consider user-friendliness or how users will perceive and use the finished charts, thus the results between this work and Smatt et al.'s (2020) work cannot be compared.

Huang et al. (2019) work with data that is updated in real-time, unlike this work which works with static data. In their work the data needed to update in real-time to provide a good and meaningful user experience. In this work however, the data did not need to update in real time since the dataset was already complete and would not change in the future. Thus, the results reached in this work may not be helpful in works similar to Huang et al.'s (2019).

The tool developed by Li et al. (2014) is an example of a data visualization tool that is not a JavaScript library. They mention in their paper that JavaScript libraries can be complicated to use and that in some cases the time and effort cannot be justified. Therefore, they developed their own tool that instead uses a "simple domain-specific language" to enable the users to focus on the charts rather than the programming. However, this work is aimed at those that prefer or must use JavaScript libraries for their chart needs and for some reason can or do not want to use another tool. The tool by Li et al. (2014) is also developed for social media and not any other datasets. This work is aimed to apply to different types of datasets if they are somewhat similar to those used in the experiment.

7.4 Ethics and threats to validity

Ethics and threats to validity are presented in this section.

7.4.1 Ethics

No confidential or otherwise sensitive data has been used in this work. All code used for the websites used in the experiment was written by the author. There may be errors, in that case, they are not intentional. All data and code are available on GitHub (GitHub BachelorDegreeProjectIT2021, 2021) for replicability. However, the fact that the libraries are open source could result in a recreation of this experiment yielding different results, since the libraries may be updated at any time.

It is not the author's intent to rank one library as "better" than another overall, a comparison is simply made to see what libraries best suit certain cases. There may be cases that are not considered in this work where a library might be the best fit even though it was the worst fit for a case in this work.

7.4.2 Internal validity

As mentioned, the websites were measured on a separate machine with only the necessary applications running. However, the measurements were performed at different times of the day and there is no guarantee that no background processes were running. This may have

caused small fluctuations in the measurements. To rule this out, the experiment would have to be repeated several more times over a longer period and more precautions would have to be taken to ensure other factors are not affecting the result. However, because of the low variation of the results, it is probable that all measurements were either affected similarly or the affecting factors had such a small impact that it did not affect the results significantly.

The websites' render time was only measured five times each which could make the results less reliable. However, the distribution is small, and no category had a standard deviation bigger than 0,5 seconds (all raw measurements and standard deviations, as well as confidence intervals, can be found on GitHub (GitHub BachelorDegreeProjectIT2021, 2021)). Because of this in addition to the fact that the measurements were performed manually, it was deemed that five measurements were enough for this work. To ensure correct results, the measurement process can be automated using a tool such as TamperMonkey (TamperMonkey 2018) which gives the possibility of doing hundreds of measurements with little effort.

7.4.3 External validity

Since all websites were run on a local webserver these results may not represent how the websites would behave in a different setting. There was no other traffic, and the websites did not contain any content beyond the charts themselves. However, even though the render times may differ the ranking of the libraries should stay similar in a different setting and the answers to the research questions are still valid.

Since all libraries were implemented using CDN the latest version of the library was always used, ensuring the results can be applied to a real-life setting.

The browser used in this work was Google Chrome, and the results may not be the same if using other browsers. To confirm this one would have to perform the same experiment on different browsers as well.

7.4.4 Construct validity

There are more variables to consider when choosing what library to use other than render time. Some might look for fewer lines of code or more ability to customize. This work focuses on render time as that is important for every website.

7.4.5 Conclusion validity

All results were analyzed with confidence intervals and where needed, ANOVA-tests. This ensures that the conclusions drawn from the measurements are correct and sound.

The variation between the five measurements done on each website is low (less than 1 second), meaning that the results are accurate.

7.5 Future work

With more time, this experiment could quite easily be expanded to include more libraries, more chart types, and even more variables to measure such as lines of code, customization settings available, memory usage, user interface. Something similar to Smatt et al.'s (2020) work could be done where user-friendliness is considered as well as render time and other factors. This could aid in enhancing the user experience when it comes to viewing and using the charts. It could also be interesting to do something like ElTayeby et al.'s (2013) work where actual features of libraries were looked at more closely, but to also include more factors. This

could also enhance the user experience by ensuring features that users seek are present where needed. One could even compare the JavaScript libraries to other tools with the same purpose, similar to Lee et al. (2014). This could uncover any better alternatives to using JavaScript libraries to visualize data.

The method of fetching the data may affect the render time making it a topic for further research. This work uses the getJSON function in the jQuery library, but there are many ways to fetch the data and some ways may yield different results. Since neither Carlström (2018) or Ehrling (2019) use the function getJSON there is a possibility that is what is leading to the slower render times measured in this work. This is something that should be explored further by measuring different stages of render time and comparing them to see what stage of the rendering process is taking the longest. This could also lead to results showing what method of fetching data is the most effective, which would also be interesting to decrease render time further.

The tool used to measure the render times may also yield different results depending on how it measures render time. This project uses Lighthouse, which deemed measurements over about 30 seconds a failure. Another tool may not do this and could give more insightful results regarding the charts that took the longest to render. There may be interest in doing a comparison of the available tools to find which are more accurate and suited for different projects.

In this work, all measurements were done manually which led to a quite small number of measurements in total. To get possibly more accurate results it could be interesting to use some tool to automate the render time measurement process to gather more results more efficiently. There are several tools available for this, one being TamperMonkey (2018) that allows the user to write scripts that execute on a website. Gathering more results would lead to higher reliability to confirm that the conclusions reached in this work were the correct ones.

One specific aspect that could be researched further is the reason bar charts rendered faster than line charts with Frappe Charts and Google Charts using the 5-thousand-point dataset, when the opposite was true for the other larger datasets. This could be done by doing a similar experiment like that in this work but focusing solely on Frappe Charts and Google Charts with several datasets that are around 5 thousand datapoints. This could be interesting to someone looking to optimize their usage of these libraries.

References

Adobe 2021, Adobe Flash Player EOL General Information Page, viewed 23 April 2021, https://www.adobe.com/se/products/flashplayer/end-of-life.html

Apache 2019, Apache License Version 2.0, viewed 23 April 2021, https://www.apache.org/licenses/LICENSE-2.0

ApexCharts 2021, APEXCHARTS.JS Modern & Interactive Open-source Charts, viewed 23 April 2021, https://apexcharts.com/

Cambridge Dictionary n.d., chart, viewed 10 March 2021, https://dictionary.cambridge.org/dictionary/english/chart

Cambridge Dictionary n.d., Open-source, viewed 28 April 2021, https://dictionary.cambridge.org/dictionary/english/open-source

Carlström, A. (2018). A comparative study between different JavaScript libraries for visualization: Performance measurements of JavaScript libraries for statistical graphs and diagrams(Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-15491

D3.js 2020, Data-Driven Documents, viewed 27 April 2021, https://d3js.org/

Ehrling, J. (2019). Visualization of web server log data in diagrams with interaction : A comparison of visualization techniques with D3.js and Google Charts based on rendering times (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-17121

ElTayeby, O., John, D., Patel, P., & Simmerman, S. (2013, October). Comparative case study between D3 & Highcharts on Lustre metadata visualization. In 2013 *IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)* (pp. 127-128). IEEE.

Frappe n.d., Modern, Open Source SVG Charts, viewed 9 March 2021, https://frappe.io/charts

GitHub 2020, TeeChartJS, viewed 9 March 2021, https://github.com/Steema/TeeChartJS

GitHub 2021, BachelorDegreeProjectIT2021, viewed 29 April 2021, https://github.com/a18jonpe/BachelorDegreeProjectIT2021

GitHub 2021, frappe/charts, viewed 23 April 2021, https://github.com/frappe/charts

GitHub 2021, google-chart, viewed 23 April 2021, https://github.com/GoogleWebComponents/google-chart

GitHub 2021., apexcharts.js, viewed 23 April 2021, https://github.com/apexcharts/apexcharts.js

Google Charts 2015, Google Charts Support, viewed 23 April 2021, https://developers.google.com/chart/interactive/support

Google Developers 2021, Lighthouse, viewed April 23 2021, https://developers.google.com/web/tools/lighthouse

Google Search n.d., How Search algorithms work, viewed 25 March 2021, https://www.google.com/intl/en_uk/search/howsearchworks/algorithms/

Google Trends n.d, Explore what the world is searching, viewed 9 March 2021, https://trends.google.com/trends

HTML n.d, HTML5 Basics For Everyone Tired Of Reading About Deprecated Code, viewed 9 March 2021, https://html.com/html5/

Huang, W., Jayaraman, P. P., Morshed, A., Blackburn, S., Redpath, C., Guerney, T., ... & Mui, R. (2019, July). Sens-e-motion: Capturing and visualising emotional status of computer users in real time. In *2019 23rd International Conference in Information Visualization–Part II* (pp. 96-99). IEEE.

jQuery 2021, What is jQuery?, viewed 23 April 2021, https://jquery.com/

Khan Academy 2021, What's a JS library?, viewed 4 March 2021, https://www.khanacademy.org/computing/computer-programming/html-css-js/using-js-libraries-in-your-webpage/a/whats-a-js-library

Lee, S., Jo, J. Y., & Kim, Y. (2014, October). Performance testing of web-based data visualization. In 2014 *IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 1648-1653). IEEE.

Li, W., Eickhoff, C., & De Vries, A. P. (2014, August). Interactive summarization of social media. In *Proceedings of the 5th Information Interaction in Context Symposium* (pp. 312-315).

Mozilla 2020, CDN, viewed 23 April 2021, https://developer.mozilla.org/en-US/docs/Glossary/CDN

Mozilla 2021, JavaScript, viewed 4 March 2021, https://developer.mozilla.org/en-US/docs/Web/JavaScript

Open Source Initiative n.d, The MIT License, viewed 23 April 2021, https://opensource.org/licenses/MIT

Rau, R., Bohk-Ewald, C., Muszyńska, M. M., & Vaupel, J. W. (2018). Introduction: Why Do We Visualize Data and What Is This Book About?. In *Visualizing Mortality Dynamics in the Lexis Diagram* (pp. 1-4). Springer, Cham.

Smatt, C., Pratt, R., Abegaz, T., & Dobbs, A. (2020). TOWARDS EFFECTIVE CUSTOMER DATA VISUALIZATION: DATA-DRIVEN DOCUMENTS (D3. JS) VS. GOOGLE CHARTS. *QRBD*, 207.

SMHI 2021, About SMHI, viewed 23 April 2021, https://www.smhi.se/en/about-smhi

Solarwinds Pingdom 2018, Does Page Load Time Really Affect Bounce Rate?, viewed 9 March 2021, https://www.pingdom.com/blog/page-load-time-really-affect-bounce-rate/

Szoka, K. (1982). A guide to choosing the right chart type. *IEEE Transactions on Professional Communication*, (2), 98-101.

TamperMonkey 2018, TamperMonkey, viewed 21 May 2021, https://www.tampermonkey.net/

W3Schools 2021, HTML Introduction, viewed 9 March 2021, https://www.w3schools.com/html/html_intro.asp

W3Techs 2021, Usage statistics of JavaScript as client-side programming language on websites, viewed 23 April 2021, https://w3techs.com/technologies/details/cp-javascript

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Appendix A - Original list of libraries

Chartist.js
FusionCharts
Dygraphs
Chart.js
Google Charts
Highcharts
Flot
D3.js
n3-charts
NVD3
Ember Charts
Morris.js
C3.js
Rickshaw
Cubism.js
Plottable.js
Canvas.js
JSCharting
amCharts
ZingChart
KOOLCHART
EJSCharts
uvCharts
Plotly.js
TOAST UI Chart
AnyChart
Smoothie Charts
DC.js
YUI charts
ApexCharts
billboard.js
canvasXpress
charts 4 php
cytoscape.js
dc
DevExtreme
DHTMLX charts
dimple
dojo charting
echarts
factmint charts
flotr2
frappe charts
gRaphael

Jenscript
jqplot
jqxchart
lightningchart js
metricsgraphics
muzejs
nextcharts
olapcharts
pizza amore
pluscharts
reactiveChart
rgraph
shield ui
syncfusion
teechart js
vaadin charts
vancharts
visjs
webcharts
webix js charts
xcharts
zoomcharts
victory