Evaluating and Improving Web Performance Using Free-to-Use Tools

University of Oulu
Faculty of Information Technology and Electrical Engineering / Degree Programme in Information Processing Science
Master's Thesis
Matias Kinnunen
May 9, 2020

Abstract

Fast website loading speeds can increase conversion rates and search engine rankings as well as encourage users to explore the site further, among other positive things. The purpose of the study was to find and compare free-to-use tools that can both evaluate the performance (loading and rendering speed) of a website and give suggestions how the performance could be improved. In addition, three tools were used to evaluate the performance of an existing WordPress site. Some of the performance improvement suggestions given by the tools were then acted upon, and the performance of the website was re-evaluated using the same tools. The research method used in the study was experimental research, and the research question was "How to evaluate and improve web performance using free-to-use tools?" There were also five sub-questions, of which the first two related to the tools and their features, and the last three to the case website.

Eight free-to-use web performance evaluation tools were compared focusing on what performance metrics they evaluate, what performance improvement suggestions they can give, and six other features that can be useful to know in practice. In alphabetical order, the tools were: GTmetrix, Lighthouse, PageSpeed Insights, Pingdom Tools, Test My Site, WebPageTest, Website Speed Test (by Dotcom-Tools) and Website Speed Test (by Uptrends). The amounts of metrics evaluated by the tools ranged from one to fifteen. The performance improvement suggestions given by the tools could be put into three categories, meaning that the suggestions largely overlapped between the tools. All tools except Lighthouse were web-based tools.

The performance of the case website was evaluated using GTmetrix, PageSpeed Insights and WebPageTest. On desktop, the performance was in the high-end range though varying between the three tools, and on mobile, the performance was noticeably slower due to the challenges of mobile devices (e.g. lower processing power compared to desktop computers) and mobile networks (e.g. higher latency compared to broadband connections). The common bottlenecks based on the suggestions given by the three tools seemed to be lack of using a CDN (Content Delivery Network), serving unoptimized images and serving large amounts of JavaScript. The results of the performance re-evaluation were mixed, highlighting the importance of carefully considering each performance improvement suggestion.

The main takeaways of the study for practitioners are to use multiple tools to get a wide variety of performance metrics and suggestions, and to regard the suggestions and relative performance scores given by the tools only as guidelines with the main goal being to improve the time-based performance metrics.

Keywords
web performance, loading speed, rendering speed, performance evaluation, performance improvement

Supervisor
PhD, University Lecturer Antti Siirtola

Abbreviations

API	Application Programming Interface
CDN	Content Delivery Network
CPU	Central Processing Unit
CSS	Cascading Style Sheets
DOM	Document Object Model
FCP	First Contentful Paint
FMP	First Meaningful Paint
FP	First Paint
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HVAC	Heating, Ventilation and Air Conditioning
JPEG	an image file with the extension “.jpg” or “.jpeg”
JS	JavaScript
PNG	an image file with the extension “.png”
SEO	Search Engine Optimization
TTFB	Time to First Byte
TTI	Time to Interactive
URL	Uniform Resource Locator

Contents

1. Introduction

Computers and phones have been becoming faster and more performant for decades (Waldrop, 2016). Likewise, broadband and mobile internet speeds have increased throughout the world over the past few years (McKetta, 2019). However, websites have become increasingly heavier (Rodman, 2015), and it is common for websites to take several seconds to fully load (An, 2018). This is worrying, since slow performance can lead to users abandoning the loading page (An, 2018; Galletta, Henry, McCoy, & Polak, 2004; Nah, 2004; Nielsen, 1999b; Rose, Lees, & Meuter, 2001), worse search engine ranking (Singhal & Cutts, 2010; Wang & Phan, 2018) and other negative consequences.

The purpose of the study was to explore and compare free-to-use tools that can both evaluate the performance of a website and give suggestions about how to improve the performance. An existing WordPress site was used as a case for the experimental part of the comparison. The performance of the site was evaluated using the tools, improved by following the suggestions provided by the tools, and finally re-evaluated after doing the performance improvements.

The study is a follow-up to the author's previous work (Kinnunen, 2016), which is a narrative literature review about the effects of website loading speed on user satisfaction. The focus of this follow-up study was more on the practical side. Ideally, the process presented in the study can be followed to evaluate and improve the performance of other websites as well.

The research question in the study was:

How to evaluate and improve web performance using free-to-use tools?

The term "web performance" was used in the study to describe the loading and rendering speed of a webpage. Web performance does not end there, though, as it can also mean for example the smoothness of animations and scrolling, or the quickness of responding to taps and clicks (Walton, n.d.).

Five sub-questions are listed below. The first two sub-questions relate to the tools, while the rest of the sub-questions relate to the case website. The research method used in the study is experimental research (Gustedt, Jeannot, & Quinson, 2009; Runeson & Höst, 2008; Tedre & Moisseinen, 2014). Prior research was reviewed using the narrative literature review method.

1. What free-to-use tools currently exist for evaluating and improving web performance?
2. How do the tools compare in terms of features?
3. What performance results do the tools initially (before interventions) report when evaluating the performance of the case website?
4. What are the biggest performance bottlenecks of the case website reported by the tools?
5. How does the performance of the case website improve after implementing the performance improvements suggested by the tools?

The study provides a detailed, up-to-date comparison of eight free-to-use tools that can both evaluate web performance and suggest how the performance could be improved. Three tools were used on an existing website and the results were analyzed. The author is not aware of prior academic studies in which web performance has been both evaluated and improved like in this study. While the details of the experiment are not generally applicable outside the study since the experiment targets just a single website and all websites are different, the details of the comparison of the eight tools as well as the details how the three tools were used on the case website can be utilized both in future research and by practitioners to evaluate and improve the performance of other websites.

The structure of the study is as follows. Chapter 2 covers the literature review of prior research. Chapter 3 describes the research method used in the study and the case website. Chapter 4 describes how the free-to-use web performance evaluation tools were searched for and the comparison results. Chapter 5 documents the results of the performance evaluations and improvements done for the case website. Chapter 6 consists of discussion and implications of the study. Chapter 7 ends the study with a conclusion.

2. Prior Research

The literature review of prior research was done using the narrative literature review method. Chapter 2.1 and Chapter 2.2 as well as parts of Chapter 2.3 are largely based on the author's previous work (Kinnunen, 2016), which is a narrative literature review about the effects of website loading speed on user satisfaction. Since then, new relevant literature has also been released.

The chapter is divided into four sub-chapters. Chapter 2.1 covers prior literature about the importance of web performance; Chapter 2.2 about suitable website loading speeds; Chapter 2.3 about different web performance metrics; and Chapter 2.4 about ways and tools to evaluate web performance.

2.1 Importance of web performance

Literature about human-computer interaction has been published since the late 1960s (Galletta et al., 2004). For example, Miller (1968) contrasted human-computer interaction with a conversation between two humans. If the other party does not respond within a few seconds, it breaks the flow of the conversation. Too long delays also interfere with human short-term memory and thinking continuity. (Miller, 1968.)

Lightner, Bose and Salvendy (1996) conducted a pilot survey about the likes, dislikes and difficulties of using internet. "Speed of data access" was the most common dislike and the second most common difficulty of using internet. Conversely, "searching for specific information" was the second most common dislike and the most common difficulty. In the main survey, the authors then focused on the topic of searching difficulties since they thought that the survey method is not as suitable for the topic of website speeds. (Lightner et al., 1996.)

Sears, Jacko and Borella (1997) compared the effects of short, medium and long website loading delays (mean delays of 575ms, 3,500ms and 6,750ms, respectively). Test subjects were asked to perform information retrieval tasks and then to rate four qualities of the website: ease of locating information, organization of the information, quality of the information and navigation problems. When the webpage contained text and graphics, short delays resulted in favorable responses. When the webpage contained only text, long delays resulted in favorable responses. Medium delays resulted in mixed responses. The authors interpret the results that users may prefer multimedia websites, but do not like the increased delays of retrieving the multimedia. The study was "the first that establishes sensitivity to delays of different lengths in the context of Internet use" (p. 354). (Sears et al., 1997.)

Ramsay, Barbesi and Preece (1998) showed users seven different webpages with different webpage loading delays ranging from two seconds to two minutes, and then asked the users to rate the pages by answering a few questions. The most quickly loading pages were rated as significantly more interesting than the most slowly loading pages. The authors conclude that it is because frustration and boredom grow as the delay grows, while motivation reduces. Slowly loading pages were also rated as significantly less scannable, possibly due to impatience, as the scannability ratings themselves did not differ. (Ramsay et al., 1998.)

Nielsen's (1999a) list of "top ten mistakes" in web usability contains four mistakes with the score of "very severe." One of the very severe usability mistakes is "slow download times," (Nielsen, 1999a) which reduces trust and "always" causes a loss of traffic (Nielsen, 1999b). Nielsen (1999c) reviewed websites of ten big corporations as well as ten popular websites for violations of his "top ten mistakes" list. "Slow download times" was the most common design mistake, present in 84% of the reviewed websites. The second most common design mistake, "non-standard link colors," was present in just 17% of the websites. (Nielsen, 1999c.) The studies by Galletta et al. (2004), Nah (2004) as well as Rose et al. (2001) also confirm that increased loading times increase the probability that the user abandons the loading page.

Gehrke and Turban (1999) surveyed 130 e-commerce consumers and potential consumers about their opinions of the relative importance of these five attributes of websites: page loading speed, business content, navigation efficiency, security and marketing/consumer focus. Page loading speed was rated as the most important (21.9% of respondents), albeit barely – the other percentages were 20.0%, 20.1%, 18.1% and 19.9%, respectively. (Gehrke & Turban, 1999.)

Hoxmeier and DiCesare (2000) tested the effects of different response times of a browser-based software application. Participants were split into five groups, so that each group had a different response time rate: 0s, 3s, 6s, 9s or 12s. The app was presented to each participant individually, and at the end, the participant was surveyed. As the response times increased, user satisfaction and the perceived power of the system decreased. However, some delays positively affected the perceived ease of use and learnability of the system. (Hoxmeier & DiCesare, 2000.)

According to the literature review by Galletta et al. (2004), quickly loading websites are preferred over slowly loading websites by users and "seem to encourage exploration and decrease the penalty for making errors" (p. 13). The results of the authors' laboratory experiment also indicate that increased loading times negatively affect users' performance, attitudes and behavioral intentions. As the website becomes more familiar to a user, loading speeds become more important and affect the user's attitudes and intentions more. (Galletta et al., 2004.)

Aberdeen Group (2008) surveyed over 160 organizations regarding optimizing web application performance. On average, a one second delay in response times was seen to reduce customer satisfaction by up to 16%, page views by up to 11% and conversions by up to 7%. (Aberdeen Group, 2008.) According to the study by Hernández, Jiménez and Martín (2009), a slow website loading speed might keep the user from purchasing and completing a transaction online.

An experiment by Google shows that slowing down their search results pages by 100 to 400 milliseconds reduced the number of searches performed by a user between 0.2% and 0.6% (Brutlag, 2009). In other words, an extra delay of less than half a second had a measurable impact. The longer a user was exposed to the extra delay, the less the user performed searches. In addition, users who were exposed to the extra delay performed fewer searches on average even after the experiment. (Brutlag, 2009.) Website loading speed has affected the website's ranking in Google's search results since 2010 (Singhal & Cutts, 2010).

Rempel (2015) performed a case study of an internal business web application whose users were motivated and encouraged to report poor performance via a service desk. As performance reduced, the amount of performance complaints increased. Conversely, as performance increased, the amount of performance complaints decreased. After a certain level of performance was attained, no new performance complaints were made

by the users. Rempel considers the number of performance complaints as a good indicator of user satisfaction. (Rempel, 2015.)

Google analyzed 11 million mobile ads' landing pages and reported insights about their loading speeds, bounce rates and other data points (An, 2018). The majority of the sites were found to take several seconds to load and to be bloated in terms of the page weight (in bytes) and the number of elements on the page. A deep neural network predicted that the probability of a mobile site visitor abandoning the loading page increases as the page load time increases, and, similarly, that the probability of conversion drops as the number of elements on the page increases. (An, 2018.)

2.2 Suitable website loading speeds

Tolerable website loading speeds can be defined from various perspectives, for example through changes in the user's satisfaction or frustration (Nah, 2004). Other perspectives include the user's intentions to visit or not to visit the page again; the user's perceived waiting times; the quality or accuracy of the user's performance; and user behavior, such as whether the user abandons the page. In her own study, Nah defined tolerable waiting time as "the amount of time users are willing to wait before giving up on the download of a Web page" (p. 154). (Nah, 2004.)

Response times of over two seconds interfere with thinking continuity and short-term memory and cause the human-computer interaction to not feel conversational (Miller, 1968). Response times of about half a second result in the most effective conversational flow. (Miller, 1968.) The "2-second rule" in web design has been heavily cited and has a strong face validity (Galletta et al., 2004).

Response times of about tenth of a second make the user feel like the system reacts instantaneously (Nielsen, 1993). When response times are about one second at most, the user's flow of thought stays uninterrupted. (Nielsen, 1993.) When response times exceed one second, the user begins to use the system in a more restricted manner; for example, they might try fewer options or visit fewer pages (Nielsen, 1995). When response times are about ten seconds at most, the user's attention stays focused on the dialogue (Nielsen, 1993). If the response times are longer than that, the user should be given feedback of the estimated delay, especially if the delay is not consistent. This way the user can do something else while waiting. (Nielsen, 1993.) The response time limit of keeping the user's attention could be acceptable to be raised from ten seconds to fifteen seconds on the web, because web users are accustomed to slowness (Nielsen, 1995, 1996). However, Nielsen (1997) later stated that a maximum of ten seconds should be the minimum goal for response times, again referring to the ability to stay focused.

In the study by Hoxmeier and DiCesare (2000), where the participants were presented with a response time rate of 0s, 3s, 6s, 9s or 12s, user satisfaction was at the highest in the first group (0s response time rate), stayed high and quite uniform in the next three groups, and dropped in the last group (12s response time rate). Probability of not using the application again was the highest in the last group. The authors state that users appear to tolerate response times of about ten seconds, and that it is not possible to draw more accurate conclusions from the study due to the 3-second intervals between the participant groups. (Hoxmeier & DiCesare, 2000.)

The tolerable waiting time for a website may depend on various factors, such as the user's level of experience and age, expected content and download time of the page, and feedback about the wait (Nah, 2004). Similarly, in the study by Galletta, Henry, McCoy

and Polak (2006a), Mexican subjects were more patient towards longer website loading times than American subjects (p. 29).

In the study by Nah (2004), the tolerable waiting time for most users doing simple information retrieval was about two seconds. The tolerable waiting time can be prolonged by providing feedback of longer waiting times, for example by showing a loading bar. In the study, 15 seconds was the upper limit of how long users were willing to wait when they were provided no feedback. (Nah, 2004.) Providing feedback of the delays reduces anxiety and uncertainty as well as improves user attitudes (Galletta et al., 2006a). Hernández et al. (2009) also state that users are more likely to tolerate longer waiting times if they are provided feedback, for example by showing a progression percentage of the page load process.

Galletta et al. (2004) measured the performance, attitudes and behavioral intentions of 196 test subjects doing information retrieval tasks on webpages with artificial delays of 0, 2, 4, 6, 8, 10 or 12 seconds. Increased delays were related to decreased performance, attitudes and behavioral intentions. To promote a positive attitude, a website's loading delay should ideally be below 8 seconds. To encourage the user to continue using the site or to return later, the delay should be below 4 seconds. As the site becomes more familiar to the user, the role of delays become more important. The authors point out that more detailed conclusions cannot be made, for example between 2 and 4 seconds, because of the 2-second intervals used in the study. (Galletta et al., 2004.)

On the other hand, Galletta, Henry, McCoy and Polak (2006b) point out that website delay and other website design variables do not affect the user's behavior only individually, but also in combination. The authors conducted an experimental study to examine the effects of website delay, site breadth and content familiarity on user performance, attitudes and behavioral intentions. All three factors were found to strongly impact user performance and attitudes, which in turn affects behavioral intentions to return to the site. The effects of the three factors are, however, multiplicative instead of additive, so difficulties in some factors can be compensated by manipulating one or more of the other factors. If content familiarity can be expected to be high and the site breadth has only a few levels, the impact of delay is smaller. Correspondingly, depth of site hierarchy and lack of content familiarity can be compensated with small delays. The authors suggest that future studies could examine other factors as well, like user motivation and screen size, to gain a better understanding of the relationships between the different factors under different circumstances. (Galletta et al., 2006b.)

2.3 Web performance metrics

Webpage load times should not be presented as just a single number, because they can vary greatly between users, depending on the user's device and network conditions (Walton, n.d.). Other affecting factors are performance of the web browser, internet connection speed, local network traffic, load on the web server, and structure and format of the webpage (Nah, 2004). Delays in the chain from the server to the user's browser are cumulative, and the user's experienced response time is determined by the weakest link (Nielsen, 1997). The chain contains the following links: server performance, the server's internet connection speed, bottlenecks in the internet itself (for example related to cross-continent connections and rush hours), the user's internet connection speed, and the rendering speed of the user's browser and device. (Nielsen, 1997.) Mobile devices introduce other challenges: reduced bandwidth and increased latency of mobile network connections, reduced processing power of mobile devices (Gardner, 2011) and slow web browsers (Wang, Lin, Zhong, & Chishtie, 2011).

Another reason why a single performance metric is not enough is that there are multiple moments in the loading of a webpage that affect the perceived quickness of the loading (Walton, n.d.). For example, if a page renders quickly but takes several seconds to become interactive, the user probably does not care that the page was rendered quickly. There are four user experiences related to the performance of a webpage, summarized in Table 1. (Walton, n.d.)

Table 1. Four experiences related to the performance of a webpage (Walton, n.d.).

Experience	Description	Performance metrics
Is it happening?	Did the navigation start successfully? Has the server responded?	First Paint (FP) First Contentful Paint (FCP)
Is it useful?	Has enough content rendered that users can engage with it?	First Meaningful Paint (FMP) Hero Element Timing
Is it usable?	Can users interact with the page, or is it still busy loading?	Time to Interactive (TTI)
Is it delightful?	Are the interactions smooth and natural, free of lag and jank?	Long Tasks (technically the absence of long tasks)

When a user requests a webpage, the time between the request (e.g. clicking a link) and the first byte of the content coming in is measured by Time to First Byte (TTFB) metric (Halepovic, Pang, & Spatscheck, 2012; Miller & Osmani, 2019). The user's thought of whether the page has started loading – the first experience in Table 1 – is measured by First Paint (FP) and First Contentful Paint (FCP) metrics (Walton, n.d.). FP denotes the moment when the browser renders any pixel on the screen, whereas FCP denotes the moment when the browser renders the first bits of the requested content (e.g. article body) on the screen (Miller & Osmani, 2019; Walton, n.d.).

The second experience in Table 1 tells whether anything useful has been rendered on the screen (Walton, n.d.). If the useful parts of a webpage load quickly, the user might not even notice if the rest of the page loads slowly. (Walton, n.d.) Nielsen (1997) states similarly that longer loading times matter less if the user can quickly see and act on some useful information. For example, the top of the page should be rendered quickly, and it should contain meaningful text even before images have been downloaded. (Nielsen, 1997.) This experience is measured by First Meaningful Paint (FMP) and Hero Element Timing metrics (Walton, n.d.). There is no standardized definition for FMP, partly because it is difficult to generally *specify* what is useful on a webpage. Though it is easy to *identify* what parts are the most useful for the users. For example, on a video streaming site, it is the main video, and on a news site, it could be the cover image and the beginning of the story. The most useful parts are often referred to as "hero elements," and FMP can be considered as the moment when they are visible on the screen. (Walton, n.d.)

The third experience in Table 1 tells whether the page is usable and interactable and is measured by Time to Interactive (TTI) metric (Walton, n.d.). "Interactive" in this case is defined as the point where the page has displayed content (measured with FCP), "event handlers are registered for most visible page elements," and "the page responds to user interactions within 50 milliseconds" (*Time to Interactive*, n.d.). If the content is displayed and the page appears to be ready, but the page is not interactable, it can cause a frustrating user experience. (*Time to Interactive*, n.d.)

The fourth experience in Table 1 relates to the performance after the page has loaded (Walton, n.d.). For example, performance problems can be caused by janky scrolling and animations or slow responses to clicks and taps. (Walton, n.d.) Because this study focuses on the loading and rendering speed, this fourth experience is not relevant in this study and thus not covered more extensively.

Four of the web performance metrics – FP, FCP, FMP and TTI – are visually represented in Figure 1. The figure contains screenshots of different phases in the loading of a webpage and associates the four metrics to specific screenshots. The four metrics in Figure 1 fit in the three first experiences in Table 1.

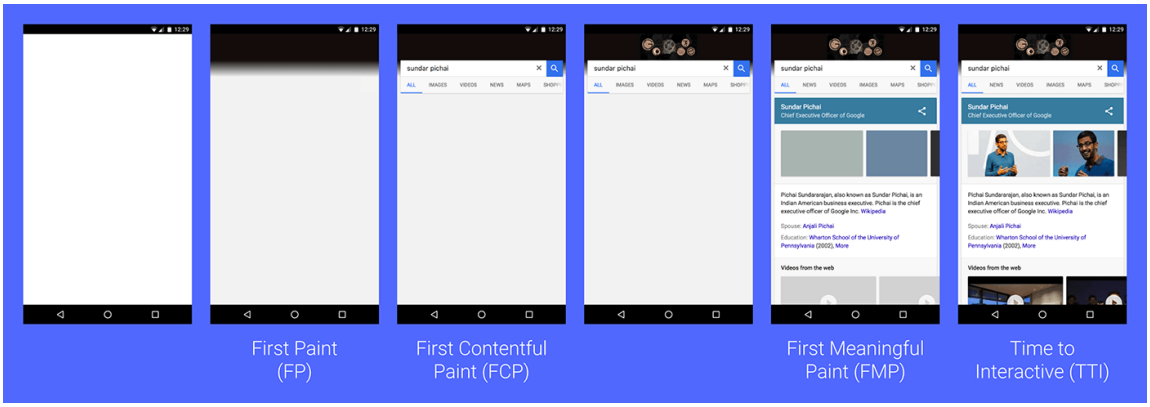


Figure 1. A visual representation of the loading of a webpage (Walton, n.d.). Licensed under the Creative Commons Attribution 4.0 License.

There are other metrics as well than the ones covered earlier in Chapter 2.3. For example, First Input Delay metric was introduced in 2018 and correlates well with TTI (Walton, 2018). Load time and DOMContentLoaded time are traditional but "extremely unreliable" performance metrics because "when they occur may or may not correspond to when the user thinks the app is loaded" (Walton, n.d.).

2.4 Evaluating web performance

The paper by Mateos, Mera, González and López (2001) "provides an original Web Assessment Index" (p. 226) which focuses on accessibility, speed, navigability and content. Speed is indirectly measured by the size (in bytes) of the website's home page. The page size and loading speed are related so that "a shorter [*sic*] home page will imply a faster access" (p. 228). The authors then applied the index to assess the websites of all Spanish universities. (Mateos et al., 2001.)

The paper by González and Palacios (2004) similarly "aims to provide a detailed and original Web Assessment Index, which focuses on [the same] four categories" (p. 313). Speed is indirectly measured by "the size (in bytes) of the home page, including the size of all the image files included" (p. 316). The page size and loading speed are again stated to be related, "so a shorter [*sic*] home page will imply a faster access" (p. 316). The authors state in a footnote that this is not an optimal way to measure speed, but acceptable due to its objective nature. The authors assessed the websites of the largest firms in Spain by applying the index. The authors then applied the index "to assess the web sites of the largest firms in Spain" (p. 313). (González & Palacios, 2004.)

Miranda, Cortés and Barriuso (2006) as well as Miranda, Sanguino and Bañegil (2009) assessed the quality of some websites by using an "original Web Assessment Index"

which focuses on the same four categories mentioned in the papers by Mateos et al. (2001) and González and Palacios (2004). Miranda, Cortés and Barriuso (2006) assessed the quality of the websites of Spanish private and savings banks, and Miranda, Sanguino and Bañegil (2009) the quality of 84 European municipal websites. However, both groups of authors measured speed (in seconds) with a "chronometer." Sources of recording errors were minimized by performing the tests at the same time and on the same computer on consecutive days. (Miranda, Cortés, & Barriuso, 2006; Miranda, Sanguino, & Bañegil, 2009.)

Google's PageSpeed and Yahoo!'s YSlow are tools that suggest best practices for optimizing web performance (Nicolaou, 2013). The practices apply well when developing websites for desktops. When targeting mobile devices, some of the practices apply also well, but some of them should be used only occasionally and some should be ignored. That is because mobile networks are problematic, and phones are slower than desktops at parsing and executing JavaScript code. (Nicolaou, 2013.)

Ismailova and Inal (2016) compared the accessibility and quality of governmental websites in Kyrgyzstan, Azerbaijan, Kazakhstan and Turkey. Performance was tested by using a speed test tool called Pingdom "since it is available online (at http://tools.pingdom.com/fpt/) and commonly used by researchers in this domain" (p. 990). The authors also mention PageSpeed tool by Google for identifying page loading issues. (Ismailova & Inal, 2016.)

Gash (n.d.) describes four different tools for measuring web performance: Lighthouse, PageSpeed Insights, WebPageTest and Pingdom. There are many other tools available as well, but these tools are "among the most popular and useful." It is good to use many tools and "compare the analyses for both common factors and unique outliers." (Gash, n.d.) Other performance tools include TestMySite and Chrome DevTools (*How To Think About Speed Tools*, 2019).

3. Research Method

This chapter is divided into two sub-chapters. Chapter 3.1 describes the experimental research method used in the study, and Chapter 3.2 the website used in the experimental part of the tool comparison.

3.1 Experimental research method

Computer science is often seen as empirical or experimental science (Gustedt et al., 2009; Tedre & Moisseinen, 2014), though what is meant by empirical and experimental in this context is often unclear (Tedre & Moisseinen, 2014). The research method used in this study is experimental research. Some of the studies covered in Chapter 2, like the studies by Galletta et al. (2006b), Hoxmeier and DiCesare (2000) and Nah (2004), have also utilized an experimental research methodology.

Experimental research methodology has three characteristics: the primary objective is explanatory (versus exploratory, descriptive or improving), the primary data is quantitative (versus qualitative), and the design process is fixed (versus flexible) (Runeson & Höst, 2008). The primary objective being explanatory means "seeking an explanation of a situation or a problem, mostly but not necessary in the form of a causal relationship" (p. 135). Quantitative data involves numbers and is analyzed using statistics. A fixed design process means that "all parameters are defined at the launch of the study" (p. 136) and not changed during the study. (Runeson & Höst, 2008.)

Any of the four primary research objectives mentioned by Runeson and Höst (2008) – exploratory, descriptive, explanatory and improving – could be thought to apply to this study because the authors describe the different objectives very tersely. Quantitative primary data and a fixed design process, however, apply well to this study. Many web performance evaluation tools output numerical data, as can be seen later in Chapter 5, making the primary data in this study quantitative and thus analyzable and comparable in a statistical manner. A fixed design process was mandated by the author's limited resources available for this study as to prevent scope creep. It is unclear what Runeson and Höst (2008) mean by the parameters of a fixed design process, but in this study, no major changes were made to the research question and sub-questions (Chapter 1), selection criteria of the tools (Chapter 4.2 and Chapter 5.1) or what pieces of information were compared (Chapter 4.3) after they were defined.

A good experiment is reproducible, extensible, applicable and revisable (Gustedt et al., 2009). The authors focus on large-scale and distributed systems, but state that their paper is largely applicable to any fields of computer science. Reproducibility enables other researchers to reproduce the experimental conditions and get the same results with the same input. This requires that the conditions are designed and described thusly. An extensible experiment "must target comparison with past and future results, extension with more or different processors, larger data set, or different architectures" (pp. 5–6). Applicability requires "realistic and representative set of entry parameters, input data, and usage" (p. 6). A revisable experiment helps to identify the reasons if the experimental hypotheses are not met. (Gustedt et al., 2009.) To make the experiment in this study reproducible and extensible, the process of using the different web performance evaluation tools were documented so that the process could be repeated for

other websites. As the case website in the study was a real website of a real company, the experiment can be considered as being applicable.

Gustedt et al. (2009) split experimental methodologies into four types based on whether the application and environment are real or models. In *in-situ* experiments, a real application is run in a real environment; in *emulation* experiments, a real application is run on a model of the environment; in *benchmarking* experiments, a model of the application is run in a real environment; and in *simulation* experiments, a model of the application is run on a model of the environment. The authors consider the classification as "very general and applicable to other domains that [*sic*] large-scale systems" (p. 7). (Gustedt et al., 2009.) In this study, both the application and environment were real, as is described later in Chapter 3.2. Thus, the experiment falls under the group of in-situ experiments.

Among the four experimental methodologies, in-situ experiments offer the least amount of abstraction (Gustedt et al., 2009). The methodology can be necessary if some complex behaviors and interactions cannot be easily captured and then simulated or emulated. However, reproducibility is lowered because controlling the network traffic or CPU (Central Processing Unit) usage of real machines is difficult. If experimental control or reproducibility are important, doing an emulation and simulation experiment should be considered. (Gustedt et al., 2009.) In this study, because the in-situ experiment methodology was used, the reproducibility of the results is likely low. However, the process of using the tools and applying their performance improvement suggestions is ideally reproducible.

Tedre and Moisseinen (2014) describe five different kinds of experiments conducted in computer science literature: feasibility experiments, trial experiments, field experiments, comparison experiments and controlled experiments. A comparison experiment is a comparison between solutions (Tedre & Moisseinen, 2014) and is the most fitting classification for this study as the performance of the website before and after implementing the performance improvements were compared between each other. The different web performance evaluation tools were also compared between each other. Comparison experiments seem objective but are susceptible to different kinds of biases, like failing to follow the blinding principle (Tedre & Moisseinen, 2014). In this study, bias was tried to be kept at minimum by comparing the tools based on their objective factors.

In a field experiment, some aspects of a system, for example performance, are measured in a live environment (Tedre & Moisseinen, 2014). Since the experiment in this study was an in-situ experiment in which the performance of the case website was evaluated, the field experiment is another suitable classification for this study. A common downside of field experiments is reduced reproducibility (Tedre & Moisseinen, 2014). Reproducibility of this study has been addressed earlier in this chapter.

3.2 The website used in the experiment

An existing WordPress site was used for the experimental part of the comparison of web performance evaluation tools. It is the website of a small Finnish company doing designing and contract work of electrical and HVAC (heating, ventilation and air conditioning) systems. The website consists of about a dozen pages. All of them are relatively simple, containing only text, graphics and a contact form. The website is presumably a rather typical website for a small company, given that the pages are simple and mostly informative, and the website uses WordPress – which "powers 36% of the internet" (WordPress.com, n.d.).

To reduce the scope of the experiment, only the front page of the website was selected for the experiment. The front page is possibly the page that is visited first by most users, so making the front page faster would provide a better first impression of the website. In addition, faster loading times encourage exploration as mentioned in Chapter 2, so making the front page faster can possibly increase the total amount of page visits. Figure 2 contains a screenshot of the top of the front page. A screenshot of the full front page can be seen in Appendix A. The screenshots were taken before the performance improvements were made, on March 25, 2020.

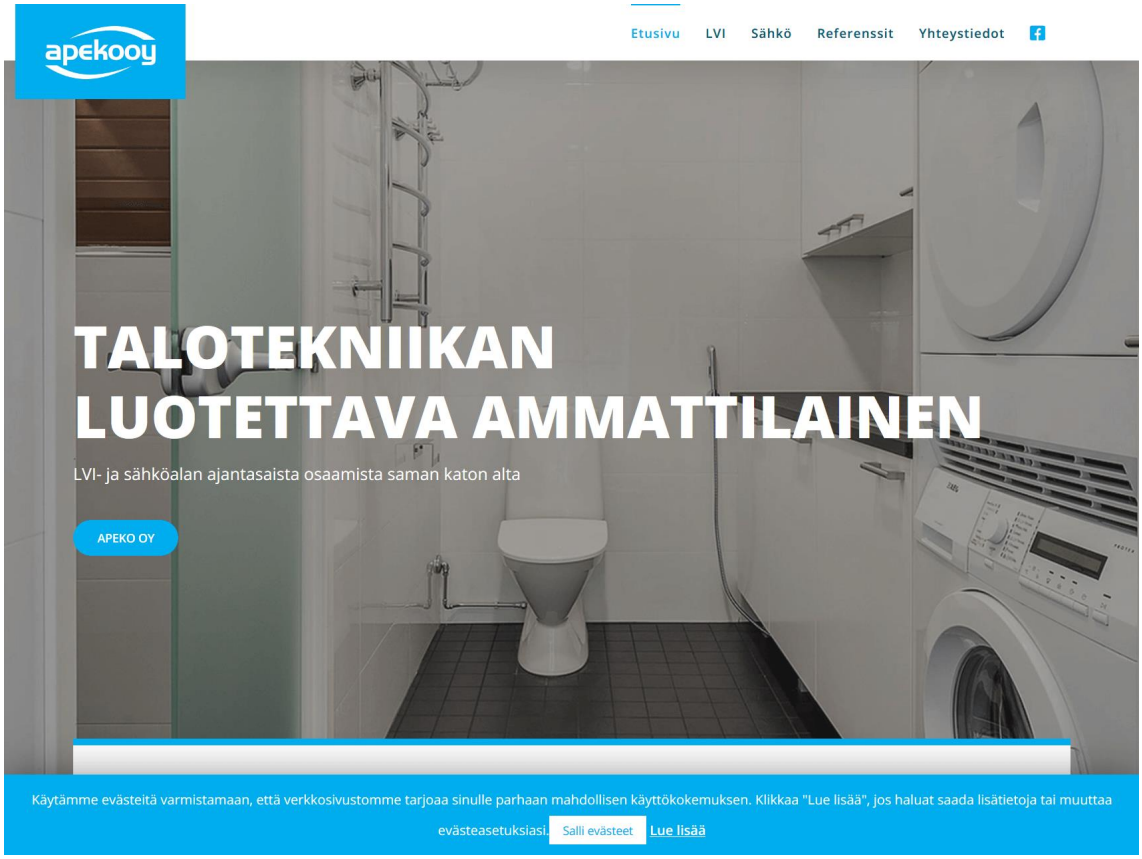


Figure 2. The top of the front page of the website that was used in the experiment.

The experiment consisted of the following steps:

1. The performance of the front page was evaluated using select tools (Chapter 5.1 and Chapter 5.2).
2. The suggestions provided by the tools were implemented to improve the performance of the website (Chapter 5.3). The functionality and content of the website were kept as intact as possible to avoid changes in functionality or content from affecting the site's performance.
3. The performance of the front page was re-evaluated using the same tools to see if and how the interventions changed the performance (Chapter 5.4).

As mentioned in Chapter 3.1, the experiment done in the study was an in-situ experiment: a real application was run in a real environment. Despite this, the performance improvements were first implemented and tested in a local development environment as to avoid bugs and other errors in the real environment, that is, in the production environment. The performance re-evaluation was done only after the improvements had been deployed to production.

4. Finding and Comparing Tools

This chapter is divided into three sub-chapters. Chapter 4.1 describes the way the tools were searched for; Chapter 4.2 describes the criteria used for selecting the tools for the comparison; and Chapter 4.3 lays out the comparison of the tools. The information retrievals mentioned in this chapter were done in February 2020 throughout several days.

4.1 Searching for the tools

Eight different tools were already mentioned in Chapter 2.4. In alphabetical order, they are: Chrome DevTools, Lighthouse, PageSpeed, PageSpeed Insights, Pingdom, TestMySite, WebPageTest and YSlow. To find more tools, two search engines were used: Startpage.com and DuckDuckGo.

Startpage.com uses Google's search results but offers a "wider variety of results" by not using the user's search habits to filter and personalize the search results, unlike other search engines (Startpage.com, n.d.). Thus, by using Startpage.com instead of Google, the search results are not filtered and personalized based on the author's prior searches, making the search results more neutral and more reproducible.

DuckDuckGo is a private search engine that is used to make over 1.5 billion web searches every month (DuckDuckGo, n.d.). DuckDuckGo does not use Google's search servers or algorithms except for showing results from YouTube (Weinberg, 2018), so using DuckDuckGo alongside Google or Startpage.com might provide more variety to the search results.

The search term "web performance tool" was used in both search engines. Similarly, only the first thirty search results of both search engines were reviewed to prevent the total amount of tools from growing too large. Only search results linking directly to a tool, or to a page describing a tool, were taken into consideration. Other types of search results, for example blog posts that list several tools, were skipped. Other irrelevant search results were also skipped. The searches were carried out on February 4, 2020.

Table 2 lists relevant search results from Startpage.com in the order they appeared in the search results: fifteen tools that are not mentioned in Chapter 2.4. Three tools mentioned in Chapter 2.4 appeared in the search results but are not listed in the table. Those tools are WebPageTest, PageSpeed Insights and Pingdom. Search results that were skipped as described in the previous paragraph are also not included in the table.

Table 2. Relevant search results from Startpage.com.

Search result title	Search result URL (Uniform Resource Locator)
GTmetrix | Website Speed and Performance Optimization	https://gtmetrix.com/
Web Performance: Load Testing Consulting Services & Tools	https://www.webperformance.com/
Website Speed Test | Check Web Performance » Dotcom-Tools	https://www.dotcom-tools.com/website-speed-test.aspx
Dareboost: Website Speed Test and Website Analysis – Free test	https://www.dareboost.com/en
Performance Test - Check URL Speed From 14 ... - KeyCDN Tools	https://tools.keycdn.com/performance
SpeedCurve: Monitor front-end performance	https://speedcurve.com/
Load Impact | Performance testing for DevOps teams	https://loadimpact.com/
Website Speed Test | Uptrends	https://www.uptrends.com/tools/website-speed-test
Calibre - Web Performance Monitoring	https://calibreapp.com/
LoadNinja | Performance Testing and Load Testing Tool	https://loadninja.com/
Welcome to the wonderful world of Web Performance	https://www.sitespeed.io/
Webserver Stress Tool - Performance, stress & load test. - Paessler AG	https://www.paessler.com/tools/webstress
LoadTracer - Free Load, Performance and Stress testing tool for ...	http://www.loadtracer.com/
Monitis: Web Performance Monitoring Tools	https://www.monitis.com/
Web Performance Monitor - SolarWinds	https://www.solarwinds.com/web-performance-monitor

DuckDuckGo yielded only one tool that has not been mentioned earlier: a Chrome extension called Performance-Analyser, available at https://chrome.google.com/webstore/detail/performance-analyser/djgfmlohefpomchfabngccpbaflcahjf. Tools that were found using DuckDuckGo but are mentioned earlier in this chapter are (in the order they appeared in the search results): WebPageTest, Web Performance (webperformance.com), GTmetrix, Pingdom, PageSpeed Insights, SolarWinds and Website Speed Test (by Uptrends).

4.2 Selection criteria for the comparison

A total of twenty-four tools were found, as described in Chapter 4.1. Some of the tools were found to focus on measuring the performance of a website under heavy load. That kind of web performance is different than what is focused on in this study – as stated in Chapter 1, the term “web performance” is used in this study to describe the loading and rendering speed of a webpage. The following five tools were such load testing tools and thus discarded from the comparison: Load Impact, LoadNinja, LoadTracer, Web Performance (webperformance.com) and Webserver Stress Tool (by Paessler). This reduced the number of tools to nineteen.

When trying to find information about the PageSpeed tool on Google, little to no information was found. Instead, most search results seemed to be about the PageSpeed Insights tool. It seems likely that the two tools are just one with two different names. It

is also possible that the tool was previously called “PageSpeed” and later renamed to “PageSpeed Insights.” Regardless, because the search results pointed towards the PageSpeed Insights tool, the PageSpeed tool was discarded from the comparison. This reduced the number of tools to eighteen.

When exploring the Pingdom tool, it turned out that there are two different versions of it available. Both Ismailova and Inal (2016) and Gash (n.d.) talk about a tool called “Pingdom” but actually refer to a tool called “Pingdom Tools,” as indicated by the link in both sources: https://tools.pingdom.com instead of https://www.pingdom.com. This increased the number of tools to nineteen.

Sitespeed.io is a set of open source tools for measuring and monitoring web performance (Sitespeed.io, 2020). One of the tools is called Browsertime. It is the most suitable tool for measuring the performance of a website when the primary interest is timing metrics. Another tool is called Coach, which gives advice how to make websites load faster and can be thought as a “modern version of YSlow.” The other tools focus on other things, such as monitoring web performance and building performance tools. (Sitespeed.io, 2020.) Initially, Browsertime and Coach were going to be included in the comparison as a pair, so that Browsertime would have been used for evaluating the performance and Coach for getting suggestions how to improve the performance. However, upon closer inspection, the two tools turned out to be too briefly documented and to output an overwhelming amount of data points. Because the data was so extensive yet briefly documented, the tools were discarded from the comparison. This reduced the number of tools to eighteen.

How To Think About Speed Tools (2019) mentions a tool called TestMySite, available at https://testmysite.thinkwithgoogle.com. As of February 7, 2020, the address redirected to https://www.thinkwithgoogle.com/intl/en-gb/feature/testmysite/. The name of the tool seemed to have changed as well to “Test My Site.” When doing a web search for “TestMySite,” another tool with almost the same name, Testmysite.io, was discovered. This increased the number of tools to nineteen.

YSlow turned out to be obsolete software. Installing and using it as a bookmarklet did not work, at least on the case website. The Firefox and Chrome add-ons were not available anymore for download. As of February 7, 2020, the latest commit on GitHub was from March 15, 2014, which made it look like the project is not actively developed. Because of these observations, YSlow was discarded from the comparison. This decreased the number of tools finally back to eighteen.

The remaining eighteen tools and their URL addresses are listed in Table 3. The tools are alphabetized in the table.

Table 3. URL addresses of the eighteen tools.

Tool	URL address
Calibre	https://calibreapp.com/
Chrome DevTools	Not applicable, the tool is built into the Chrome browser
Dareboost	https://www.dareboost.com/en
GTmetrix	https://gtmetrix.com/
Lighthouse	Not applicable, the tool is built into the Chrome browser
Monitis	https://www.monitis.com/
PageSpeed Insights	https://developers.google.com/speed/pagespeed/insights/
Performance-Analyser	https://chrome.google.com/webstore/detail/performance-analyser/djgfmlohefpomchfabngccpbaflcahjf
Performance Test (by KeyCDN Tools)	https://tools.keycdn.com/performance
Pingdom	https://www.pingdom.com/
Pingdom Tools	https://tools.pingdom.com/
SolarWinds	https://www.solarwinds.com/web-performance-monitor
SpeedCurve	https://speedcurve.com/
Testmysite.io	https://testmysite.io/
Test My Site	https://www.thinkwithgoogle.com/intl/en-gb/feature/testmysite/
WebPageTest	https://webpagetest.org/
Website Speed Test (by Dotcom-Tools)	https://www.dotcom-tools.com/website-speed-test.aspx
Website Speed Test (by Uptrends)	https://www.uptrends.com/tools/website-speed-test

The following three criteria were used for selecting some of the eighteen tools for the comparison covered in Chapter 4.3:

1. The tool can be used for free without time restrictions.
2. The tool reports at least one quantitative web performance metric.
3. The tool gives suggestions how to improve the performance of the evaluated webpage.

The selection criteria for each tool are mapped in Table 4. The criteria were determined by reading about the tools on their respective pages, by trying them out, or by both means. The tools are alphabetized in the table.

Table 4. What selection criteria the tools pass. Tools passing all three criteria are highlighted in green.

Tool	Criterion 1 (free-to-use)	Criterion 2 (1+ metric)	Criterion 3 (suggestions)
Calibre	No – the free trial plan is time-limited	Yes	?
Chrome DevTools	Yes – part of the Chrome browser	Yes	No
Dareboost	No – free accounts can do only five analyses per month	Yes	Yes
GTmetrix	Yes – no account required, but some features require registration. A free plan is available	Yes	Yes
Lighthouse	Yes – open source, part of the Chrome browser	Yes	Yes
Monitis	No – the free trial plan is time-limited	?	?
PageSpeed Insights	Yes	Yes	Yes
Performance-Analyser	Yes	Yes	No
Performance Test (by KeyCDN Tools)	Yes	Yes	No
Pingdom	No – the free trial plan is time-limited	Yes	?
Pingdom Tools	Yes	Yes	Yes
SolarWinds	No – the free trial plan is time-limited	?	?
SpeedCurve	No – the free trial plan is time-limited	Yes	?
Testmysite.io	Yes	Yes	No
Test My Site	Yes – downloading a detailed report requires giving out your email address, company name and country	Yes	Yes
WebPageTest	Yes	Yes	Yes
Website Speed Test (by Dotcom-Tools)	Yes	Yes	Yes
Website Speed Test (by Uptrends)	Yes	Yes	Yes

As indicated by Table 4, eight tools matched all three selection criteria: GTmetrix, Lighthouse, PageSpeed Insights, Pingdom Tools, Test My Site, WebPageTest, Website Speed Test (by Dotcom-Tools) and Website Speed Test (by Uptrends). Ten tools failed one or more criteria. Unclear criteria are marked with question marks, but all tools that have one or two unclear criteria failed to pass at least one other criterion, so those tools would not have been selected for the comparison anyway. The comparison of the eight selected tools is covered next, in Chapter 4.3.

4.3 Comparison of the tools

Information about the eight selected tools (Chapter 4.2) were collected by reading about the tools on their respective websites as well as by using each tool to evaluate the performance of the front page of the website described in Chapter 3.2. The performance evaluation results were not written down as the preliminary performance evaluations were done properly later (Chapter 5.1). Similarly, the results were not compared between each other as they were not written down. Instead, only the types and qualities of the results were collected. The following main pieces of information were looked for:

- Metrics: What performance metrics does the tool evaluate?
- Suggestions: What performance improvement suggestions does the tool give? (All possible suggestions, not just the ones that the tool gives for the case website.)

In addition, the following minor pieces of information were also looked for. They are minor or "less important" pieces of information in the sense that they do not necessarily affect what performance metrics the tools evaluate and what performance improvement suggestions the tools give, but they can still be relevant for practitioners using the tools.

- Open source: Does the tool have an open source license?
- Type: Is the tool a web-based software or something else (for example a downloadable software)?
- Physical locations: If the tool is web-based, what physical locations do the test servers have?
- Devices: What devices can be chosen for the evaluation (for example mobile and desktop)?
- Offline: Can the tool be used offline? This can be essential when the website is being developed but has not been published yet. This can also be important for intranets, websites requiring authentication and websites that contain sensitive content.
- Other evaluations: Does the tool evaluate other aspects of the website besides performance?

Features that fall outside of the items in the two lists above were not observed or written down. Any configuration options that the tools offered were not tested as some tools had a large amount of different configuration options and combinations. Comparison of the performance metrics is covered in Chapter 4.3.1, comparison of the performance improvement suggestions in Chapter 4.3.2 and comparison of all the minor pieces of information in Chapter 4.3.3.

4.3.1 Performance metrics

Some of the performance metrics introduced in Chapter 2.3 – TTFB, FP, FCP, FMP and TTI – are mapped in Table 5. Lighthouse and PageSpeed Insights were combined for the table, because the results page of PageSpeed Insights mentioned that "the speed score is based on the lab data analyzed by Lighthouse."

Table 5. Which tools evaluated some of the metrics introduced in Chapter 2.3.

Tool	TTFB	FP	FCP	FMP	TTI
GTmetrix	✓	✓	✓		
Lighthouse (and PageSpeed Insights)			✓	✓	✓
Pingdom Tools					
Test My Site			✓		
WebPageTest	✓		✓		
Website Speed Test (by Dotcom-Tools)					
Website Speed Test (by Uptrends)					

As can be seen in Table 5, none of the tools evaluated all five metrics. To evaluate each one of them, that is, to get a full coverage of the five metrics, one would need to use two tools: GTmetrix to evaluate TTFB, FP and FCP, and Lighthouse or PageSpeed Insights to evaluate FCP, FMP and TTI. Three tools did not evaluate any of the five metrics: Pingdom Tools, Website Speed Test (by Dotcom-Tools) and Website Speed Test (by Uptrends). Test My Site evaluated only FCP, and WebPageTest evaluated TTFB and FCP.

Other common metrics, present in at least three tools, are mapped in Table 6. Such metrics are Load time, Number of requests and Page size (in bytes). Lighthouse and PageSpeed Insights were again combined for the table.

Table 6. Which tools evaluated other common metrics (present in at least three tools).

Tool	Load time	Number of requests	Page size (in bytes)
GTmetrix	✓	✓	✓
Lighthouse (and PageSpeed Insights)			
Pingdom Tools	✓	✓	✓
Test My Site			
WebPageTest	✓	✓	✓
Website Speed Test (by Dotcom-Tools)	✓	✓	✓
Website Speed Test (by Uptrends)	✓	✓	✓

All three metrics in Table 6 were evaluated by most tools: GTmetrix, Pingdom Tools, WebPageTest, Website Speed Test (by Dotcom-Tools) and Website Speed Test (by Uptrends). Lighthouse, PageSpeed Insights and Test My Site did not evaluate any of these metrics.

GTmetrix evaluated a total of fifteen different metrics, which is the highest amount of metrics among the tools in the comparison (WebPageTest evaluated the same amount as well). Without an account, GTmetrix showed the results of six different metrics: PageSpeed Score, YSlow Score, Fully Loaded time, Page size (in bytes), Number of requests and Load time (this metric could be seen in a section called Waterfall Chart without an account). A section called Page Load Timings required an account and showed the results of nine other metrics: RUM[1] Speed Index, Redirect duration, Connection duration, Backend duration, TTFB, FP, FCP, DOM [(Document Object Model)] interactive time and DOM content loaded time. Load time was shown in this section as well.

WebPageTest evaluated a total of fifteen different metrics as well. Eleven of them were shown on the summary page: Load time, TTFB, Start Render, FCP, Speed Index, Last Painted Hero, First CPU Idle, Document Complete Time, Number of requests, Page size (in bytes) and Fully Loaded Time. Four more metrics were shown on the Details page: Visually Complete, DOM interactive time, DOM content loaded time and loadEvent time.

[1] GTmetrix did not provide a definition for the acronym “RUM.”

Lighthouse evaluated seven metrics: Performance score, FCP, FMP, Speed Index, First CPU Idle, TTI and Max Potential First Input Delay. PageSpeed Insights evaluated the same metrics, as mentioned earlier in this chapter.

Website Speed Test by Dotcom-Tools evaluated three metrics. All of them are mentioned in Table 6: Load time, Number of requests and Page size (in bytes). Pingdom Tools and Website Speed Test by Uptrends evaluated one additional metric (a total of four metrics by each tool): Performance grade and PageSpeed Score, respectively.

Test My Site evaluated only one metric: FCP, labeled as "mobile site speed." Though the tool reported two versions of the metric, one for a 3G and one for a 4G connection. Based on the FCP value, the tool also showed a rating of Fast (0–1 seconds), Average (1–2.5s) or Slow (2.5s and up). FCP is shown "as simulated in a lab environment and available from Google PageSpeed Insights" (*Frequently Asked Questions*, n.d.).

It should be noted that it is unclear whether some of the different metrics mentioned in this chapter are the same but with just different names. For example, the "Start Render" metric evaluated by WebPageTest sounds like it could be the same as FP, but due to lack of proper documentation, this assumption could not be verified.

4.3.2 Performance improvement suggestions

Suggestions given by the tools can be put into three categories: PageSpeed suggestions, YSlow suggestions and WebPageTest suggestions. GTmetrix listed its suggestions in two sections called PageSpeed and YSlow. Test My Site uses PageSpeed Insights as the source of the performance optimizations (*Frequently Asked Questions*, n.d.). Uptrends stated that its recommendations are by Google PageSpeed Insights, and Website Speed Test by Dotcom-Tools listed its suggestions in a section called "Page Speed [*sic*] Insights." Finally, the suggestions by PageSpeed Insights were the same as in Lighthouse. In addition to GTmetrix, Pingdom Tools also provided YSlow suggestions. WebPageTest had its own suggestions.

GTmetrix provided a total of 45 performance improvement suggestions, which is the highest amount of suggestions among the tools in the comparison. Out of the 45 suggestions, 26 were PageSpeed suggestions. Lighthouse, PageSpeed Insights and Website Speed Test by Uptrends provided 24 PageSpeed suggestions. Test My site provided 22 and Website Speed Test by Dotcom-Tools provided 10 PageSpeed suggestions. Test My Site showed only a couple of suggestions in the results page. Getting a list of all the possible suggestions required generating a report, which in turn required giving out one's email address, company name and country. All PageSpeed suggestions are listed in the table in Appendix B.

Some of the PageSpeed suggestions given by GTmetrix were apparently retrieved using an old, deprecated version of PageSpeed Insights API (Application Programming Interface). This was evident by clicking the "What's this mean?" button next to a PageSpeed suggestion on GTmetrix to read the suggestion's documentation, and then clicking the "Read More" link on the documentation page. For example, the "Read More" link in the documentation page of the "Enable compression" suggestion led to a page which stated the following: "Deprecated. This page was written for version 4 of the PageSpeed Insights API, which is deprecated and will be shut down in May 2019. Version 5 is the latest and provides both real-world data from the Chrome User Experience Report and lab data from Lighthouse." (*Enable Compression*, 2018). 11 suggestions out of the 26 led to pages with the deprecation message.

Some of the PageSpeed suggestions given by Website Test Tool by Dotcom-Tools were the same as given by GTmetrix, as can be seen in Appendix B. Because GTmetrix apparently retrieved at least some of those suggestions using a deprecated version of the PageSpeed Insights API, it seems likely that Website Test Tool by Dotcom-Tools also used an old version of the PageSpeed Insights API.

In addition to the PageSpeed suggestions, GTmetrix also provided 19 YSlow suggestions. Pingdom Tools provided 22 YSlow suggestions, though only the seven most important suggestions were shown in the tool. The rest of the suggestions were found by exploring the front-end JavaScript code of the tool by using Chrome DevTools. The suggestions were labeled as "yslow_rules." One more suggestion, "Use a Content Delivery Network (CDN)," was also found, which was however disabled for some reason. All YSlow suggestions are listed in the table in Appendix C.

WebPageTest provided seven suggestions. In alphabetical order, they were: "Compress Images," "First Byte Time (back-end processing)," "Leverage browser caching of static assets," "Use a CDN for all static assets," "Use gzip compression for transferring compressable [*sic*] responses," "Use persistent connections (keep alive)" and "Use Progressive JPEGs [(JPEG = an image file with the extension '.jpg' or '.jpeg')]."

4.3.3 Other features

Two of the eight tools were open source software: Lighthouse and WebPageTest. The tools' source codes were available at https://github.com/GoogleChrome/lighthouse and https://github.com/WPO-Foundation/webpagetest, respectively.

All tools except Lighthouse were web-based tools, that is, they were used on their respective websites using a web browser. Lighthouse was built into the Chrome browser and was also available as a Chrome browser extension and a Node module, as described in the tool's GitHub page. However, only the built-in version of Lighthouse was examined in Chapter 4.

When comparing the physical locations of the tools' test servers, US states were counted as distinct locations. GTmetrix supported seven locations, and changing the location required registering an account. Lighthouse was a locally run tool, so the question of physical locations was not applicable. Physical locations of PageSpeed Insights were not mentioned in the tool and could not be found in the tool's documentation either. Pingdom Tools supported seven locations. Physical locations of Test My Site were not mentioned. Depending on the tested webpage, the tool sometimes showed a dropdown menu with a couple of different locations. However, it was mentioned that "the page speed metric shown is for 4G and is not specific to any given country" (*Frequently Asked Questions*, n.d.). WebPageTest supported 33, Website Speed Test by Dotcom-Tools 24, and Website Speed Test by Uptrends nine locations. All locations are listed in the table in Appendix D. Some tools may have more than one location in a single country or US state. However, those were not written down as it would have made the comparison more complex and difficult. The locations supported by Test My Site were left out of Appendix D, because the locations were not documented, and the tool was inconsistent about whether it listed any locations at all on the results page.

GTmetrix supported two desktop browsers – Chrome and Firefox – and a single mobile device: Galaxy Nexus with the Chrome browser. Lighthouse had only a general choice between a desktop or mobile device. PageSpeed Insights supported a desktop device (not mentioned more specifically) and a mobile device: "a mid-tier device (Moto G4) on

a mobile network" (*About PageSpeed Insights*, 2019). Pingdom Tools did not offer devices to choose from. The results included a screenshot, which was wide, indicating that a desktop browser was used. Test My Site did not offer either and the performance metric it evaluated was labeled as "mobile site speed," as mentioned in Chapter 4.3.1. WebPageTest offered several versions of several desktop browsers: Brave, Chrome, Chromium, Edge, Firefox, Internet Explorer, Opera. There were 14 different mobile devices with various browser choices and some with portrait and landscape orientations. Website Speed Test by Dotcom-Tools offered Firefox, Chrome and multiple versions of Internet Explorer as the desktop browsers as well as 39 Android devices, 20 iOS devices, four Windows Phone devices and three Blackberry devices. Website Speed Test by Uptrends had four desktop browsers – Chrome, Firefox, Internet Explorer and Phantom JS – and various mobile devices: 12 Android devices, 11 iOS devices, one Windows Phone device and one MeeGo device (Nokia N9). Depending on the tool, some devices were available only when choosing certain physical locations.

Because all tools except Lighthouse were web-based tools, they were not usable offline. When trying to use Lighthouse for a local HTML (Hypertext Markup Language) file (so that the file was opened in the Chrome browser), the tool showed the following error message: "Can only audit HTTP/HTTPS [(Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure)] pages and Chrome extensions. Navigate to a different page to start an audit." When serving the same file using a Node package called "serve," so that the file was available at http://localhost:5000, Lighthouse was able to perform an audit. However, without an internet connection, Lighthouse got stuck in the "Lighthouse is warming up…" phase. It thus seems like Lighthouse is not usable offline but is usable for local websites that are not available on the internet.

Lighthouse was the only tool that could evaluate other aspects besides performance. The other evaluable aspects in Lighthouse were "Progressive Web App," "Best practices," "Accessibility" and "SEO" (Search Engine Optimization). In addition to the categories, the tool had a section called "Community Plugins(beta)" [*sic*] with one option: "Publisher Ads."

5. Performance Evaluations and Improvements

This chapter is divided into four sub-chapters. Reasons why some of the tools compared in Chapter 4.3 were not selected for the experiment are described in Chapter 5.1. Results of the preliminary performance evaluation are reported in Chapter 5.2. What performance improvements were made, and how they were made, is detailed in Chapter 5.3. Results of the performance re-evaluation are finally reported in Chapter 5.4.

5.1 Selecting tools for the experiment

Only some of the eight tools compared in Chapter 4.3 were selected for the experiment based on whether each tool evaluates unique metrics compared to the other tools and/or whether each tool gives unique suggestions compared to the other tools. This is because the point of the experiment was not to compare the results of the different tools per se, but to answer the last three research sub-questions presented in Chapter 1; selecting tools whose metrics and/or suggestions overlap is not necessarily important in that regard. Another reason is that selecting only a few tools reduced the scope of the experiment, which was important due to the time constraints of the author.

The performance evaluations of PageSpeed Insights are based on Lighthouse, and the suggestions are also the same. When comparing the two tools, PageSpeed Insights likely produces more reproducible results between different researchers and practitioners because PageSpeed Insights is a web-based tool, whereas Lighthouse is run locally, so Lighthouse's results possibly depend on the performance of the user's computer. Because the two tools report the same metrics and suggestions, it is not relevant which one was chosen for the experiment. Ultimately PageSpeed Insights was chosen because the author found it more convenient to use.

Website Speed Test by Dotcom-Tools evaluates only three metrics: Load time, which is an "extremely unreliable" performance metric (Walton, n.d.), and Page size and Number of requests, which neither directly measure performance. The tool gives two unique suggestions (see Appendix B) – though the suggestion "Eliminate render-blocking JavaScript and CSS [(Cascading Style Sheets)] in above-the-fold content" sounds like it is essentially the same but just differently worded as the suggestion "Eliminate render-blocking resources" given by some other tools, reducing the amount of unique suggestions to one. Because of the lack of proper performance metrics and unique suggestions, the tool was discarded from the experiment.

Website Speed Test by Uptrends evaluates only four metrics: the same three as Website Speed Test by Dotcom-Tools as well as PageSpeed Score, which is also evaluated by GTmetrix. The tool by Uptrends also gives no unique suggestions, so it was discarded from the experiment. Test My Site evaluates only one metric (FCP) and gives only one unique suggestion (see Appendix B), so it was also discarded from the comparison.

Pingdom Tools evaluates only four metrics as well: the same three as Website Speed Test by Dotcom-Tools, and Performance grade. The YSlow suggestions given by Pingdom Tools are also given by GTmetrix, apart from three rules (see Appendix C). Thus, using Pingdom Tools along with GTmetrix would not have produced any

significant value to the experiment, so Pingdom Tools was discarded from the experiment.

Out of the eight tools compared in Chapter 4.3, three were thus selected for the experiment: GTmetrix, PageSpeed Insights and WebPageTest. The three tools have some overlaps, but each one of them also provides information that the others do not: each tool evaluates more than one unique metric (see Table 7 in Chapter 5.2.1) and gives suggestions not given by the other tools. GTmetrix is the only tool to give YSlow suggestions, PageSpeed Insights gives newer PageSpeed suggestions than GTmetrix, and WebPageTest is the only tool to give other kind of suggestions than PageSpeed or YSlow suggestions.

5.2 Preliminary performance evaluation

Test servers close to Finland were chosen because the case website is also hosted in Finland. The closest location offered by GTmetrix was London, United Kingdom. The same location would have been used in WebPageTest as well for better comparability between the results, but the test did not work for some reason when choosing that location. Stockholm, Sweden was chosen next, but the results were missing three metrics: Start Render, Speed Index and Last Painted Hero. Paris, France was finally chosen and produced all metrics. PageSpeed Insights did not support choosing a location for the test servers.

The Chrome browser (desktop) was selected in GTmetrix's options. Firefox would have been chosen in WebPageTest to add variety to the browsers used to evaluate the performance, but the tool did not give any suggestions when Firefox was chosen. Chrome (desktop) was chosen instead, and on the other hand, this made the comparability of the results between these two tools better. Mobile device options were not available for the chosen locations in GTmetrix and WebPageTest. PageSpeed Insights automatically produced both mobile and desktop results.

Any other configuration options that the tools offered were not tested or changed. The preliminary performance evaluations were conducted on February 28, 2020 between 7 AM and 8 AM (UTC+2). The tools were run one at a time (that is, sequentially instead of concurrently), and each tool was run ten times (sequentially). Running each tool and each run sequentially potentially avoided increasing the momentary load of the site, which could have made the site load momentarily slower. The average values of the results were calculated and are reported in Chapter 5.2.1. Suggestions given by the tools are reported in Chapter 5.2.2.

5.2.1 Performance metrics

The metrics reported by the tools are listed in Table 7. All values use the units and precisions reported by the tools.

Table 7. Performance metrics evaluated by the selected tools. The values are average values between the ten runs unless otherwise noted.

Metric	GTmetrix	PageSpeed Insights		WebPageTest
		Mobile	Desktop	
Redirect duration	0ms[2]			
Connection duration	119ms			
Backend duration	96ms			
TTFB	215ms			0.397s
FP	0.7s			
FCP	0.7s	4.1s	1.0s	1.682s[3]
FMP		4.7s	1.2s	
TTI		7.1s	1.6s	
DOM interactive time	1.0s			2.690s
DOM content loaded time	1.0s (117ms)			2.690s…3.022s (0.332s)
Load time	1.15s (87ms)			4.180s
Document complete time				4.180s
loadEvent				4.189s…4.341s (0.152s)
Fully Loaded Time	1.39s			4.621s
Max Potential First Input Delay		543ms	172ms	
First CPU Idle		6.6s	1.5s	> 4.346s
Start Render				1.690s
Speed Index		6.4s	1.7s	3.696s
RUM Speed Index	1,127			
Last Painted Hero				5.620s
Visually Complete				5.650s
Number of requests	38[2]			37[2]
Page size (in bytes)	1.64MB[2]			1,779KB[2]
PageSpeed Score	B (82%)[2]			
Performance score		47	89	
YSlow Score	A (90%)[2]			

The first three metrics shown in Table 7 – Redirect duration, Connection duration and Backend duration – were evaluated only by GTmetrix, and their sum is equal to the TTFB metric, as documented in the tool. The next five metrics in the table are the same as in Table 5: TTFB, FP, FCP, FMP and TTI. GTmetrix evaluated the earliest metrics: TTFB (and its three subcomponents), FP and FCP, whereas PageSpeed Insights evaluated metrics that happen later during the loading and rendering of a webpage: FCP, FMP and TTI. WebPageTest evaluated TTFB and FCP.

---

[2] Constant value between the ten runs.

[3] Missing in the 2nd, 3rd, 4th and 6th runs.

The next six metrics seem to be related to each other, based on their names and numerical values. Of the six metrics, at least Load time and DOM content loaded time are “extremely unreliable” performance metrics (Walton, n.d.). The other metrics are DOM interactive time, Document complete time, loadEvent and Fully Loaded Time.

The next two metrics in the table are Max Potential First Input Delay and First CPU Idle. Then, the next five metrics are based on a video analysis of the loading of the webpage (GT.net, 2020; *Speed Index*, 2019). The metrics are Start Render, Speed Index, RUM Speed Index, Last Painted Hero and Visually Complete.

Lastly, at the end of the table are metrics that do not directly measure the performance (Number of Requests and Page size in bytes) and score-based metrics (PageSpeed Score, Performance score and YSlow Score). The Performance score ranges from 0 to 100, the latter being the perfect score. Similarly, PageSpeed Score and YSlow Score range from 0% to 100%, the latter being the perfect score. In addition to the percentages, a rating from A to F is given, the *former* being the perfect rating. For example, “A (100%)” is the best possible score/rating, and “F (0%)” is the worst possible score/rating.

Overall, common metrics (reported by two or three tools) were quite varied between each other. The page loaded and rendered the fastest when evaluated by GTmetrix, the second fastest when evaluated by PageSpeed Insights (desktop), and the third fastest when evaluated by WebPageTest. The mobile speeds, evaluated by PageSpeed Insights, were noticeably the slowest. Slower speeds on mobile are expectable based on the challenges of mobile devices and mobile networks covered in Chapter 2.3 and Chapter 2.4. GTmetrix and WebPageTest reported different values for the Number of requests and Page size (in bytes) even though they sound like they should be equal. The differences are possibly due to a potentially different way of evaluating those metrics.

The values of the two metrics DOM content loaded time and Load time, as evaluated by GTmetrix, have additional values in brackets. The times in the brackets are the times spent executing JavaScript triggered by the DOM content loaded event or the Onload event, respectively, as documented in the tool.

The values of the two metrics DOM content loaded time and loadEvent, as evaluated by WebPageTest, are ranges of times with additional values in brackets. The tool provided no explanation for the ranges but linked to a draft specification of “Navigation Timing Level 2” (Grigorik, 2020) which describes the DOM events domContentLoadedEventStart, domContentLoadedEventEnd, loadEventStart and loadEventEnd. Thus, the ranges seem to refer to the four DOM events, and the values in the brackets seem to be the differences between the respective end and start values. WebPageTest provided no explanation for why the value of the metric First CPU Idle started with the greater-than sign (“>”).

5.2.2 Performance improvement suggestions

Performance improvement suggestions given by GTmetrix, PageSpeed Insights and WebPageTest are listed and described in the following three sub-chapters, respectively.

GTmetrix

Suggestions given by GTmetrix are listed in Table 8. The scores of the suggestions ranged from 0 to 100, the latter being the perfect score. In addition, a rating from A to F

was given, the *former* being the perfect rating. Suggestions with the perfect score ("A (100)") are not included in the table, meaning that 20 PageSpeed suggestions and 16 YSlow suggestions had the perfect score. One YSlow suggestion, "Make JavaScript and CSS external," had the score of "(n/a)," meaning "not applicable." All scores were constant between the ten evaluation runs.

Table 8. Suggestions given by GTmetrix. The scores were constant between the ten runs.

	Score
PageSpeed suggestions	
Defer parsing of JavaScript	F (29)
Optimize images	F (49)
Serve scaled images	E (57)
Minify JavaScript	A (97)
Minify CSS	A (99)
Minify HTML	A (99)
YSlow suggestions	
Use a Content Delivery Network (CDN)	F (0)
Make fewer HTTP requests	B (84)
Make JavaScript and CSS external	(n/a)

The YSlow suggestion "Use a Content Delivery Network (CDN)" had the worst possible score of "F (0)." A CDN is a "collection of web servers distributed across multiple locations" which delivers static content like images, stylesheets and scripts to the user from the most suitable server, for example based on the lowest amount of network hops (Yahoo Developer Network, n.d.). The score was the lowest possible because a CDN was not used at all on the site.

The suggestion with the second worst score of "F (29)" was the PageSpeed suggestion "Defer parsing of JavaScript." The description of the suggestion states that "by minimizing the amount of JavaScript needed to render the page, and deferring parsing of unneeded JavaScript until it needs to be executed, you can reduce the initial load time of your page." The page indeed loaded several JavaScript files, many of them at the beginning of the page's source.

The next two lowest-ranking suggestions were PageSpeed suggestions "Optimize images" and "Serve scaled images" with the scores of "F (49)" and "E (57)," respectively. Optimizing images is described as using the correct image format depending on the situation, removing unnecessary data from the image and saving the images in the dimensions that they are displayed in. Serving scaled images is described as "serving appropriately-sized images," for example smaller images for mobile users.

Techniques for improving the YSlow suggestion "Make fewer HTTP requests," scored "B (84)," include combining multiple JavaScript scripts into a single script, combining multiple CSS styles into a single stylesheet file, and embedding images' data to the HTML document (Yahoo Developer Network, n.d.). Three PageSpeed suggestions "Minify JavaScript," "Minify CSS" and "Minify HTML" had almost perfect scores of "A (97)," "A (99)" and "A (99)," respectively, and can be further improved by removing unnecessary bytes from the respective files for example by using compression tools.

PageSpeed Insights

Suggestions given by PageSpeed Insights are listed in Table 9. Suggestions that were in the "Passed audits" category are not included, meaning that 12 audits passed in the mobile runs and 17 or 18 audits in the desktop runs (one audit did not pass only in the first desktop run). The values were not constant between the ten runs, so the table contains average values.

The suggestions were split in two categories: Opportunities and Diagnostics. The time values of the Opportunities in Table 9 are "Estimated savings," that is, how much the loading is estimated to improve when following the respective suggestion. The time values of the Diagnostics have various meanings, as described in Table 9.

Table 9. Suggestions given by PageSpeed Insights. The values are average values between the ten runs unless otherwise noted.

	Mobile	**Desktop**
Opportunities – "Estimated savings"		
Serve images in next-gen formats	4.02s	0.48s[4]
Efficiently encode images	2.4s[4]	0.32s[4]
Eliminate render-blocking resources	2.07s	0.45s
Preload key requests	1.58s	–
Remove unused CSS	0.90s	0.2s[4]
Defer offscreen images	0.3s[4]	–
Reduce server response times (TTFB)	0.28s	–
Diagnostics		
Ensure text remains visible during webfont load ("Potential Savings" in milliseconds, total of five fonts)	114ms	99ms
Minimize main-thread work (Time spent in seconds)	4.9s	2.1s[5]
Reduce JavaScript execution time (Time spent in seconds)	2.1s	–

The Opportunity with the highest estimated savings was "Serve images in next-gen formats." The description of the suggestion in the tool stated that "image formats like JPEG 2000, JPEG XR, and WebP often provide better compression than PNG [(an image file with the extension '.png')] or JPEG, which means faster downloads and less data consumption." The Opportunity with the second highest estimated savings, "Efficiently encode images," is basically the same as GTmetrix's PageSpeed suggestion "Optimize images." The Opportunity "Defer offscreen images" was described as loading offscreen and hidden images only after critical resources have finished loading.

The description of the Opportunity "Eliminate render-blocking resources" stated that the first paint of the page can be improved by delivering critical JS (JavaScript) and CSS

[4] Constant value between the ten runs.

[5] Only present in the first desktop run.

inline and deferring non-critical JS and CSS. Somewhat similarly, the Diagnostics "Minimize main-thread work" and "Reduce JavaScript execution time" were described as "reducing the time spent parsing, compiling and executing JS." The Diagnostic "Ensure text remains visible during webfont load" can be improved by "leveraging the font-display CSS feature," as described in the tool.

The description of the Opportunity "Preload key requests" stated that the loading of resources that are requested later (that is, not immediately when the page loads) can be improved by using the HTML tag <link rel=preload>. The description of the Opportunity "Reduce server response times (TTFB)" contained a WordPress-specific suggestion to find a more optimized WordPress theme, to select an optimization plugin and/or to upgrade the web server. The remaining Opportunity "Remove unused CSS" is self-explanatory; it suggests removing such CSS code that is not used on the evaluated page.

WebPageTest

Suggestions given by WebPageTest are listed in Table 10. The scores of the suggestions ranged from 0 to 100, the latter being the perfect score. Suggestions with the perfect score ("100/100") are not included in the table, meaning that four suggestions had the perfect score. All scores were constant between the ten evaluation runs.

Table 10. Suggestions given by WebPageTest. The scores were constant between the ten runs.

Suggestion	**Score**
Use Progressive JPEGs	0/100
Use a CDN for all static assets	14/100
Compress Images	55/100

Normal JPEG images are displayed in full quality but line-by-line, whereas progressive JPEG images are displayed entirely right away but in progressively increasing quality (Pock, 2019). The "Use Progressive JPEGs" suggestion had a score of zero, which indicates that all JPEG images were normal JPEGs.

The "Use a CDN for all static assets" is basically the same as GTmetrix's YSlow suggestion "Use a Content Delivery Network (CDN)." While GTmetrix gave a score of zero for its suggestion, WebPageTest gave a score of 14 because it detected that some font files were delivered from Google's CDN (fonts.gstatic.com).

The "Compress Images" suggestion is basically the same as GTmetrix's PageSpeed suggestion "Optimize images" and PageSpeed Insights's suggestion "Efficiently encode images."

5.3 Performance improvements

The steps taken to implement the performance improvement suggestions – using a CDN, optimizing images, concatenating CSS and JS files, and preloading key requests – are described in the first four sub-chapters, respectively. Some suggestions were not followed for various reasons; those suggestions and the reasons for skipping them are listed in Chapter 5.3.5.

5.3.1 Use a CDN

Since GTmetrix's YSlow suggestion "Use a Content Delivery Network (CDN)" and WebPageTest's suggestion "Use a CDN for all static assets" both had the worst possible score, the first performance improvement that was done was to configure a CDN. Cloudflare was chosen, mainly because the author was already familiar with it and because it could be used for free.

Cloudflare has data centers in "200 cities in more than 90 countries," including Helsinki, Finland (where the case website's web server is located), Paris, France (where the chosen WebPageTest server is located) and London, United Kingdom (where the chosen GTmetrix server is located) (Cloudflare, 2020a). Setting up Cloudflare was relatively simple and easy as the main step was to configure the website's domain to point to Cloudflare's name servers.

5.3.2 Optimize images

The front page contained a total of 53 JPEG and PNG images. Not all of them were visible on the page at once, because some images had multiple versions with different resolutions, so that the browser displayed the most suitable sized image. Some images were device-specific, for example touch icons for iOS devices.

The images were manually optimized by uploading them to a service called TinyPNG, available at https://tinypng.com/. Despite its name, the tool supports both PNG and JPEG files. The optimized images were then uploaded to the website, so that the old, uncompressed images were overwritten. TinyPNG reported saving a total of 1,193KB. Savings per file ranged from 0% to 95%. The following suggestions were targeted by optimizing the file sizes of the images: "Optimize images" PageSpeed suggestion given by GTmetrix, "Efficiently encode images" suggestion given by PageSpeed Insights, and "Compress Images" suggestion given by WebPageTest.

The optimized JPEG files were verified to be progressive by using a tool available at http://techslides.com/demos/progressive-test.html. Thus, optimizing the images using TinyPNG affected also the "Use Progressive JPEGs" suggestion given by WebPageTest. However, three optimized JPEG files were not progressive. This is likely because TinyPNG could not reduce the file size of the three images (savings were 0% as reported by the tool). Because of this, it is possible that WebPageTest will not give a perfect score for the "Use Progressive JPEGs" suggestion when doing the performance re-evaluation.

TinyPNG was chosen for optimizing the images because the author was already familiar with it and because it was free to use. Other tools might have produced different results. To optimize a greater number of images, an automated solution would make the task easier and faster.

5.3.3 Concatenate CSS and JS files

One way to improve GTmetrix's YSlow suggestion "Make fewer HTTP requests" is to combine multiple JavaScript scripts into a single script and multiple CSS styles to a single stylesheet file (Yahoo Developer Network, n.d.). The fewer files there are in total, the fewer HTTP requests the browser needs to do. Combining the JavaScript and CSS files manually turned out to be difficult due to how the WordPress theme was built, so a WordPress plug-in called Autoptimize was installed. The plug-in automatically

concatenates JS and CSS files. It also minifies the concatenated JS and CSS files, so GTmetrix's PageSpeed suggestions "Minify JavaScript" and "Minify CSS" were targeted by this improvement as well.

The compression rate of concatenated files tends to be better because "the compression algorithm can exploit similarities between the concatenated pieces of data" (The Spooniest, 2015). Because of this, PageSpeed Insights's suggestions "Minimize main-thread work" and "Reduce JavaScript execution time" can possibly score better in the re-evaluation as the description of both suggestions state (emphasis added): "Consider reducing the time spent parsing, compiling and executing JS. You may find *delivering smaller JS payloads* helps with this." Similarly, GTmetrix's PageSpeed suggestion "Defer parsing of JavaScript" is partially targeted, as the description of the suggestion also advises "minimizing the amount of JavaScript needed to render the page."

5.3.4 Preload key requests

In the description of the suggestion "Preload key requests," PageSpeed Insights mentioned two font files that would be beneficial to be preloaded by using the HTML tag <link rel=preload>. The WordPress plugin Autoptimize that was installed earlier (Chapter 5.3.3) had a feature to preload the files by specifying the file paths to the plugin's configuration screen. Thus, this suggestion was very straightforward to implement.

5.3.5 Suggestions not followed

Some suggestions were not followed because it would have been too difficult, error-prone and/or time consuming (given the time constraints of the author) to start reverse-engineering and modifying the WordPress theme to implement the suggestions. One of those suggestions is GTmetrix's PageSpeed suggestion "Defer parsing of JavaScript." However, it is expected that this suggestion scores better in the re-evaluation, because the suggestion stated that "912.5KiB of JavaScript is parsed during initial page load." So, because the JS files were concatenated as described in Chapter 5.3.3, the amount of JS that is parsed should be smaller in the re-evaluation.

Three suggestions given by PageSpeed Insights were not followed for the mentioned reasons. The first one is "Eliminate render-blocking resources," because it would require reverse-engineering the existing CSS and JS files. The second one is "Serve images in next-gen formats," because it would require modifying the WordPress theme to support other image formats. Furthermore, the suggestion likely scores better in the re-evaluation because the file sizes were anyway reduced (Chapter 5.3.2). The third suggestion is "Remove unused CSS," because it would require analyzing all pages of the website instead of just the front page; otherwise removing CSS that is unused on the front page would possibly break the layout of other pages.

Three suggestions had an almost perfect score or so little estimated savings that they were deemed unworthy of following. The suggestions are GTmetrix's PageSpeed suggestion "Minify HTML" as well as PageSpeed Insights's suggestions "Ensure text remains visible during webfont load" and "Defer offscreen images." The latter suggestion is expected to score better anyway in the re-evaluation as the images were optimized (Chapter 5.3.2). Similarly, GTmetrix's PageSpeed suggestion "Server scaled images" was not followed, because some images were already offered in various resolutions (as mentioned in Chapter 5.3.2), and the suggestion likely scores better in the re-evaluation since the file sizes were reduced (Chapter 5.3.2).

Finally, GTmetrix's YSlow suggestion "Make JavaScript and CSS external" was not followed, because its score was "(n/a)," which means "not applicable"; and PageSpeed Insights's suggestion "Reduce server response times (TTFB)" was not followed, because it was not possible to change the web server in this case, so there were no clear ways to reduce the TTFB metric.

5.4 Performance re-evaluation

The performance re-evaluations were conducted on March 29, 2020 between 8 AM and 9 AM (UTC+3), one tool at a time (that is, the tools were not run concurrently). Each tool was run ten times (sequentially), and the average values of the results were calculated. Performance metrics (average values) are reported in Chapter 5.4.1, and suggestions given by the tools in Chapter 5.4.2. WebPageTest did no longer have an option to choose Paris, France as the location, so "Paris - EC2" was chosen instead.

5.4.1 Performance metrics

The metrics reported by the tools are listed in Table 11. The metrics are the same and in the same order as in Table 7. Values that improved or worsened compared to the previous values (Table 7) are highlighted in green and red, respectively. Absolute changes of the values are shown in square brackets. Percentage points, used by PageSpeed Score and YSlow Score, are abbreviated as "pp."

Table 11. Performance metrics of the re-evaluation. The values are average values between the ten runs unless otherwise noted.

Metric	GTmetrix	PageSpeed Insights		WebPageTest
		Mobile	Desktop	
Redirect duration	0ms[6] [±0ms]			
Connection duration	109ms [-10ms]			
Backend duration	271ms [+175ms]			
TTFB	379ms [+164ms]			0.412s [+0.015s]
FP	0.9s [+0.2s]			
FCP	0.9s [+0.2s]	3.4s [-0.7s]	0.8s [-0.2s]	1.124s [-0.558s]
FMP		3.6s [-1.1s]	1.0s [-0.2s]	
TTI		7.7s [+0.6s]	1.8s [+0.2s]	
DOM interactive time	0.9s [-0.1s]			1.105s [-1.585s]
DOM content loaded time	1.1s (138ms) [+0.1s (+21ms)]			2.390s…2.465s (0.075s) [-0.300s…-0.557s (-0.257s)]
Load time	1.38s (130ms) [+0.23s (+43ms)]			2.628s [-1.552s]
Document complete time				2.628s [-1.552s]
loadEvent				2.627s…2.666s (0.039s) [-1.562s…-1.675s (-0.113s)]
Fully Loaded Time	1.68s [+0.29s]			2.782s [-1.839s]
Max Potential First Input Delay		492ms [-51ms]	146ms [-26ms]	
First CPU Idle		5.9s [-0.7s]	1.2s [-0.3s]	> 2.465s [-1.881s]
Start Render				1.150s [-0.540s]
Speed Index		7.1s [+0.7s]	1.7s [±0s]	2.708s [-0.988s]
RUM Speed Index	1,304 [+177]			
Last Painted Hero				3.940s [-1.680s]
Visually Complete				3.970s [-1.680s]
Number of requests	33[6] [-5]			33[6] [-4]
Page size (in bytes)	961.3KB[6] [ca. -0.68MB]			1,093KB[6] [-686KB]
PageSpeed Score	A (93%)[6] [B→A, +11pp]			
Performance score		48 [+1]	91 [+2]	
YSlow Score	A (96%)[6] [+6pp]			

[6] Constant value between the ten runs.

Looking at the very first metrics, Redirect duration stayed at 0ms and Connection duration improved by 10ms. However, Backend duration and TTFB deteriorated, possibly because the WordPress plugin Autoptimize was installed (Chapter 5.3.3), meaning that there is more code to run when loading a page.

GTmetrix reported better scores for Connection duration (as already mentioned), DOM interactive time and the four metrics with constant scores, but worse scores for all other metrics. Apart from TTFB, all metrics evaluated by WebPageTest improved. Since WebPageTest did no longer have an option to choose the same physical location as in the previous evaluation runs, the comparability of the results is unclear.

Based on the scores reported by PageSpeed Insights, the page now renders meaningful content faster (FCP and FMP improved) but takes longer to become interactive (TTI deteriorated). Speed Index stayed the same for the desktop evaluation runs and worsened for the mobile desktop runs. Other metrics improved, albeit only slightly.

5.4.2 Performance improvement suggestions

Performance improvement suggestions given by GTmetrix, PageSpeed Insights and WebPageTest are listed and described in the following three sub-chapters, respectively.

GTmetrix

Suggestions given by GTmetrix are listed in Table 12. Scores of the suggestions that improved or worsened compared to the previous values (Table 8) are highlighted in green and red, respectively. New suggestions (not present in Table 8) are in cursive. Absolute changes of the values are shown in square brackets.

Table 12. Suggestions given by GTmetrix. The scores were constant between the ten runs.

	Score
PageSpeed suggestions	
Defer parsing of JavaScript	A (93) [F→A, +64]
Optimize images	A (90) [F→A, +41]
Serve scaled images	D (63) [E→D, +6]
Minify JavaScript	A (99) [+2]
Minify CSS	A (99) [±0]
Minify HTML	A (99) [±0]
YSlow suggestions	
Use a Content Delivery Network (CDN)	A (100) [F→A, +100]
Make fewer HTTP requests	A (100) [B→A, +16]
Make JavaScript and CSS external	(n/a)
Use cookie-free domains	*F (0) [A→F, -100]*

The score for the YSlow suggestion “Use a Content Delivery Network (CDN)” grew from zero to hundred due to the CDN being set up (Chapter 5.3.1). The YSlow suggestion “Use cookie-free domains” is not present in Table 8 but has now a score of zero in Table 12. This is because Cloudflare uses cookies to “maximize network

resources, manage traffic, and protect … sites from malicious traffic" (Cloudflare, 2020b). Disabling the cookies is possible for Cloudflare's Enterprise customers but doing so will impair Cloudflare's ability to protect the site from malicious visitors. Furthermore, the performance implications of disabling the cookies would be minimal. (Cloudflare, 2020b.)

The PageSpeed suggestion "Defer parsing of JavaScript" improved greatly due to the improvements described in Chapter 5.3.3. Similarly, the YSlow suggestion "Make fewer HTTP requests" improved so that its score is now a perfect hundred. The PageSpeed suggestion "Minify JavaScript" also slightly improved. The PageSpeed suggestion "Optimize images" improved greatly, and the PageSpeed suggestion "Serve scaled images" improved slightly, both due to the image optimizations described in Chapter 5.3.2.

PageSpeed Insights

Suggestions given by PageSpeed Insights are listed in Table 13. Scores of the suggestions that improved or worsened compared to the previous values (Table 9) are highlighted in green and red, respectively. Absolute changes of the values are shown in square brackets.

Table 13. Suggestions given by PageSpeed Insights. The values are average values between the ten runs unless otherwise noted.

	Mobile	**Desktop**
Opportunities – "Estimated savings"		
Serve images in next-gen formats	0.83s [-3.19s]	– [-0.48s]
Efficiently encode images	– [-2.4s]	– [-0.32s]
Eliminate render-blocking resources	0.83s [-1.24s]	– [-0.45s]
Preload key requests	– [-1.58s]	–
Remove unused CSS	0.60s[7] [-0.30s]	0.16s[8] [-0.04s]
Defer offscreen images	0.75s[7] [+0.45s]	–
Reduce server response times (TTFB)	1.08s [+0.80s]	–
Diagnostics		
Ensure text remains visible during webfont load ("Potential Savings" in milliseconds, total of five fonts)	1ms [-113ms]	7ms [-92ms]
Minimize main-thread work (Time spent in seconds)	3.5s [-1.4s]	– [-2.1s]
Reduce JavaScript execution time (Time spent in seconds)	1.6s [-0.5s]	–

Most suggestions scored better in the re-evaluation runs, as can be seen in Table 13. Only two suggestions scored worse in the mobile runs. Two suggestions were not

[7] Constant value between the ten runs.

[8] Only present in the 3rd, 4th, 5th, 7th and 10th runs; constant value between the five runs.

suggested anymore in the mobile runs (that is, the audits passed), and similarly four suggestions in the desktop runs.

The Opportunities “Serve images in next-gen formats” and “Efficiently encode images” scored better due to the image optimizations described in Chapter 5.3.2. The Opportunity “Defer offscreen images” scored worse, but the reason for that is unclear. Another unclarity is why the Diagnostic “Ensure text remains visible during webfont load” scored better.

The Opportunity “Eliminate render-blocking resources” and the Diagnostics “Minimize main-thread work” and “Reduce JavaScript execution time” scored better, as was presumed in Chapter 5.3.3. The estimated savings for the Opportunity “Remove unused CSS” were also lower, possibly because the WordPress plugin Autoptimize minified the CSS files (as described in Chapter 5.3.3). On the other hand, installing the plugin is a possible reason why the Opportunity “Reduce server response times (TTFB)” now scored worse. The Opportunity “Preload key requests” scored expectedly better due to the improvements described in Chapter 5.3.4.

WebPageTest

Suggestions given by WebPageTest are listed in Table 14. Scores of the suggestions that improved or worsened compared to the previous values (Table 10) are highlighted in green and red, respectively. New suggestions (not present in Table 10) are in cursive. Absolute changes of the values are shown in square brackets. All scores were constant between the ten evaluation runs, expect the last score (“First Byte Time (back-end processing)”). The last score in Table 14 is thus the average value of the score.

Table 14. Suggestions given by WebPageTest. The first four scores were constant between the ten runs. The last score is the average value between the ten runs.

Suggestion	**Score**
Use Progressive JPEGs	48/100 [+48]
Use a CDN for all static assets	100/100 [+86]
Compress Images	95/100 [+40]
Leverage browser caching of static assets	*98/100 [-2]*
First Byte Time (back-end processing)	*91/100 [-9]*

Like in the case of GTmetrix, WebPageTest’s suggestion “Use a CDN for all static assets” expectedly grew all the way to the score of hundred because a CDN was set up (Chapter 5.3.1). The suggestions “Compress Images” and “Use Progressive JPEGs” also scored better because the images were optimized (Chapter 5.3.2). The latter suggestion did not score perfectly, as was presumed in the said chapter.

Table 14 contains two suggestions that were not in Table 10 (because they scored perfectly): “Leverage browser caching of static assets” and “First Byte Time (back-end processing).” The former’s deterioration was not clear from the results. The latter’s deterioration is possibly due to the WordPress plugin that was installed (Chapter 5.3.3).

6. Discussion and Implications

The research question in the study was *"How to evaluate and improve web performance using free-to-use tools?"* This question and the five sub-questions listed in Chapter 1 are answered in this chapter. Suggestions for practitioners are presented at the end of the chapter.

There are many tools for measuring web performance; Lighthouse, PageSpeed Insights, WebPageTest and Pingdom are "among the most popular and useful" (Gash, n.d.). Other tools include TestMySite and Chrome DevTools (*How To Think About Speed Tools*, 2019), and it is good to use many tools and compare their results (Gash, n.d.). Some tools, like Google's PageSpeed and Yahoo!'s YSlow, suggest best practices for optimizing web performance (Nicolaou, 2013). When optimizing for mobile devices, some of the practices should be used only occasionally or ignored altogether due to mobile networks' problems and mobile devices' inferior performance compared to desktops. (Nicolaou, 2013.)

Eight web performance evaluation tools were mentioned in the prior literature covered in Chapter 2.4, though two of them (PageSpeed and PageSpeed Insights) turned out to be just one tool, as described in Chapter 4.2. Sixteen more tools were found (Chapter 4.1) using the two search engines Startpage.com and DuckDuckGo. When researching some of the tools (Pingdom and TestMySite), two more tools with similar names were discovered (Pingdom Tools and Testmysite.io, respectively; see Chapter 4.2). Five tools were discarded for being irrelevant in this study – they focused on different kind of web performance than what is meant in this study: loading and rendering speed, as mentioned in Chapter 1. Sitespeed.io was discarded due to its perceived poor documentation and the overwhelming amount of data it produced, and YSlow was discarded for being obsolete software, as described in Chapter 4.2.

Out of the remaining eighteen tools, ten were discarded for not passing one or more of the selection criteria mentioned in Chapter 4.2: each tool can be used for free without time restrictions; each tool reports at least one quantitative web performance metric; and each tool gives suggestions how to improve the performance of the evaluated webpage. The first research sub-question *"What free-to-use tools currently exist for evaluating and improving web performance?"* can be thus answered with the following list of eight tools, in alphabetical order: GTmetrix, Lighthouse, PageSpeed Insights, Pingdom Tools, Test My Site, WebPageTest, Website Speed Test (by Dotcom-Tools) and Website Speed Test (by Uptrends). It should be noted, however, that it is possible that more tools that match the three criteria exist but were just not discovered when searching for the tools (Chapter 4.1).

The eight tools were compared feature-wise (Chapter 4.3). More specifically, the comparison covered the performance metrics evaluated by the tools (Chapter 4.3.1), the performance improvement suggestions given by the tools (Chapter 4.3.2), and six other features (Chapter 4.3.3). Lighthouse and PageSpeed Insights were found to evaluate the same metrics and to give the same suggestions. The second research sub-question *"How do the tools compare in terms of features?"* is answered in the next three paragraphs.

Webpage load times should not be presented as just a single number as there are multiple moments in the loading of a webpage (Walton, n.d.). The amounts of metrics

evaluated by the tools ranged from one (Test My Site) to fifteen (GTmetrix and WebPageTest). While there are overlaps in the evaluated metrics between the tools, using more than one tool is required to get a comprehensive combination of different metrics. Some metrics, like Load time and DOM content loaded time, are "extremely unreliable" metrics (Walton, n.d.), and some metrics, like Page size and Number of requests, do not directly evaluate performance. Some tools evaluate only unreliable and indirect metrics. Due to lack of proper documentation of some tools, it is possible that some metrics were considered as multiple metrics with different names.

The performance improvement suggestions given by the tools can be categorized as PageSpeed suggestions, YSlow suggestions and WebPageTest suggestions. GTmetrix provides 26 PageSpeed suggestions; Lighthouse, PageSpeed Insights and Website Speed Test by Uptrends provide 24; Test My site provides 22; and Website Speed Test by Dotcom-Tools provides 10 PageSpeed suggestions. GTmetrix and Website Test Tool by Dotcom-Tools apparently use an old, deprecated version of PageSpeed Insights API to provide the suggestions. All PageSpeed suggestions are listed in the table in Appendix B. GTmetrix is the only tool that provides two kinds of suggestions; in addition to the 26 PageSpeed suggestions, it provides 22 YSlow suggestions. Pingdom Tools provides 22 YSlow suggestions and was found to support one more YSlow suggestion, which was however disabled. All YSlow suggestions are listed in the table in Appendix C. WebPageTest provides seven suggestions that are neither PageSpeed nor YSlow suggestions. These suggestions are listed in Chapter 4.3.2.

Lighthouse and WebPageTest are the only tools that are open source software. All tools are web-based, except Lighthouse, which is built into the Chrome browser and available also as a Chrome browser extension and a Node module. The physical locations of the tools' test servers are listed in the table in Appendix D, excluding the locations supported by Test My Site due to lack of documentation of the locations. Some tools offered no option to choose the location. The different devices and browsers that the tools support are detailed in Chapter 4.3.3. None of the tools except Lighthouse are usable offline due to being web-based tools. Lighthouse can neither be used offline but can be used for local websites that are not available on the internet. Lighthouse is the only tool that can evaluate other aspects of a website besides performance.

The last three research sub-questions relate to the case website. Three tools – GTmetrix, PageSpeed Insights and WebPageTest – were selected (Chapter 5.1) to evaluate, improve and re-evaluate the performance of the front page of the case website. When compared to the study by Ismailova and Inal (2016) in which the authors tested the performance of several governmental websites using Pingdom [Tools], this study focused only on the front page of a single website, but utilized three tools and involved doing performance improvements and performance re-evaluations after doing the improvements. Presuming that Ismailova and Inal (2016) ran the evaluation tool on each website only once – given that the authors do not mention otherwise – another difference between the two studies is that in this study each tool was run ten times both in the first evaluation and in the re-evaluation. Calculating the average values between the ten runs made the results less sensitive to possible momentary outliers.

Based on Chapter 5.2.1, the answer to the third sub-question *"What performance results do the tools initially (before interventions) report when evaluating the performance of the case website?"* can be summarized so that the performance results of the desktop runs were in the high-end range though varying between the three tools, and the performance results of the mobile runs were noticeably slower. It is good to use many tools and "compare the analyses for both common factors and unique outliers" (Gash, n.d.). Lower performance on mobile is expectable due to the challenges of mobile

devices and mobile networks discussed by Gardner (2011), Nicolaou (2013) and Wang, Lin, Zhong and Chishtie (2011).

Answering the fourth sub-question *"What are the biggest performance bottlenecks of the case website reported by the tools?"* based on the performance metrics of the preliminary evaluation (Chapter 5.2.1) is difficult because the metrics do not have reference values of "good" or "fast" values. Another way to look at the sub-question is to see what performance improvement suggestions (Chapter 5.2.2) had the lowest score or the largest estimated savings. The common bottlenecks based on the suggestions given by the three tools seemed to be lack of using a CDN (Content Delivery Network), serving unoptimized images and serving large amounts of JavaScript. Gash's (n.d.) suggestion to use many tools and look for "common factors and unique outliers" seems to apply well for the suggestions as well.

After doing the preliminary performance evaluation (Chapter 5.2), and before doing the re-evaluation (Chapter 5.4), some of the performance improvement suggestions given by the tools were followed (Chapter 5.3). A CDN was set up (Chapter 5.3.1) and images were optimized (Chapter 5.3.2). CSS and JS files were concatenated and minified (Chapter 5.3.3) to reduce the page size and the amount of JavaScript and HTTP requests. Two key requests (as reported by PageSpeed Insights) were configured to be preloaded (Chapter 5.3.4). Finally, rationale for not following some of the suggestions was presented in Chapter 5.3.5. Because some suggestions were not followed, it is not possible to see the greatest possible "power" of the tools, that is, how much the performance would have improved if all of the suggestions were carefully followed. When working on a real website with a limited amount of available time and resources, however, some constraints may be unavoidable. It might be beneficial then to focus on the suggestions with the lowest scores or largest estimated savings to remedy the biggest performance bottlenecks. It is also unclear which tool gave the most effective suggestions since all tools' suggestions were combined before following them. Comparing the effectiveness of the different tools' suggestions was however not the main objective of the study, as indicated by the research question and sub-questions.

The fifth sub-question *"How does the performance of the case website improve after implementing the performance improvements suggested by the tools?"* can be answered based on the results of the performance re-evaluation (Chapter 5.4). The number of HTTP requests dropped by five and the page size in bytes reduced by about 42%, as evaluated by GTmetrix. However, most time-based metrics evaluated by the tool deteriorated (Chapter 5.4.1): seven metrics deteriorated, two improved and one stayed constant. Despite that, the relative metrics PageSpeed Score and YSlow Score improved by 11 and 6 percentage points, respectively. Because most suggestions scored better in the re-evaluation (Chapter 5.4.2), it seems plausible to think that the two said performance metrics are based more on the suggestion scores than on the other, time-based performance metrics. Since the website's users will not see the scores or values of the performance metrics, but will experience the loading and rendering delays, relative performance metrics should be used only as guidelines. The main goal should be to improve the time-based performance metrics. Relative performance metrics can be even deceptive, like in this study (PageSpeed Score and YSlow Score improved but most of the other performance metrics deteriorated).

Based on the re-evaluation results by PageSpeed Insights, the main result seems to be that now the page renders faster (FCP and FMP lowered on both mobile and desktop; see Chapter 2.3 for descriptions of the metrics) but becomes interactive more slowly (TTI increased on both mobile and desktop). In the case of WebPageTest, one metric (TTFB) worsened, possibly due to a WordPress plugin being installed (Chapter 5.3.3), but all other 14 metrics improved. However, since the physical location that was used in

the preliminary performance evaluation could not be chosen in the tool when doing the re-evaluation, it is unclear whether the results were affected by the performance improvements, by the different locations, or by both. This flaw could be avoided in future studies by limiting the time span between the preliminary evaluation and the re-evaluation. In this study, the time span was a full month.

The improved performance of the case website can be also seen in the new suggestions (Chapter 5.4.2). Six suggestions given by GTmetrix scored better, and only one suggestion scored worse. Two suggestions given by PageSpeed Insights deteriorated for the mobile runs, but all other suggestions improved, some so much that they were not suggested in the re-evaluation. All three suggestions given by WebPageTest in the preliminary performance evaluation improved significantly, and two other suggestions deteriorated, but only slightly. Since the suggestions are only suggestions, they should also be used only as guidelines, whereas the main goal should be to improve the (time-based) performance metrics.

Based on the results presented in this chapter, the following four suggestions can be raised for practitioners (in no particular order):

1. Since the tools evaluate and suggest different things, it is good to use many tools to get a wider variety of performance metrics evaluated and more performance improvement suggestions.
2. Optimizing for mobile devices seems to be more difficult compared to desktops, so it would be beneficial to configure the tools to use mobile devices for the evaluation. In this study, the mobile performance was evaluated only by PageSpeed Insights.
3. Since time and resources are finite, it might be beneficial to focus on the biggest performance bottlenecks first. Comparing the different tools' suggestions and looking for "common factors and unique outliers" (Gash, n.d.) could be used as an alternative approach. Some suggestions given by the tools are straightforward (e.g. preloading key requests), while some suggestions mainly point out specific performance issues, so following some suggestions may require researching the topic further.
4. Following some suggestions may reduce performance (as an example, the case website's TTFB clearly deteriorated in this study), so it would be beneficial to re-evaluate the performance after doing each performance improvement to see if the performance improved or not. Data-driven performance optimization (re-evaluate after each improvement) gives a clearer picture of each improvement compared to blindly following the suggestions given by the tools. It is not clear what improvements in this study were good and what were bad, because all improvements were implemented before doing the re-evaluation.

Ultimately, the suggestions as well as relative performance metrics should be used only as guidelines. The main goal should be to improve the time-based performance metrics, since they determine in the end how the users perceive the loading and rendering of the webpage (see Walton, n.d.; Chapter 2.3).

7. Conclusion

The study provided a detailed, up-to-date comparison of eight free-to-use tools that can both evaluate the performance (loading and rendering speed) of a website and suggest how the performance could be improved. The comparison (Chapter 4) covered the performance metrics and the performance improvement suggestions that the tools report as well as six other features of the tools that can be useful to know in practice. The author is not aware of similar prior comparisons that cover so many details of so many tools as this study.

In addition, three tools were used on an existing WordPress website. The preliminary performance evaluation results were reported (Chapter 5.2), the suggestions that the tools gave were acted upon where possible (Chapter 5.3), and the performance re-evaluation results were reported and compared to the preliminary results (Chapter 5.4). The author is not aware of prior academic studies in which web performance has been both evaluated and improved like in this study.

However, the study has a few limitations as well. First, the study covers only free-to-use tools that can both evaluate the performance of a website and give suggestions how to improve the performance. There are numerous tools that are either paid and/or do not give performance improvement suggestions (and there are possibly also tools that can give suggestions but not evaluate performance); such tools were not part of the study. Second, since the way of searching for the tools was rather simple (Chapter 4.1), it is possible that there are more tools that match the selection criteria set in Chapter 4.2; such tools were also not part of the study. Third, since some suggestions given by the tools were not followed (Chapter 5.3.5), the greatest possible "power" of the tools is not apparent, that is, how much the performance would have improved if all of the suggestions were carefully followed.

In future studies, the effects of individual improvements (based on a single suggestion given by a tool) could be evaluated more rigorously; in this study, all improvements were made before doing the re-evaluation. The results between individual tools could also be researched; this study focused more on the differences between the website's performance before and after the improvements. Studying the effects of improved web performance – for example tracking how it affects conversion rates or search engine rankings – is a large and separate topic on its own and may require a long time span to see the effects reliably, but that topic could nevertheless be explored more in the context of utilizing the tools covered in the study. And since web performance can mean any moment during the usage of a webpage (Walton, n.d.), studying the other kinds of web performance than loading and rendering speed could also be done in future studies.

Since the experiment targeted just a single website and all websites are different, the details and results of the experiment are not generally applicable outside the study. However, the practical information presented in the study – the details of the comparison of the eight tools as well as the details how the three tools were used on the case website – can be utilized both in future research and by practitioners to evaluate and improve the performance of other websites. Ideally, multiple tools are used to get a wide variety of performance metrics and suggestions, and the suggestions and relative performance scores given by the tools are used only as guidelines with the main goal being to improve the time-based performance metrics.

References

Aberdeen Group. (2008). *The Performance of Web Applications: Customers Are Won or Lost in One Second* [Research report]. Retrieved November 22, 2019 from https://info.headspin.io/hubfs/Analyst%20Reports/5136-RR-performance-web-application.pdf

About PageSpeed Insights. (2019). Retrieved February 21, 2020 from https://developers.google.com/speed/docs/insights/v5/about

An, D. (2018). *Find out how you stack up to new industry benchmarks for mobile page speed.* Retrieved September 8, 2019 from https://www.thinkwithgoogle.com/marketing-resources/data-measurement/mobile-page-speed-new-industry-benchmarks/

Brutlag, J. (2009). *Speed Matters.* Retrieved November 17, 2019 from https://ai.googleblog.com/2009/06/speed-matters.html

Cloudflare. (2020a). *The Cloudflare Global Anycast Network.* Retrieved April 5, 2020 from https://www.cloudflare.com/network/

Cloudflare. (2020b). *Understanding the Cloudflare Cookies.* Retrieved April 13, 2020 from https://support.cloudflare.com/hc/en-us/articles/200170156

DuckDuckGo. (n.d.). *Welcome to DuckDuckGo.* Retrieved February 4, 2020 from https://duckduckgo.com/about

Enable Compression. (2018). Retrieved February 18, 2020 from https://developers.google.com/speed/docs/insights/EnableCompression

Frequently Asked Questions. (n.d.). Retrieved February 15, 2020 from https://www.thinkwithgoogle.com/feature/testmysite/faq/

Galletta, D. F., Henry, R., McCoy, S., & Polak, P. (2004). Web Site Delays: How Tolerant are Users? *Journal of the Association for Information Systems, 5*(1), 1-28.

Galletta, D., Henry, R. M., McCoy, S., & Polak, P. (2006a). Understanding the Direct and Interaction Effects of Web Delay and Related Factors: A Research Program. In D. Galletta & P. Zhang (Eds.), *Human-Computer Interaction and Management Information Systems: Applications. Advances in Management Information Systems: Vol. 6* (pp. 29-69). Armonk, NY: M.E. Sharpe.

Gash, D. (n.d.). *Measuring Performance.* Retrieved January 20, 2020 from https://developers.google.com/web/fundamentals/performance/get-started/measuringperf-2

Gehrke, D., & Turban, E. (1999). Determinants of successful website design: relative importance and recommendations for effectiveness. *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences.*

González, F. J. M., & Palacios, T. M. B. (2004). Quantitative evaluation of commercial web sites: an empirical study of Spanish firms. *International journal of information management, 24*(4), 313-328.

Grigorik, I. (Ed.). (2020). *Navigation Timing Level 2: W3C Editor's Draft 21 January 2020.* Retrieved April 13, 2020 from https://w3c.github.io/navigation-timing/

GT.net. (2020). *Frequently Asked Questions.* Retrieved February 23, 2020 from https://gtmetrix.com/faq.html

Gustedt, J., Jeannot, E., & Quinson, M. (2009). Experimental validation in large-scale systems: a survey of methodologies. *Parallel Processing Letters, 19*(3), 399-418.

Halepovic, E., Pang, J., & Spatscheck, O. (2012). Can you GET Me Now? Estimating the Time-to-First-Byte of HTTP Transactions with Passive Measurements. *Proceedings of the 2012 Internet Measurement Conference*, 115-122.

Hernández, B., Jiménez, J., & Martín, M. J. (2009). Key website factors in e-business strategy. *International Journal of Information Management, 29*(5), 362-371.

How To Think About Speed Tools. (2019). Retrieved January 20, 2020 from https://developers.google.com/web/fundamentals/performance/speed-tools

Hoxmeier, J. A., & DiCesare, C. (2000). System Response Time and User Satisfaction: An Experimental Study of Browser-based Applications. *AMCIS 2000 Proceedings*, 347.

Ismailova, R., & Inal, Y. (2016). Web site accessibility and quality in use: a comparative study of government Web sites in Kyrgyzstan, Azerbaijan, Kazakhstan and Turkey. *Universal Access in the Information Society, 16*(4), 987-996.

Kinnunen, M. (2016). Verkkosivuston latautumisnopeuden vaikutus käyttäjän tyytyväisyyteen. Retrieved September 8, 2019 from http://urn.fi/URN:NBN:fi:oulu-201603311368

Lightner, N. J., Bose, I., & Salvendy, G. (1996). What is wrong with the World-Wide Web?: a diagnosis of some problems and prescription of some remedies. *Ergonomics, 39*(8), 995-1004.

Mateos, M. B., Mera, A. C., González, F. J. M., & López, Ó. R. G. (2001). A new Web assessment index: Spanish universities analysis. *Internet research, 11*(3), 226-234.

McKetta, I. (2019). *In-Depth Analysis of Changes in World Internet Performance Using the Speedtest Global Index.* Retrieved September 6, 2019 from https://www.speedtest.net/insights/blog/global-index-2019-internet-report/

Miller, J., & Osmani, A. (2019). *Rendering on the Web.* Retrieved December 13, 2019 from https://developers.google.com/web/updates/2019/02/rendering-on-the-web

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proceedings of the AFIPS Fall Joint Computing Conference*, 267-277.

Miranda, F. J., Cortés, R., & Barriuso, C. (2006). Quantitative Evaluation of e-Banking Web Sites: an Empirical Study of Spanish Banks. *Electronic Journal of Information Systems Evaluation, 9*(2), 73-82.

Miranda, F. J., Sanguino, R., & Bañegil, T. M. (2009). Quantitative assessment of European municipal web sites: Development and use of an evaluation tool. *Internet Research, 19*(4), 425-441.

Nah, F. F.-H. (2004). A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour & Information Technology, 23*(3), 153-163.

Nicolaou, A. (2013). Best Practices on the Move: Building Web Apps for Mobile Devices. *Communications of the ACM, 56*(8), 45-51.

Nielsen, J. (1993). *Response Times: The 3 Important Limits.* Retrieved December 6, 2019 from https://www.nngroup.com/articles/response-times-3-important-limits/

Nielsen, J. (1995). *Guidelines for Multimedia on the Web.* Retrieved December 6, 2019 from https://www.nngroup.com/articles/guidelines-for-multimedia-on-the-web/

Nielsen, J. (1996). *Original Top 10 Mistakes in Web Design.* Retrieved December 6, 2019 from https://www.nngroup.com/articles/original-top-ten-mistakes-in-web-design/

Nielsen, J. (1997). *The Need for Speed.* Retrieved December 6, 2019 from https://www.nngroup.com/articles/the-need-for-speed/

Nielsen, J. (1999a). *"Top Ten Mistakes" Revisited Three Years Later*. Retrieved November 17, 2019 from http://www.nngroup.com/articles/top-ten-mistakes-revisited-three-years-later/

Nielsen, J. (1999b). *The Top 10 Web Design Mistakes of 1999.* Retrieved September 8, 2019 from http://www.nngroup.com/articles/the-top-ten-web-design-mistakes-of-1999/

Nielsen, J. (1999c). *Who Commits The "Top Ten Mistakes" of Web Design?* Retrieved November 17, 2019 from http://www.nngroup.com/articles/who-commits-the-top-ten-mistakes-of-web-design/

Pock, D. (2019). *What is a progressive JPEG?* Retrieved April 12, 2020 from https://www.liquidweb.com/kb/what-is-a-progressive-jpeg/

Ramsay, J., Barbesi, A., & Preece, J. (1998). A psychological investigation of long retrieval times on the World Wide Web. *Interacting with Computers, 10*(1), 77-86.

Rempel, G. (2015). Defining Standards for Web Page Performance in Business Applications. *ICPE '15 Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, 245-252.

Rodman, T. (2015). *Web Page Sizes: A (Not So) Brief History of Page Size through 2015.* Retrieved September 8, 2019 from https://www.yottaa.com/a-brief-history-of-web-page-size/

Rose, G. M., Lees, J., & Meuter, M. L. (2001). A refined view of download time impacts on e-consumer attitudes and patronage intentions toward e-retailers. *International Journal on Media Management, 3*(2), 105-111.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering, 14*(2), 131-164.

Sears, A., Jacko, J. A., & Borella, M. S. (1997). Internet Delay Effects: How Users Perceive Quality, Organization, and Ease of Use of Information. *CHI'97 Extended Abstracts on Human Factors in Computing Systems*, 353-354.

Singhal, A, & Cutts, M. (2010). *Using site speed in web search ranking.* Retrieved September 8, 2019 from https://webmasters.googleblog.com/2010/04/using-site-speed-in-web-search-ranking.html

Sitespeed.io. (2020). *Welcome to the wonderful world of Web Performance.* Retrieved February 10, 2020 from https://www.sitespeed.io/

Speed Index. (2019). Retrieved February 23, 2020 from https://web.dev/speed-index/

Startpage.com. (n.d.). *The world's most private search engine.* Retrieved February 4, 2020 from https://www.startpage.com/

Tedre, M., & Moisseinen, N. (2014). Experiments in Computing: A Survey. *The Scientific World Journal, 2014.*

The Spooniest. (2015). *Does minifying and concatenating JS/CSS files, and using sprites for images still provide performance benefits when using HTTP/2?* [Answer]. Retrieved April 5, 2020 from https://stackoverflow.com/questions/28630108/does-minifying-and-concatenating-js-css-files-and-using-sprites-for-images-stil/28631462#28631462

Time to Interactive. (n.d.). Retrieved January 10, 2019 from https://developers.google.com/web/tools/lighthouse/audits/time-to-interactive

Waldrop, M. M. (2016). The chips are down for Moore's law. *Nature, 530*(7589), 144-147.

Walton, P. (n.d.). *User-centric Performance Metrics.* Retrieved December 13, 2019 from https://developers.google.com/web/fundamentals/performance/user-centric-performance-metrics

Walton, P. (2018). *First Input Delay.* Retrieved January 10, 2019 from https://developers.google.com/web/updates/2018/05/first-input-delay

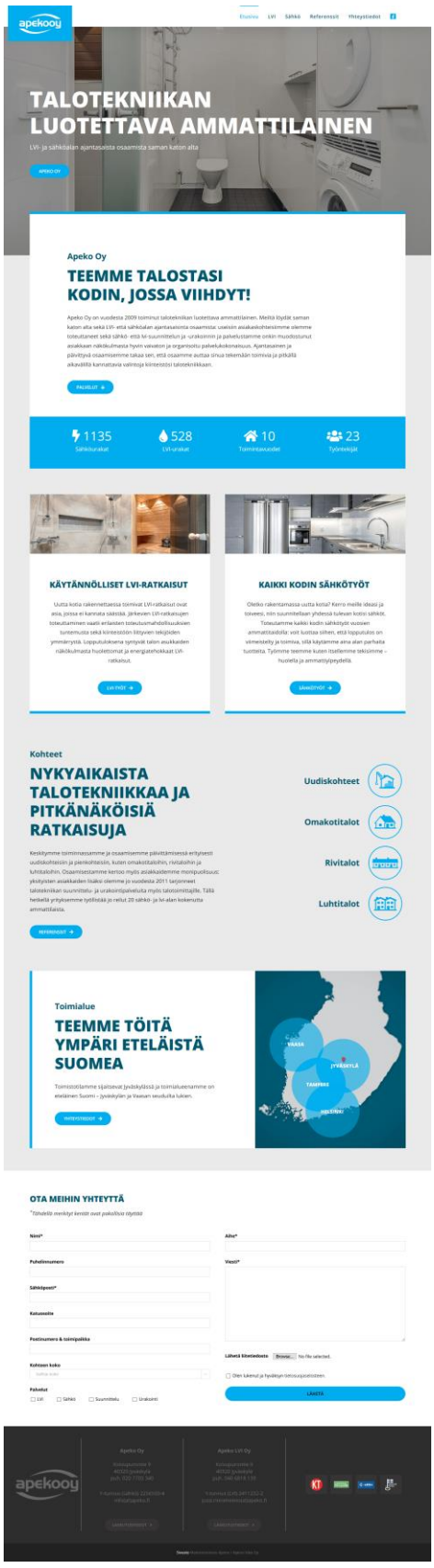
Wang, Z., & Phan, D. (2018). *Using page speed in mobile search ranking.* Retrieved September 8, 2019 from https://webmasters.googleblog.com/2018/01/using-page-speed-in-mobile-search.html

Weinberg, G. (2018). *Is it true that DuckDuckgo uses Google search servers and algorithms instead of proprietary ones?* Retrieved February 4, 2020 from https://www.quora.com/Is-it-true-that-DuckDuckgo-uses-Google-search-servers-and-algorithms-instead-of-proprietary-ones/answer/Gabriel-Weinberg

WordPress.com. (n.d.). *WordPress powers 36% of the internet.* Retrieved March 25, 2020 from https://wordpress.com/

Yahoo Developer Network. (n.d.). *Best Practices for Speeding Up Your Web Site.* Retrieved April 5, 2020 from https://developer.yahoo.com/performance/rules.html

Appendix A. Full screenshot of the front page



Appendix B. PageSpeed suggestions

Suggestion	**Lighthouse, PageSpeed Insights, Website Speed Test (by Uptrends)**	**Test My Site**	**GTmetrix**	**Website Speed Test (by Dotcom-Tools)**
All text remains visible during webfont loads	✓	✓ (“Ensure text remains visible during webfont load”)		
AMP your page		✓		
Avoid a character set in the meta tag			✓	
Avoid bad requests			✓	
Avoid chaining critical requests	✓	✓ (“Minimise [*sic*] critical requests depth”)		
Avoid CSS @import			✓	
Avoid landing page redirects			✓	✓
Avoid multiple page redirects	✓	✓	✓ (“Minimize redirects”)	
Avoids an excessive DOM size	✓	✓		
Avoids enormous network payloads	✓	✓		
Combine images using CSS sprites			✓	
Defer offscreen images	✓	✓		
Defer parsing of JavaScript			✓	
Efficiently encode images	✓	✓	✓ (“Optimize images”)	✓
Eliminate render-blocking JavaScript and CSS in above-the-fold content				✓
Eliminate render-blocking resources	✓	✓		
Enable Keep-Alive			✓	
Enable text compression	✓	✓	✓ (“Enable compression”)	✓ (“Enable compression”)
Inline small CSS			✓	
Inline small JavaScript			✓	
Keep request counts low and transfer sizes small	✓			

Leverage browser caching			✓	✓
Minify CSS	✓	✓	✓	✓
Minify HTML			✓	✓
Minify JavaScript	✓	✓	✓	✓
Minimize main-thread work	✓			
Minimize request size			✓	
Minimize third-party usage	✓			
Optimize the order of styles and scripts			✓	
Preconnect to required origins	✓	✓		
Prefer asynchronous resources			✓	
Preload key requests	✓	✓		
Prioritize visible content				✓
Properly size images	✓	✓	✓ (“Serve scaled images”)	
Put CSS in the document head			✓	
Reduce JavaScript execution time	✓	✓		
Reduce server response times (TTFB)	✓	✓ (“Resolve slow server response times”)		✓ (“Reduce server response time”)
Remove unused CSS	✓	✓ (“Defer unused CSS”)		
Serve images in next-gen formats	✓	✓		
Serve resources from a consistent URL			✓	
Specify a cache validator			✓	
Specify a character set early			✓	
Specify a Vary: Accept-Encoding header			✓	
Specify image dimensions			✓	
Use video formats for animated content	✓	✓		
User Timing marks and measures	✓	✓ (“Instrument your scripts with User Timing marks and measures”)		
Uses efficient cache policy on static assets	✓	✓ (“Serve static assets with an efficient cache policy”)		

Appendix C. YSlow suggestions

Suggestion	GTmetrix	Pingdom Tools
Add Expires headers	✓	✓
Avoid AlphaImageLoader filter	✓	✓
Avoid CSS expressions	✓	✓
Avoid empty src or href		✓
Avoid HTTP 404 (Not Found) error	✓	✓
Avoid URL redirects	✓	✓
Compress components	✓	✓ ("Compress components with gzip")
Configure entity tags (ETags)	✓	✓
Do not scale images in HTML		✓
Make AJAX cacheable	✓	✓
Make favicon small and cacheable	✓	✓
Make fewer HTTP requests	✓	✓
Make JavaScript and CSS external	✓	✓
Minify JavaScript and CSS	✓	✓
Put CSS at top		✓
Reduce cookie size	✓	✓
Reduce DNS lookups	✓	✓
Reduce the number of DOM elements	✓	✓
Remove duplicate JavaScript and CSS	✓	✓
Use a Content Delivery Network (CDN)	✓	✓ (but disabled)
Use cookie-free domains	✓	✓
Use GET for AJAX requests	✓	✓

Appendix D. Physical locations of the tools' test servers

	GTmetrix	Pingdom Tools	WebPage-Test	Website Speed Test (by Dotcom-Tools)	Website Speed Test (by Uptrends)
Africa					
Mauritius			✓		
South Africa				✓	
Asia					
China	✓		✓	✓	✓
Hong Kong				✓	
India	✓		✓	✓	
Indonesia			✓		
Japan		✓	✓	✓	
Singapore			✓		✓
South Korea			✓		
Vietnam			✓		
Europe					
Belgium			✓		
Czech Republic			✓		
France			✓	✓	✓
Germany		✓	✓	✓	✓
Ireland			✓		
Netherlands			✓	✓	✓
Poland			✓	✓	
Spain			✓	✓	
Sweden			✓		
Switzerland			✓		
United Kingdom	✓	✓	✓	✓	✓
Middle East					
Israel			✓	✓	
Turkey			✓		
United Arab Emirates			✓		

North America					
Canada	✓		✓	✓	
US, California		✓	✓	✓	✓
US, Colorado				✓	
US, District of Columbia		✓		✓	
US, Florida			✓	✓	
US, Illinois			✓		
US, Minnesota				✓	
US, Nebraska			✓		
US, New Jersey			✓		
US, New York				✓	✓
US, Texas	✓		✓	✓	
US, Virginia			✓	✓	
US, Washington				✓	
Oceania					
Australia	✓	✓	✓	✓	✓
South America					
Argentina			✓	✓	
Brazil	✓	✓	✓		