Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Master's Thesis

Performance-optimized A/B-Testing for E-Commerce-Websites

Aram Yesildeniz

aram.yesildeniz@studium.uni-hamburg.de
M. Sc. Informatics
Matr.-No. 6890370

1st supervisor: Dr. Wolfram Wingerath
2nd supervisor: Benjamin Wollmer

Hamburg, 1st of September 2021

A distributed system is one where the failure of some
computer I've never heard of can keep me from getting my work done.
– *Leslie Lamport*

Contents

1 Introduction and Background

[tbd]

- Explain structure and main goal of this thesis
- Describe shortly all sections from this chapter and what the reader can expect
- Give short outlook to following chapter

1.1 E-Commerce

This chapter gives an introduction to e-commerce. Before going into the details, the emergence of e-commerce is described, starting with the Internet in section 1.1.1. In the following, I will briefly discuss the history of e-commerce and its types in section 1.1.2. The relationship between user satisfaction and the website performance is covered in section 1.1.3, which then leads to the chapter 1.2 which is about web analytics.

1.1.1 The Internet

In the last 50 years, a new technology emerged, spread over the entire world and influenced many aspects of most peoples life. Within the turmoil of the cold war, the United State's *Advanced Research Projects Agency* (ARPA) established in 1957 a communication network to bring together universities and their researches all around the country in order to be able compete against the USSR [CA11]. What started as a tool for scientific collaboration evolved half a century later into the *Internet*, a global network and phenomenon, to which every user with a dedicated device has access and can contribute to. The internet is an integral part, if not the backbone of today's everyday life. Users of the internet use it for almost everything, from sending emails, watching television, chatting with friends, order lunch, checking the weather for the next day or renting motorized scooters.

In 2021, the internet has 4.66 billion users, which is around 60% of the world population.[1] Compared to 2020, the number of internet users increased by 7.3%. In Europe, more than 90% of the population are internet users. For a developed country like Germany, the numbers are even more impressing: 94% of the German population are using the internet with an average daily time of over five hours.

[1] Following statistics are taken from `https://datareportal.com/reports/digital-2021-germany` [14.05.2021]

Those numbers demonstrate impressive that the internet is an integral part of our daily life. Along the rise of the internet, transactions and processes falling under the term of e-commerce are climbing as well. Before discussing the term "e-commerce" and take a grasp at its history and types, some statistics are presented to demonstrate the importance of e-commerce.

1.1.2 E-Commerce

Introduction

From the global data report[2], one can read out that over 90% of the world population visited an online retail site and over 76% of the world population purchased a product online. As usual, for or a western country like Germany, the figures are higher: 92.5% of the German population visited an online retail site and over 80% purchased a product online. And the usage is expanding: the growth of the amount spent within the category food and personal care is 28.6%, and 17.6% for the category fashion and beauty.

E-commerce sales have grown steadily over the past 20 years, topping to 57.8 billion in 2019.[3]

The COVID-19 pandemic with its implications had and still has an not negligible impact on the growth of e-commerce. Several measures were taken to stop the spread of the virus and the number of deaths, one of which was to minimize physical interaction between people. This leads consequently to a shift of human interactions to the internet. Along this, e-commerce benefits. Bhatti et al. [BAB+20] conclude that "e-commerce enhanced by COVID-19".

Brief History

E-Commerce, or electronic commerce, is according to the *Encyclopædia Britannica* about "maintaining relationships and conducting business transactions that include selling information, services, and goods by means of computer telecommunications networks."[4] In short, e-commerce is about buying and selling products and services via the internet.

The success of e-commerce is closely linked to the tremendous advances in Internet technology in recent years: The development of the *Electronic Data Interchange* (EDI) starting in the 1960s standardised the communication between two machines. Personal computers were introduced in the 1980s, and one of the first examples of an online shop is the *Electronic Mall* opened by CompuServe in 1984. Another crucial milestone is the launch of the *World Wide Web* (WWW) in 1990, which made the internet accessible to everyone. With social media visible on the horizon from the 2000s, new possibilities for

[2] https://datareportal.com/reports/digital-2021-germany [14.05.2021]
[3] https://einzelhandel.de/presse/zahlenfaktengrafiken/861-online-handel/1889-e-commerce-umsaetze [14.05.2021]
[4] https://www.britannica.com/technology/e-commerce [19.05.2021]

businesses and consumers alike to participate in e-commerce arise, for example, by enabling new marketing strategies or providing new sales channels. New devices such as smart phones and tablets lowered the barrier to participate in e-commerce. While e-commerce was available at any time, the new devices brought flexibility and mobility, making e-commerce available everywhere [Her19].

With the continued advancement in technology, e-commerce can expect a bright future with trends such as AI recommendation systems, outstanding UX thanks to virtual reality, or even more simpler payment methods through cryptocurrencies.[5]

Types

There are several types in e-commerce and they emerge from the possible combinations between the actors *business*, *consumer* and *government* [SSMG17].

	Business	Consumer	Government
Business	B2B	B2C	B2G
Consumer		C2C	C2G
Government			G2G

Table 1.1: Types of e-commerce.

B2C Business to Consumer in e-commerce describes basically online shopping, by means of a business offering its services and products to the consumer over the *WWW*. The consumer can browse through the products and services presented within an online shop and order them directly via the website. A variety of payment and delivery options conclude the B2C type [Hei20].

For an aspiring business, there are several ready-made software solutions for setting up an online store, as for example. *Shopify*, *ePages*, *Magento* or *WooCommerce* [SBR+19].

A famous example of a B2C company is *Amazon*. On the 16th of July in 1995, Amazon launched as a website and entered the stock market on the 15th of May 1997 [SB19]. Amazon has been successful, with the stock starting at $1.5, which is at around $3200 as of this writing.[6] Today, Amazon employs over 1 million people[7] and serves the desires of 200 million paying prime members.[8]

By taking a quick look at the pros and cons of an online store, it becomes clear that some of the advantages are that: there is no need of a real, physical store to showcase and sell the products; the virtual shop is available to the consumer at any time and has no closing

[5] https://www.spiralytics.com/blog/past-present-future-ecommerce/ [19.05.2021]
[6] https://finance.yahoo.com/quote/AMZN?p=AMZN [19.05.2021]
[7] https://www.statista.com/statistics/234488/number-of-amazon-employees/ [19.05.2021]
[8] https://www.statista.com/statistics/829113/number-of-paying-amazon-prime-members/ [20.05.2021]

hours; there is a high potential for the online shop as it is part of growing market; online business is scalable; due to tracking algorithms, precise targeting as well as data analysis is possible; to start an online business, there is not so much floating required and there are generally lower costs; it is possible to provide a personalized customer experience.

Some disadvantages are that the speed of market is rapid, competitors arise everyday everywhere and technology evolves quickly while consumers expectations go high [Her19], [LO20].

Another disadvantage is that there is no direct or physical connection with the consumer. As described above, online shopping takes place on the virtual WWW, i.e. personal interaction between buyer and seller is not possible and the shopping experience takes place on a website, from which it follows that the overall virtual user experience must be excellent in order to compete.

In the next section, I will describe the findings between the correlation between user satisfaction and the performance of the retailers web presence.

1.1.3 User Satisfaction and Performance

The aim of this thesis is not to deep dive into terms and concepts or the non-trivial problem of defining user satisfaction, usability or the like. Therefore the term user satisfaction is in this context loosely defined as how happy the user is with the website he or she interacts with.[9]

Performance can be understood as the speed of an online shop, e.g. how long it takes the page to load, how quickly the user can interact with the page, and how the user perceives the performance of the website. In chapter X I will discuss that measuring performance is not so trivial and there are a lot of ideas and metrics to measure it.

SpeedHub

A plethora of information and studies about the phenomenon of user satisfaction and web site performance is collected at *SpeedHub.org*, a portal by *Baqend* in cooperation with *Google* which provides "the largest systematic study of Mobile Site Speed and the Impact on E-Commerce."[10] Not only are studies and reports available on the hub, but also collections of videos and blog posts.

In his talk at code.talks 2019, Felix Gessert summarizes the results and provides insights into the most important aspects and questions of the study so far [Ges21]:

The first observation when asking for a correlation between the performance of a system and user satisfaction is that users need to be differentiated, which leads to the concept of a *User Profile*: In terms of gender, young women are the most demanding consumers

[9] For a discussion cf. "User satisfaction measurement" in [IKOK10]
[10] `https://www.speedhub.org/` [21.05.2021]

and are less likely to buy from slow sites. In general, people between the ages of 18 and 24 have higher expectations of a site's speed than their older counterparts.

There are also differences between nations and regions, for example people from Japan have the highest expectations, which is almost certainly related to technological advancements in that country. Not only the expectations themselves differ geographically, but also how speed influences the users, for example "speed influences New Yorkers more than Californians." [Unb], [GA]

What all users have in common is their human psychology. In terms of performance, researchers generally suggest keeping wait times below one second to keep users' attention. (cf. "Performance perception" in 1.2.5).

After considering the user himself, the next step is to examine the influence of the device used: Studies show that mobile users are more likely to buy products and services than their colleagues using a desktop computer, where iOS users have generally more expectations regarding site speed [Dev].

Last but not least, the context and state of the user is important, with naturally relaxed and calm users perceiving pages faster than stressed or hurried users. Also users experience websites more slowly while on the go [Akab].

There are many real world examples and studies that prove and demonstrate the importance of website speed in terms of user satisfaction and ultimately sales: *Amazon* found out that a decrease of 100 ms in page loading leads to -1% conversion rates. If the site loads 100 ms faster, *Walmart* observed that the revenue increases by 1%. For *Zalando*, increasing site speed by 100 ms has led to an uplift of 0.7% revenue per session [Lin], [SKG+], [CKR].

Search engine optimization is heavily impacted by load speed: For *Google*, 500 ms slower sites led to a decrease of 20% in traffic. *GQs* traffic increased by 80% after the page load went down from 7 s to 2 s. And for *Pinterest*, 40% faster loads led to 15% more SEO traffic [Mos], [May], [MAC].

User engagement and satisfaction also depend heavily on loading times: *Forrester* noted an increase of 60% for the session length while brining down the load time by 80%. *Akamai* monitored that the bounce rate climbed up incredible 103% when the load time increased by 2 seconds. And for the *AberdeedGroup*, the customer satisfaction dropped by 16% at one more second delay in response times [For], [Akaa], [Abe].

In summary, it can be said that many studies and practical examples prove and demonstrate that faster websites and online stores lead to a better user experience and usually to happier customers. In commercial terms, one can conclude that page speed equals money.

In order to properly test the effects of performance on users, a scientific method is required. A/B testing as a controlled experiment is one of them and will be explained in

the next section. After discussing A/B testing, I will move on to examining *Web Analytics*, a term that encompasses methods, tools, and instruments for companies to better understand their business and customers.

A/B Testing

Controlled experiments like A/B testing are not a new tool for scientists and researchers and were used as early as the 1920s [KL17]. With the advent of the Internet in the 1990s, the concept was adopted into the online domain and is now used by large companies such as Amazon, Facebook or Google to test ideas and hypotheses directly on a live system. Controlled experiments such as A/B testing are used to aid decision making and provide a "causal relationship with high probability" [KL17]. They enable a data-driven and quantitative validation of the hypothesis [Mor18].

Controlled experiments help to test hypothesis and questions of form: "If I change feature X, will it help to improve the key performance indicator Y?"

To answer this question, two systems are needed: *Version A*, the control variant or default version, and a slightly different *Version B,* called the treatment. If more than two versions or one treatment should be evaluated at the same time, an A/B/n split test has to be implemented. With a univariable setup, only one variable differs between the systems; with a multivariable structure, several variables are changed at the same time.

Usually, the users of the system are randomly split into two groups and testing is directly performed with real users on a production system. It is advantageous, also compared to other experimental set-ups, that the users and participants are not aware that they are part of an experiment, which leads to fewer biases and side effects. In order to measure the differences and the user behaviour, web analytics has to be integrated within the system [KL17].

A brief and general discussion of controlled experiments in computer science can be found in chapter X.

To continue with the question of performance and user satisfaction, A/B testing allows two different versions of the same site to be served to two groups at the same time, one site being slow and the other being fast, without users knowing.

An implemented web analytics system makes it possible to measure how the various systems and user groups behave. What web analytics exactly is, what tools are available and what a web analytics process looks like, is discussed in the next section.

1.2 Web Analytics

This chapter is about web analytics. I will first discuss definitions and give a short introduction to web analytics in section 1.2.1. Then, a brief history of web analytics will be given in section 1.2.2 to help contextualize. After characterizing two web analytics

process descriptions in section 1.2.3, data collection methods will be discussed in section 1.2.4. Finally, web performance will be explained in 1.2.5, which leads the way to the research questions.

1.2.1 Introduction

What is *Web Analytics*? Reviewing the literature, it is clear that there are several definitions:

Nakatani et al. state that "Web analytics is used to understand online customers and their behaviors, design actions influential to them, and ultimately foster behaviors beneficial to the business and achieve the organization's goal." [NC11] According to this definition, web analytics is about getting insights of the users using the system, not only who or what they are, but also how they interact with the system. Additionally, the definition emphasizes that the underlying motivation of web analytics is to achieve business goals.

Singal et al. provide a more technical definition by pointing out that "Web Analytics is the objective tracking, collection, measurement, reporting and analysis of quantitative internet data to optimize websites and web marketing initiatives." [SKS14] Again, the ultimate goal is to drive business, but supported by data science methods and tools such as tracking, collecting and analysing massive amounts of data.

Bekavac et al. provide a similar definition by pointing out that web analytics is "the analysis of qualitative and quantitative data on the website in order to continuously improve the online experience of visitors, which leads to more efficient and effective realization of the company's planned goals." [BGP15]

Summarizing the above definitions, it is noticeable that web analytics consists of two important elements: a data-driven, information-oriented and technical element of collecting and analysing data about users and a commercial and business-driven element that provides the main motivation for collecting the data primarily by setting business goals.

Moving from definitions to the practical realm, Zheng et al. describe four main use cases for web analytics [ZP15]:

- Improving the overall design and usability, for example of the navigation or layout of the website.
- Optimize for your business goals: Whatever goals the business is trying to achieve, generating conversions is the goal.
- Monitor campaigns: Understand and measure the success of advertising campaigns.
- Improve performance by examining metrics such as page load time. This is discussed further in chapter 1.2.5.

Web analytics is also difficult and there are some obstacles and challenges to overcome. Kumar et al. describe some of the hurdles as follows [KO20]: The analysis and interpretation of the data and the goals and measures derived from it are reactive rather than anticipatory, since "predictive modelling applications" for web analytics are not yet on the market.

Big data, that is the volume, variety and velocity of data collected, can be challenging to manage, for example, important insights can be lost or overlooked.

Privacy and the information collected from visitors and customers can be another challenge in terms of inappropriate use of data and GDPR compliance.

1.2.2 Brief History

The history of web analytics can be described as a transformation from an IT domain and a technical log file analysis tool to a sophisticated, polymorphic tool for marketers.

Each time a user requests an HTML file or other resource from a web server, the server makes an entry in a special log file [SKS14]. The first log entries followed the *Common Log Format* (CLF) which provides rudimentary information such as the date, the HTTP status code or the number of bytes transmitted. In 1996, the *Extended Log Format* (ELF) was introduced with more flexibility and information in mind. Thanks to the standardized format of the log files, it was possible to create software that evaluates the log files and presents them to users in a readable form. *GetStats* was one of the first tools which generated statistics and user friendly output for analysts [CP09]. In 1995, Dr. Stephen Turner developed *Analog*, the first free software for analysing log files [ZP15]. Log file analysis will be further discussed in chapter 1.2.4.

What was initially mainly interesting for maintenance and IT staff, who for example answered the question of how many 404s occurred on the server, developed into an interesting website information pool for marketers.

As the available information increased, it became clear that the data could be used for more than just analyzing server behavior. But log file analysis was not enough to provide details about how users interact with the site. Web analytics underwent a transformation from log analysis to user data tracking, analysis, and reporting.

Croll describes the move of analytics from IT to marketing with three steps: [CP09]

1. JavaScript eliminated the need for log files and enabled marketers to maintain and deploy their analytics solutions themselves, making them less dependent on IT.

2. The introduced advertising economy of search engines like Google led to a new focus of analysts on user attraction and conversion rates.

3. New cost models allowed marketers to pay for the analytics service based on website traffic, rather than paying upfront for hardware and software. Analytics spend was thus linked to website traffic and, ideally, revenue.

In 1990 the WWW started and in 1993 one of the first widely used browsers *Mosaic* was launched. At the same time *WebTrends* developed and released one of the first analytics software. A lot more services followed, such as *WebSideStory* in 1996 [CP09] or *Quantified* by Urchin in the same year [CP09].

Page tagging made it possible to collect not only technical data, but also business-relevant information. Visitors and their behaviour were the focus, shaping questions like: How is this user behaviour related to a purchase? If a user buys shoes, will he also buy socks? The development and implementation of cookies enabled the identification of unique users. Not only business-relevant questions were asked and answered, but also studies on performance and usability [CP09]. More details about page tagging can be found in chapter 1.2.4.

In 2003, Edwards, Eisenberg and Sterne founded the *Web Analytics Association* (WAA). The WAA brings together and supports all the players in web analytics such as users, marketers and IT specialists on an international stage. Due to digitalization and its all-encompassing effects, the WAA has renamed itself *Digital Analytics Association* (DAA) in 2012 because the web is not the only area where users leave their digital footprint [SKS14].

As described in the section above, participant and user numbers on the Internet continue to rise and now all Fortune 500 companies operate websites, with web analytics a key marketing tool [KO20].

Some of the most established tools today are *Google Analytics*, *Adobe SiteCatalyst*, *Webtrekk* and *Piwik* [Hei20]. Google Analytics will be further discussed in chapter **??**.

Looking ahead, Zheng et al. identify several trends for web analytics, such as mobile web and application-specific analytics like video, search, learning, or social media analytics [ZP15].

1.2.3 Web Analytics Process

Web analytics can be described as a process in which the main goal is usually to increase sales. The literature cites two main ideas and processes, the first from the Web Analytics Association and the second from industry best practices. They are briefly described in this section.

Both processes have in common that they are aimed at improving the website and thus increasing business revenue.

Key Performance Indicators (KPIs) are the ideal tool for the instrumentation of web analytics and help to identify areas and potential for improvement. They are an integral part and play an important role in any web analytics process as they provide a "in-depth picture of visitor behavior on a site" [Jan09].

KPIs can differ depending on the business in which they operate. For commercial domains, common KPIs are conversion rates, average order or visit value, customer loyalty, bounce rate, etc. [SKS14].

Defining the right KPIs and aligning them with business goals is a critical step in any web analytics process.

WAA Process Guide

The Web Analytics Association offers a web analytics guide that consists of nine steps. They are: [Jan09]

1. Identify key stake holders
2. Define primary goals of website and prioritize them
3. Identify most important site visitors
4. Determine key performance indicators
5. Identify and implement the right solution
6. Use multiple technologies and methods
7. Make improvements iteratively
8. Hire and empower a full-time analyst
9. Establish a process of continuous improvement

Industries Best Practices

On the contrary, Waisberg and Kaushik derive a five-step process from industry best practices with the main goal of improving the website and increasing sales [WK09]:

- Define Goals
- Build KPIs
- Collect Data
- Analyse Data
- Implement Changes
- Repeat last two steps

When comparing the two proposed processes, it becomes clear that both focus on identifying and being aware of the most important business goals. The WAA process is a finer-grained and more practical approach, with Waisberg and Kaushik abstracting the main activities.

1.2.4 Log File Analysis and Page Tagging

There are four main methods of collecting data for web analytics: through log file analysis, JavaScript page tagging, web beacons, and packet sniffing [WK09].

As already mentioned in chapter 1.2.2, the two most important methods of data collection are log file analysis and page tagging. In this section I will briefly describe and compare both mechanisms.

Log File Analysis

As already described, the log file analysis is about gaining knowledge from the log file records of the web server. Log file analysis is considered to be the traditional and original approach to web analytics [Mar15], [ZP15]. As soon as the user types a URL into the browser and presses the enter key, the request arrives at a web server. The server then creates an entry in a log file and sends the requested page or resource back to the client in response [WK09]. The information within the log entry can vary depending on the log format, usually IP, browser, time stamp, time required, transferred bytes, whether a cache hit occurs and the referrer is specified [WK09]. The standardized Common Log Format provides host, ident, authuser, date, request, status, bytes, as can be seen in listing 1.1.[11]

Listing 1.1: CLF

```
127.0.0.1 user-identifier frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

Standard formats of log files and entries enable log file analysis software to process, evaluate and report valuable statistics such as *Analog*, *Webalizer* or *AWStats* to users [ZP15].

Below are a few points that describe the pros and cons of log file analysis. The advantages of log file analysis are ([WK09], [NC11], [SKS14], [ZP15]):

- JavaScript and cookies are not required on the client side
- Maintainer of the website and server owns the data
- Bots and web crawler requests are also logged
- History of data is available
- Log entries are reliable
- The standard format of log files enables easy switching of analysis tools
- The web server also logs failed requests

[11] https://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format [03.06.2021]

- No modification on the web page needed
- Does not demand more bandwidth

Some disadvantages of log file analysis are ([Mar15], [ZP15]):

- Log entries provide mainly technical information, which may not be very useful for user behaviour analysis. Business related metrics such as bounce rates are not available.
- Only direct requests from the client to the web server are logged: Any user interaction in the web browser that does not trigger a request is not logged. Responses from caches and proxies are also not visible in the web server's log file. Only interactions with the web server are logged.

Page Tagging

In a nutshell, page tagging describes the analysis method used to incorporate JavaScript into the website, which collects data and sends it to an analysis server [Mar15]. To do this, JavaScript must be integrated on every page to be analysed [WK09]. Page tagging is the most important method in web analytics today [ZP15].

Croll provides a illustration (figure 1.1) of the page tagging process which is explained below [CP09].

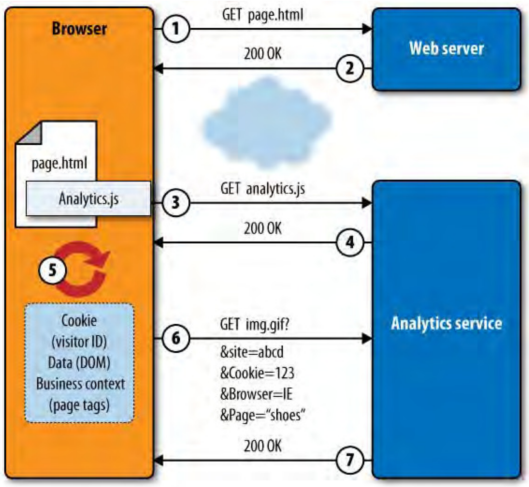


Figure 1.1: Page Tagging

The client (browser) requests a page from the web server (1, 2). Within the HTML file an external JavaScript resource, the analytics code, is linked and received from the analytics server (3, 4). The analytics script tracks and measures the user behaviour and eventually sends the data back to the analytics server (5, 6, 7).

The data collected can also be stored in cookies, which contain data beyond a session and enable the user to be identified, e.g. the next time he visits the page [KO20].

The advantages and disadvantages of page tagging are as follows, starting with the pros ([WK09], [NC11], [Mar15], [SKS14], [ZP15]):

- Every page visit is counted
- The analytics service is outsourced, which includes the storage of the data, but also the data analysis and reporting
- Page tagging is rather easy to implement and favourable when the analyst does not have access to the web server
- Highly customizable: Everything that JavaScript enables to measure, collect, and track is available. This also includes information about the client such as screen size, device used or color depth.
- Ability to track events and actions such as mouse clicks that do not send requests to the web server. This is especially important for single-page or progressive web applications that do not generate requests as often.
- Mechanics of cookies provide identification of unique and repeat visitors
- Real time reporting is possible

Some drawbacks are mainly privacy concerns and the permission to collect data by the user, that the analysis process relies on the use of JavaScript and cookies that can be disabled by the user [Mar15], that every page that is supposed to collect data must contain the analytics script and due to the use of a third party analytics service it is pretty difficult to switch tools [SKS14].

1.2.5 Web Performance

This chapter gives only a brief overview of the various aspects of web performance. Reference is made below to the appropriate sections that cover the topics in more detail.

As already described in chapter 1.1.3, web performance plays a role that cannot be neglected for user satisfaction and business success. The above studies show that increasing website performance also increases sales, or as Google states it: "Performance plays a major role in the success of any online venture".[12]

The MDN Web Docs identify multiple areas of web performance:[13]

- Reducing overall load time

[12] `https://web.dev/why-speed-matters/` [03.06.2021]

[13] `https://developer.mozilla.org/en-US/docs/Learn/Performance/What_is_web_performance` [03.06.2021]

- Making the site usable as soon as possible
- Smoothness and interactivity
- Perceived performance
- Performance measurements

Reducing load time The question of what makes websites slow is covered in chapter X.

Usability and interactivity As I will describe in chapter X, there are several metrics available that attempt to reflect areas of performance such as load time, smoothness, and interactivity, and specific metrics are available as well as for differentiating between technical and user-perceived performance.

Performance perception The perception of performance is generally subjective. As already seen in chapter 1.1.3, there are some quantifiable time intervals that correlate with human psychology regarding the received performance. Table 1.2.5 contains "unofficial rule of thumb" for delay thresholds [Gri13].

Delay	User Perception
0-100 ms	Instant
100-300 ms	Small perceptible delay
300-1000 ms	Machine is working
> 1 s	Likely mental context switch
> 10 s	Task is abandoned

Table 1.2: Rule of thumbs for delay

If one interprets the numbers from the table, one can make the statement that it is desirable to keep loading times below one second. Thresholds for certain performance metrics and the psychological rationale for setting them are discussed in chapter X.

Performance measurements There are several methods of measuring performance. *Synthetic monitoring* is discussed in chapter X. *Real User Monitoring* (RUM) is covered in chapter X.

1.3 Research Question

The e-commerce industry is booming and there are no signs that this trend is reversing; on the other hand. Performance plays an important role in terms of customer satisfaction and how this directly affects business revenue. To better understand e-commerce website visitors, page tagging is widely used and implemented.

Several questions and issues can arise in this area and context, such as: Does page tagging affect the website's performance? Intuitively, it can be said that loading additional JavaScript will reduce the performance of the website, depending on parameters such as the script size and network condition. But are there more unpredictable side effects? Do the various techniques of embedding a tracking script affect the data collected and measured? Will the various tracking scripts supplied interfere with each other?

A hypothesis of this work is that tracking tools slow down the monitored websites, reduce the speed and performance of the website and thus have an unfavourable effect on the user experience.

These questions are to be investigated within the scope of this thesis.

1.3.1 Goal

This thesis has several goals:

The Internet and websites in general are complicated, complex, and tangled. Although basic HTML structures are standardized, each website follows its own form and is unique and sui generis. In order to conduct a controlled experiment and test hypotheses, one goal is to approximate real websites with an artificial, laboratory-generated website that is completely controlled and manipulated by the researcher.

The aim is to create a reliable, but also convincing test environment in order to model and reproduce real behaviour.

Once the test environment is up and running, performance measurement issues need to be addressed. The aim is to measure, collect, visualize and analyse performance data.

As we will see in chapter X, there are many metrics for measuring performance. Another goal of this work is to establish something like a taxonomy of performance metrics.

1.3.2 Chapter Outline

[tbd]

Chapter 1 was about... In Chapter 2 we see, Chatper 3...

2 Metrics and Measurement Methods

[tbd]

- Last chapter...
- This chapter: In this chapter, I will cover measurement methods and discuss common performance metrics.
  - This chapter should cover all relevant terms and definitions within web performance measurement
  - How terms can be structured / taxonomy
  - Ambiguity of definitions
- In the next chapter...

- Technical Background: - Network - Front End: Navigation and CRP
- Metrics
- Measuring Methods
- Compare to competitors - Compare different versions of your app - Metrics should be relevant to your users, site, and business goals - should be collected and measured in a consistent manner - analyzed in a format that can be consumed and understood by non-technical stakeholders

2.1 How Websites are being loaded

- Brief technical introduction - It is important to understand how things work, because Metrics and also how to measure them are derived from those processes

In order to understand web performance metrics and the methods to measure them, it is crucial to have a basic understanding of the technical aspect of loading a website into a browser. This process includes establishing a connection between a client and a server, which will be discussed in section X, and the task of the browser to transform the received data from the server into a readable ready-to-use website, which will be discussed in section X. Always with performance in mind.

It is possible to divide the website loading process into three entities which play a role. In his code talk 2016, Witt identifies three main areas or bottlenecks where bad performance is being produced: In the Frontend, the Backend, and on the network layer.

The Front End is everyhting the user sees on the screen, client, UI, browser, sends requests to a back end, etc. The Back End is the logic, servier, also data base, handles requests and sends responses to a front end Network is what connects clients and servers, FE and BE, infrastructure element composed of routers, cables, wireless connections etc.

- BE is not discussed (server time, data base, etc.) - Section X is about Network - Section X is about Front end: how browser works, crp, - How to optimise websites is not part of this thesis

In this section, i will also say which metrics are describing the underlying process. So this section links directly to the metrics section. But all metrics are collected in the metrics section.

2.1.1 The Network Realm

Starting from hardware, ISP, routers, switches etc and the cables connecting them is part of the network. But also communication protocols such as the Internet protocol suite.

Regarding performance, latency and bandwidth come into mind, and we will see that latency has a bigger impact on performance than bandwidth in section X.

After discussing this issue, i will continue by describing the process or navigation steps which happen once the user enters a URL into the browser, up until he sees pixels on his screen and can use the website.

Latency and Bandwidth

There are two important attributes when discussing network performance: Latency and Bandwidth. The important thing to say here is that Latency is bottleneck for performance, and not bandwidth.

Bandwidth is the "maximum throughput of a logical or physical communication path". In other words, bandwidth describes the amount of data which can be sent in parallel from one node in a network to another.

Physical communication paths are most likely cables such as metal wires or fiber-optic cables, where fiber-optic cables have less signal loss, and lower lifetime maintenance costs. With methods such as wavelength-division multiplexing (WDM), it is possible to transmit up to 70 Tbit/s over a fiber-optic connection.

This high technology stuff is only used in the backbone infrastructure, e.g. for connecting Europe with America. For the end user, bandwidth is much lower, and the average was in late 2015 just 5.1 Mbps

A high bandwidth is useful for bulk or large data transfer such as streaming of video or audio. But for loading a website,or any browser activity that depends on many requests that fetch data from many different locations around the globe, the performance bottleneck is latency.

Latency is "the time from the source sending a packet to the destination receiving it".

Latency is measured in seconds and can be the time spent for one-way, or more common, how long it takes for the transmitted data package for the round-trip time (RTT), from source to destination and back. In other words, latency "describes the amount of delay on a network or Internet connection".

For the very first request when establishing a connection, latency is longer due to protocols such as DNS lookup, TCP and TLS handshakes. Those will be discussed in section X.

To get an idea about how the two aspects, bandwidth and latency, impact web performance, Mike Belshe launched a study. Once setup has a fixed latency and bandwidth is variable, and vice versa. He and compared the performance of the two experiments using the Page Load Time metric. (cf. X for this metric)

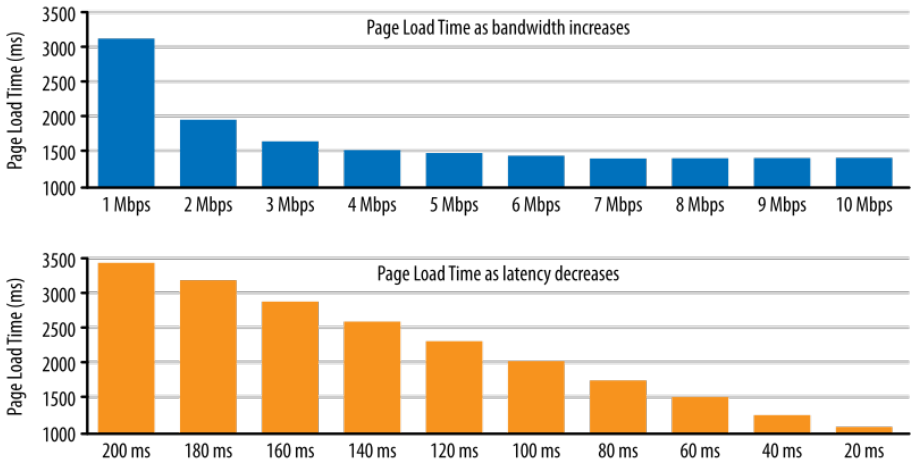


Figure 2.1: Latency vs Bandwidth

We can see that the impact of bandwidth is trivial: if the available bandwidth is doubled, e.g. from 5 to 10 Mbps, there is no change in performance load time. For Latency on the other hand, the picture is different: If the latency can be decreased by half, e.g. from 120 ms to 60 ms, the page load time also sinkt um die hälfte.

Or as Belshe states it, "[reducing] cross-atlantic RTTs from 150ms to 100ms [...] would have a larger effect on the speed of the internet than increasing a user's bandwidth from 3.9Mbps to 10Mbps or even 1Gbps."

This obersvations can be explained with the many short, small connections and requests are made when browsing websites and the contrary underlying structure of the communication protocols, which are "optimized for long-lived connections and bulk data transfers. "

But just simply decreasing the latency is not straightforward: The speed of data transfer is already at a 2/3 of light, but the physical constraint is the limiting factor, e.g. there is a minimum distance between London and New York which can not be further "optimized".

Another aspect of latency is that for wireless connections and therefore mobile devices, latency is even higher, "making networking optimization a critical priority for the mobile

web."

This is due to the infrastructure of mobile nets, latency is high for mobile users. cf. Why are mobile latencies so high? in Grigorik

As latency is a important factor, what happens on the front end is still important. And again for this thesis metrics measuring performance in the front end are the focus.

Before i will discuss what happens in the browser once the website data arrived, i will briefly describe the preceding steps of establishing a connection between the browser (client) and the server.

Navigation Process

I will explain briefly the general navigation process: It begins when the user is submitting a URL in the browser and ends when he received website data.

The main steps can be divided into networking, that is, establishing a connection with DNS etc., backend processing, e.g. data base queries etc., and the rendering in the front end, as seen in image X. The last part of this process is when browser receives finally the HTML / Document. How the browser transfers the HTML into an interactive website is part of the next section.

"To start, it is important to recognize that every HTTP request is composed of a number of separate stages (Figure 10-3): DNS resolution, TCP connection handshake, TLS negotiation (if required), dispatch of the HTTP request, followed by content download."

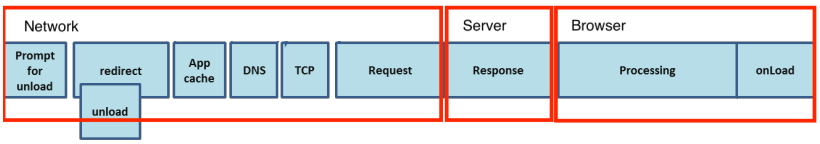


Figure 2.2: Timing Overview

DNS Lookup When the requested resource can not be loaded from the browsers cache, the first step to establish a connection is a DNS Lookup (or DNS Resolution).

This step is about translate URL to IP address. Must be done for each unknown URL,

e.g. when linked images within a website are from different server, for each unique URL DNS Lookup has to be done. The mapping of URL to IP can be cached by browser, which makes repeated views faster.

Avg. time is 20 and 120 ms Can be considered a performance metric, see section X.

TCP Handshake Once a connection between a client and a server is established, the TCP 3-way-Handshake comes into play.

The goal of TCP is to establish a reliable connection within an unreliable network. TCP "guaranteed that all bytes sent will be identical with bytes received and that they will arrive in the same order to the client. " Regarding performance, the handshake adds two more round trips, which is bad for performance as we have seen because of latency.

Many algorithms and techniques to get optimal data transfer and also avoid congestion are existing, such as Slow-Start. Slow-Start is an algorithm that determines the maximum bandwidth that can be used by gradually increasing the amount of data sent. Slow start prevents that the full capacity of the network is being used from the beginning, which in performance terms adds again more round trips and latency.

A performance metric reflecting the time spent for establishing a TCP connection is X, see section X. For a detailed discussion cf "Building Blocks of TCP" in 2013 Grigorik

TLS Negotiation TLS is another protocol which has the goal to establish a secure connection in terms of data encryption. Data transmitted over the network has to be encrypted so that aussenstehende can not read or manipulate the data. For encryption, a cipher to be used needs to be established, which will be shared between client and server during the TLS Negotiation.

TLS again adds more round trips which is bad for performance.

A performance metric reflecting the time spent for a TLS negotiating is blabla in section X.

for a detailed discussion see Transport Layer Security (TLS) in 2013 Grigorik

HTTP Request and Response Now that a secure connection is established, the client fetches the first resources via HTTP GET request. Most often, the server will respond by sending back the index.html file, which then can be used by the browser to build the website.

The time when this first response containing the first byte for building the web site is reflected in the metric TTFB which is discussed in section X.

Usually, many more resources are requested by the browser to complete the build of the web site. As of today, the median value is about 70 requests per web site.

A request is not the same as a connection. Once the connection is established via the above described procedures such as DNS lookup, TCP and TLS handshakes, multiple requests can be transmitted over the same connection. Usually, the number of requests

is much higher than the number of connections to load a website, as the browser persist connections, keep them open for multiple requests. Median connections for a web site today is about 13. Modern browsers like Chrome enable up to six open connections in parallel.

At this point, the browser has received the first data about the web site and he can start with rendering the page. How this exactly happens, is explained in the next section.

2.1.2 Front End: Critical Rendering Path

This section explains what happens after the first bytes of the web sites arrived in the browser. The following processes are typically subsumed under the term *Critical Rendering Path* (CRP). The CRP is the last part of the navigation process as seen in image X.

The CRP is the minimum steps that the browser has to take from the moment it receives the first byte of HTML to the moment that it renders pixels on the screen for the first time.

The rendering is critical as it is the very first render, the first visible content the user will see on the screen. The resources that are needed for the first render of the page delay the first render of the page are considered to be critical. Without the critical resources, the browser can not display content on the screen. An example of a critical resource is the first HTML file the browser receives, as without it, nothing is visible on the screen. Non-critical resources on the other hand will not stop the browser from displaying the first content on the screen.

There are a sequence of steps the browser goes through to render the page. The basic idea is to convert HTML, CSS and JS to actual pixels on the screen.

Image X visualizes the flow of the CRP: Once the HTML is received, the browser starts with parsing the HTML and translate it into the DOM. The content of the CSS files will be parsed to the CSSOM. JavaScript needs to be fetched and executed. Once DOM and CSSOM are available, the Render Tree is being created. When the Render Tree is available, Layout is happening. Finally, pixels can be printed on the screen.

In the following, the individual steps will be discussed in more detail.

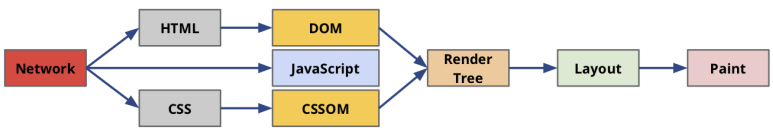


Figure 2.3: Critical Rendering Path

DOM Construction from HTML Once the browser received the first bytes of the HTML file, it starts to parse it into the *Document Object Model* (DOM). The DOM construction is the first step the browser performs when receiving data. The DOM is a tree structure and

internal representation of the HTML for the browser. The general parsing process consists of translating from bytes to characters, to tokens, to nodes and finally to the object model.The specification of the DOM is maintained by the WHATWG living standard.

The DOM tree contains information about the content of the document, but not its style. The styling is defined in the CSS. Once HTML and CSS are transmitted and processed by the browser, the *Render Tree* can be created, which reflects the actual information and its styling the browser can display. Within this context, it is possible to categorise resources into render blocking and non-render blocking. A render blocking resource is a resource that prevents the browser from rendering content to the screen. HTML and CSS are render blocking resources, as the parsing process of those files blocks the browser of displaying the page to the screen.

As soon as the first data packages of HTML arrive at the browser, the parsing process starts. The DOM is created incrementally, this means that the browser can begin to process the HTML before all of its content is transmitted over the network.

Usually, within the HTML, external resources are linked which are necessary for the website to be complete, such as CSS or JavaScript. While parsing the HTML incrementally, eventually a reference to such an external resource will be encountered. How the external resources CSS and JavaScript are being handled by the browser is discussed below.

CSSOM Construction from CSS The CSS resource contains all information about the styling of the page. As with the HTML,CSS is converted from bytes to characters, to tokens, to nodes, and finally to the *CSS Object Model* (CSSOM). CSSOM construction is usually very fast . CSSOM is standardized here

As opposed to the HTML parsing process, CSS can not be translated to the CSSOM incrementally. Cause it the cascading nature of style sheets, which has the potential that the styling rules defined at the top of the file may be overridden by rules defined at the very end of the CSS file. A partial CSSOM is therefore not possible. Hence the browser needs the entire CSS file before he can create the CSSOM.

As soon as the parser encounters a reference to an external style sheet such as

```
<link rel="stylesheet" href="styles.css">
```

it requests the resource and continues with parsing the HTML. CSS is not a parser blocking resource. When the CSS arrived at the browser, the CSSOM construction starts.

While CSSOM creation is not parser blocking, it is render blocking. The browser blocks the page rendering until it received and parsed all of the CSS. Rendering content to the screen is only possible when CSSOM and therefore CSS is available.

Once the DOM and CSSOM are created, they can be merged together into the render tree, which will be layout and painted to the screen. Before I describe this process, I will discuss how JavaScript is being handled.

JavaScript in the CRP JavaScript (JS) resources add functionality and interactivity to a web site. When the browser encounters a script tag such as

```
<script src="myScript.js"></script>
```

it will stop its current task of parsing, fetch immediately the resource and execute its content, and only then proceed with the creation of the DOM. See image X.

JS can manipulate and query the DOM tree and directly change the HTML file. As the HTML file is the input stream for the parser, the parser stops until the JS is downloaded and executed. Hence JS is parser blocking. JS fetching and execution stops the parsing of the HTML and the construction of the DOM. Only after the script finished execution, HTML parsing will continue.

Implicitly, because JS execution blocks DOM creation, and HTML processing itself is render blocking, JS is also render blocking.

The behaviour is the same for an external references JS file and a script directly added within in the HTML.

As JS can also manipulate the styling of the page, its execution is blocked until the CSSOM is available. This means that the execution of the JS is on hold until the CSSOM is ready. To summarize, while JS blocks the parsing of the HTML to the DOM, JS execution itself is blocked by the creation of the CSSOM. CSSOM blocks JS, and JS blocks DOM construction.

Several attributes on the script tag can change the behaviour of the browser. *Async* and *defer* are options to counter the blocking nature of the script tag. They will be discussed now.

With the async (asynchronous) attribute, the browser downloads the JS in the background while continuing with the parsing of the HTML. The parsing is not blocked and the browser can continue with his task. As soon as the JS is downloaded and available, it is parser blocking: the browser stops the parsing and executes the JS.

The order of all the async scripts within the document is not maintained any more. Whenever a script is downloaded and available, it will be executed. It does not matter if an async script was included at the top or bottom of the HTML document.

Like with async, scripts with the defer attribute enable the browser to download the script in parallel while continuing with the parsing of the HTML. Contrary to async, defer scripts will only be executed after the parsing of the page is complete and the DOM tree is fully constructed, and the order of the scripts will be maintained.

Figure 2.4: Scripts

The async and defer attributes not applicable on inline scripts. From the standard, "scripts may specify defer or async, but must not specify either unless the src attribute is present. " As inline scripts do not contain a src attribute, as the source of the script is within the script tags, the async and defer attributes are not applicable.

Building the Render Tree As already described above, HTML and CSS are both render blocking, as they prohibit the rendering of the page. Rendering can happen once the *Render Tree* is available. The render tree is the combination of the DOM and CSSOM and captures all visible content with its styles which will be displayed on the screen. If an element has a CSS property such as `display: none;` it will not occur in the render tree.

The computed render tree is then used to layout the content to the page, which is described in the next paragraph.

Layout In the layout process, the position and size of the nodes from the render tree are calculated. New layout calculations or reflows are triggered as soon as the screen area changes, e.g. by device rotation or window resizing, or on modifications of the DOM and render tree.

Once the layout is resolved, the browser continues with painting pixels on the screen.

Paint Finally, the browser can paint the content on the screen. If some content changes, browsers are optimized to only repaint areas on the screen affected.

2.1.3 Conclusion Technical Background

[tbd]

2.2 Measurement Methods

At this point we have a understanding of how web sites are being loaded into the browser and displayed to the user. Some of the steps are more important for performance than others. In this section, I will describe how to measure the performance of a web site.

Multiple methods exists. The prominent ones are synthetic monitoring and real user monitoring. They will be discussed below.

Some other methods are mentioned in the last section.

After discussing measurement methods, I can finally discuss the metrics we want to measure.

2.2.1 Synthetic Monitoring

The "Synthetic" Aspect in Synthetic Monitoring

As the name already suggests, synthetic monitoring is a measurement method performed in an artificial, laboratory-like, synthetic environment. Test agents simulate real users and are configured to run a browser, load the web site under observation while capturing performance data. Synthetic monitoring does not take real user traffic into account.

Performance data can be captured using common performance APIs as described in section X. Additionally, through video recording and analysis, user centric metrics such as Speed Index can be computed (see section X.)

In synthetic monitoring, many possible configurations and variables of the test agent (client) are under control, such as the location (geography), network conditions, device type, browser version, and so on. Hence, the tester has control over many variables that impact performance.

The controlled environment makes it possible to capture performance data for a specific set up of configurations, such as the test agents location or browser version, which may help to identify issues regarding certain user segments. For example., a test could check the performance of all users using Firefox in macOS in Germany.

Apart from defining the "infrastructure" configuration, the tester can also define artificial user journeys to simulate real user behaviour.

A characteristic of the controlled environment is that measured data and test results are rather consistent with low variability and can therefore provide a performance base line for the web site under observation and facilitate performance tuning.

Synthetic Monitoring is not about Real Users Synthetic monitoring does not capture data of real users as the web sites traffic is generated artificially. Real user behaviour is approximated through simulation by for example predefining user paths. The measured performance data does not necessarily reflect actual real user experience and the tester should not assume that "synthetic results are like real-user metrics".

Capturing the wide variety and diversity of real-world users such as which pages they visits,, the general configuration of the users machine such as the CPU, GPU and memory performance, what data stores the browser cache and which browser version is being used, the screen size, the operating system, and the network connection to name a few, is difficult to represent in synthetic monitoring. The selected test configuration in synthetic monitoring only reflects one special use case and can only approximate what a user with a similar set up may experience. In short, synthetic monitoring test results "are synthetic and therefore not representative for actual user data".

The "Monitoring" Aspect in Synthetic Monitoring

Synthetic monitoring can be automated and used to monitor a systems performance in real time while generating up to date reports for the systems maintainer. Monitoring enables to check the availability of the web site around the globe, to identify performance issues before real users are aware of them. and is in general helpful for continuous "health checks" of the running system.

As any web site can be tested synthetically, it is possible to compare performance data across multiple competitors.

Many synthetic monitoring tools exists (For a list of tools, see for example 2016 Kaur or some online resources here). WebPageTest is one of them and will be discussed in section X.

Coming back to e-commerce context, as discussed in section X. a online shops performance correlates with the revenue. Synthetic monitoring allows to capture performance metrics independent of real user behaviour. Real user behaviour is not measured in synthetic monitoring. As only real users are capable of generating revenue, synthetic monitoring can not identify correlations between user satisfaction and performance (as described in section X.).

In order to do this, RUM is needed. Real-User Monitoring (RUM) enables to capture data of each individual real user . RUM will be discussed next.

2.2.2 Real-User Monitoring

As the name suggests, Real-User Monitoring (RUM) is about collecting and measuring data from real users visiting the web site. As opposed to synthetic monitoring, where web site traffic is generated artificially and performance experience from real users is only approximated, RUM data relies on real user traffic and captures data directly from each users browser. RUM measures the performance as experienced by the users.

The Page Tagging Technique in RUM

As already described in section X. Page Tagging is technique to instrument the users browser in order to collect data and report it back to an analytics server. RUM is based on page tagging, in terms of that it relies on a JS code snippet (tracking or code) which will be loaded into the users browser. Once this JS code is loaded and executed in the browser, it collects data and sends it back to an analytics service. If the user blocks JS, or the script can not be downloaded due to other reasons, RUM will not work. Once the data arrives at an analytics service, is has to be stored and an interface for the analyst has to be provided in order that he can query the data and get insights, for example by providing a dashboard.

As RUM relies on the JS code, the very first opportunity to measure data is when this JS code has been downloaded and executed. Anything what happens before this step is

not visible for the tracking script. Meenan states that approximately 20 % of the loading process lies outside of the RUM measurement scope and "getting a reliable start time for measurements from the real world has been the biggest barrier to using RUM".

Another facet of RUM is that ideally the measuring of data has as little as possible impact on the web sites rendering process and that network capacity should not be occupied by RUM scripts and block resources of the CRP (see section X.) If RUM as a page tagging technique is slowing down the web site under observation is one research question of this thesis. The evaluation of the controlled experiment tackling this questions states that RUM...., as discussed in great detail in section X.

RUM is independent of the users set up or environment and collects data for all active users: Regardless of the device, the browser,the network condition or the geographical location of the user, as long as the measurement script is downloaded into the users browser, RUM collects data.Hence, RUM data represents each individual user experience.

Through the diversity of users and the unique environment of each user, RUM data tends to be more diverse and heterogeneous than data collected by synthetic monitoring.

Measure Real User Behaviour

As discussed in section X, a web sites performance and user satisfaction are directly correlated. A critical part of RUM is to not only capture performance metrics, but also measure user behaviour, for example how the user interacts with the web site and where he clicks. In an e-commerce context, user behaviour questions of interest are for example if a new campaign changes user behaviour as expected or where users leave the check out process.

RUM enables the combination of performance metrics with user behaviour and business metrics (see section X.) and can answer questions such as if and how the performance of the web site affects the user behaviour, for example if users buy more or less depending on the web sites speed.Thus RUM is not only important for understanding user behaviour, but also for optimizing the web site and to increase revenue. With techniques such as cookies (see section X.), it is also possible to track user behaviour not only for one page load but over a series of web site visits, leading to even more detailed insights about the visitor.

Multiple RUM tools and JS libraries exists, such as Boomerang by Akamai. [1] The main player Google Analytics will be discussed in greater detail in section X.

- APIs to measure real users (will be discussed in metrics chapter): - Navigation Timing - Resource TIming - User TIming - The combination of Navigation, Resource, and User timing APIs provides all the necessary tools to instrument and conduct real-user performance measurement for every web application

[1] `https://github.com/akamai/boomerang` [23.06.2021]

- Real-User Monitoring leverages browser APIs to collect data specific to each end-user transaction

- Collected information includes various timers to capture network and rendering performance - To also capture details about the user's device and browser or on the referring website over which the user arrived, the user agent string and other data artifacts are collected along with the values obtained from the Navigation and Performance Timing APIs

conversion rates etc.

Chrome User Experience Report (CrUX)

The Chrome User Experience Report (CrUX) is a RUM method implemented by Google which collects real user data of Chrome users. As soon as the Chrome user gives his consent, data collection can start and does not need any more set up.

The CrUX only captures data from Chrome users and is therefore not an exhaustive sample of the web users population, as data by users browsing the web with for example Firefox or Safari is not collected.

The collected data is available via Googles PageSpeed Insights, the CrUX Dashboard or the BigQuery and CrUX API.

2.2.3 Log Files and Surveys

Other RUM methods are surveys and log file analysis.

Log analysis: - concerned with technical performance in the backend - generates insights from the data that is already available from the application server, proxy, or content delivery network (CDN) - TTFB - Server logs typically only reflect the time it took to send the first response, not until the client actually started receiving it - Analyzing logs from application servers, CDNs, or proxies is a reasonable first step to discover potential bottlenecks, but it only covers the server side and does not provide any information on client-side processing in general or rendering performance in particular

Surveys: - directly ask the users for their opinions - direct approach to understanding whether or not users are satisfied with website performance: Just ask them for their opinion - online surveys or by offering some sort of price or chance to win a competition in exchange for the users opinions. - Other options include actual interviews or monitoring users in a lab setting to find out how they react while surfing on the website. - they only cover a small sample of the user base and therefore may be subject to a certain selection bias - user perception can be highly inaccurate - Getting reliable info from user surveys is therefore no trivial task - some forms of surveys (e.g. lab experiments) can be relatively expensive to conduct in comparison to some of the fully automated alternatives for collecting information - are great for collecting qualitative feedback on the user experience, but are not suitable for gathering quantitative measurements

2.2.4 Measurement Methods Conclusion

Multiple methods to measure performance of a web site exist. Two main complementary techniques exist, Synthetic Monitoring and Real-User Monitoring.

Synthetic Monitoring measures a web sites performance in a controlled environment using test agents and is especially useful to find a performance base line of the web site and for continuous monitoring and health checks. Performance as end users may experience it can only be approximated it is not captured by synthetic monitoring.

RUM collects data from each user visiting the web site and reports it back to an analytics server. RUM is especially useful when combining multiple metrics together such as performance and business metrics in order to analyse user behaviour. On the other hand, when no user is visiting the web site, RUM is not collecting any data.

Other performance measurement methods such as CrUX and surveys provide browser specific or quality data and complement the analysts tool box.

Metrics are critical in order to map performance to some sort of value or number. Through metrics, performance is quantifiable to some extent and therefore comparable.

As already described, measurement methods such as synthetic monitoring or RUM can measure metrics such as "performance" or "business" metrics. What are does metrics exactly? What do they reflect? How can they be measured?

Those questions will be addressed in the next section.

2.3 Metrics

Introduction: What are metrics, Different kind of metrics General, web analytics (Business) Metrics and then take a look at performance metrics Page weight, Performance, UX and custom Metrics Metrics Taxonomy

reflect in numbers numbers can be compared: over time, after change, with competitor, etc.

- Compare to competitors - Compare different versions of your app

- use these metrics to analyze Web traffic and improve a Website to meet better the expectations of the site’s traffic

"Metrics serve as the basic information for analyzing Web traffic and helps in improving a Website to meet its goals"

- Measure what is important for you tailored, best would be to have custom metrics, see section X.

- Metrics should be relevant to your users, site, and business goals

- each business has its own definitions of success - p.15 "whether your business benefited in some way from their visits." - search engine vs e-commerce website example

- should be collected and measured in a consistent manner - analyzed in a format that can be consumed and understood by non-technical stakeholders

Good metrics should be: Uncomplex, Relevant, Timely, Instantly Useful

"metrics as well as experiments have to realistically reflect possible performance improvements for actual users"

2.3.1 Business or "Non-Performance", general web analytics Metrics

Unlimited metrics, also when thinking about custom metrics. The main question here is how to categorize those metrics. Best is: numerical values, Relations, non-numerical values

Possible Categorizations of Web Analytics Metrics

Semantic categorization Orthogonal: "number" categorization: Count, ratio, ...

- User Engagement: - Session Length - Time on site - First user interaction - Bounce rate

- Business KPIs: - Cart size - Transactions - Conversion Rate - Revenue

- QA Metadata: - Page views and sessions - Browser distribution - JavaScript errors - Caching insights

-> Semantic, about the meaning of metrics -

- Argument with Customer Life Cycle: Retention - Awareness - Conversion

Measuring Reach: - Overall Traffic Volumes - Number of Visits - Number of New Visitors - Percentage of New Visitors -...

Measuring Acquisition: - Percent New Visitors - Average Number of Visits per Visitor - Average Number of Page Views per Visit - Average Pages Viewed per Visitor -...

Measuring Conversion: - Conversion Rates - Abandonment Rates - ...

Measuring Retention: - Number of Returning Visitors - ...

-> customer-related, customer-centric - Marketing

- WAA - 3 Types of web analytics metrics: - Counts: single number - Rations: Count divided by count - KPIs: count or ration infused by business strategy

- Counts and ratios are categories for number types - KPIs bring the dimension of business, give semantic meaning to numbers

4 categories: site usage, referrers, site content analysis, quality assurance 8 fundamental metrics Site usage: Demographics and System Statistics Internal Search Information Visit Length Visitor Type Referrers: Referrering URL and Keyword Analysis Site content analysis: Top Pages Visitor Path Quality assurance: Errors

All metrics answer one of the Big 4 Question: - What did they do? - How did they do it? - Why did they do it? - Could they do it?

- Very user centric

- Metrics for describing visits: entry page, landing page, exit page, visit duration, referrer, ctr - For describing Visitors: new visitors, returning visitor, repeat visitor, visits per visitor, recency, frequency - For describing Visitor engagement: page exit ratio, bounce rate, page views per visitor - Conversion metrics: conversion, conversion rate

Types: Anzahl, Relations, Werte

Counts: - Absolut value - visitors - sales total etc

Relations: - Put aboslute metrics in relation - As Relation or Percentage - Most important type - Page Views per Visitor

Values: - Referrer - Search term

I will use this categorization to list some metrics.

Metrics Categories for E-Shop: - Attraction - Acquisition - Retention - Umsatzleistung - Warenleistung - Ergebnisleitung

- also customer life cycle

We have seen that multiple categorizations of metrics exists. I will now use possible categorization: Counts, Ratios, Values For each category, i will list some metrics and explain them or even put it in table. This list is not complete.

After that, i will take a look at metrics which are specifically for performance.

Incomplete Web Analyitcs Metrics Collection

use counts, rations and non-numerical values

Counts describe how many Can be measured Over time: last hour, month, from to, etc. Averages are possible Compare with Segmentation

Combining numerical values As Ratio or Percentage

Can not be reflected as number

Counts	
Hits	blablabla
Visits	blabla
Sessions	blabla
Session Length	blabla
Page Views	blabla
Single Page Visits	blabla
New / Unique / Repeat / Return Visitors	blabla
Time on Page	blabla
Time between Visits	...
JS Errors	...
Cart Size	...
Conversions	...
...	...
Ratios	
Conversion Rate	...
Bounce Rate	...
Abandonment Rate	...
Page Views per Visitor	...
New Visitors Percentage	...
Visits per Visitor	...

Page Exit Ratio	...
Click Through Rate (CTR)	...
...	...
Non-Numerical Values	
Demographics	...
Referrer	...
...	...

Table 2.1: Your caption here

List is not complete. Now we move to performance specific metrics. As performance is directly connected to time, most of those performance metrics are measuring a timestamp. Some are also displaying a score. This will be discussed.

2.3.2 Performance Metrics

As with general web analytics metrics, there are multiple ways to categorize performance metrics.

- network-centric, browser-centric, and user-centric metrics

For me most intuitive is

Metrics regarding Page Weight and size of the web site Timing Metrics, how long each step from the CRP took UX Metrics, try to reflect how users experience performance, visual metrics Custom Metrics

Page Weight

- naive approach to measure web site, make it quantifiable - the size of the web site - Page weight has an impact on performance - Page consists of resources - More resources and bytes means more downloads and waiting time

- Generally for all resources / assets: Bytes and Requests - Total Size: Number of bytes sent over the network, which may be compressed

- Total Kilobytes - Total Requests
- HTML Bytes - HTML Requests
- CSS Bytes - CSS Requests
- JavaScript Bytes - JavaScript Requests
- Font Bytes - Font Requests
- Image Bytes - Image Requests
- Video Bytes - Video Requests
- Other Bytes - Other Requests

In approach section, i used this data to come up with a median web site. In this section are also some numbers available.

Performance Timings Metrics

Web Standards - in 1994, TimBL founded the World Wide Web Consortium (W3C), an organization that brings together representatives from many different technology companies to work together on the creation of web technology specifications

Web Standards: - technologies we use to build web sites - standards exist as long technical documents called specifications - detail exactly how the technology should work - to be used by software engineers to implement these technologies (usually in web browsers) - For example, the HTML Living Standard describes exactly how HTML (all the HTML elements, and their associated APIs, and other surrounding technologies) should be implemented. - Others WHATWG (who maintain the living standards for the HTML language), ECMA

W3C What is W3C

W3C publishes documents that define Web technologies. These documents follow a process designed to promote consensus, fairness, public accountability, and quality. At the end of this process, W3C publishes Recommendations, which are considered Web standards.

e.g.DOM, HTML, SVG, ...

- Recommendation track: Publication of the First Public Working Draft. Publication of zero or more revised Working Drafts. Publication of one or more Candidate Recommendations. Publication of a Proposed Recommendation. Publication as a W3C Recommendation.

- From Idea to concrete API which browser maintainers implement

- Working Draft (WD) - Candidate Recommendation (CR) - Proposed Recommendation - W3C Recommendation (REC)

Web Performance Working Group Multiple groups exists which are responsible for a specific topic. Performance working group is responsible for performance.

- In 2010, the browser vendors got together under the W3C banner and formed the Web Performance Working Group to standardize the interfaces and work toward improving the state of Web-performance measurement and APIs in the browsers - Clock issue -> High Resolution Time specification

- WebPerf Working Group (founded 2010) - "Provide methods to measure aspects of application performance of user agent features and APIs"

- Working Groups push new recommendations and invent them Web Performance Working Group is responsible: - Measurement - Scheduling - Adaptation -> Important is Measurement

- APIs used to create waterfall charts - Waterfall we can see in more detail in WPT section ?

- Any time you've seen a waterfall chart, you're looking at a visual representation of the data these APIs provide

Normative Specifications: - High Resolution Time - Performance Timeline - Resource Timing - Navigation Timing - User Timing - Page Visibility - Long Task - Paint Timing

Server Timing:

- attempt to address the concern about lacking insight into how or why certain stages of the request-response cycle have taken as much time as they have

others, not important for metrics: - Beacon - Cooperative Scheduling of Background Tasks - Device Memory - Reporting - Network Error Logging

Now i will take a closer look at Navigation, Resource and User Timing and which metrics are derived from there. But start with time.

High Resolution Time "Level 2" is older but already recommendation from date Newest spec from date is working draft

Is the basis for other specs

- Old: Date from ECAMScript: Date object as a time value representing time in milliseconds since 01 January, 1970 UTC - DOMTimeStamp is defined similarly - definitions of time are subject to both clock skew and adjustment of the system clock - The value of time may not always be monotonically increasing and subsequent values may either decrease or remain the same. - resolve issues by providing monotonically increasing time values with sub-millisecond resolution

- Clock issue -> High Resolution Time specification

- Non-Decreasing - Earlier: Date object -> Its relative to start of navigation instead of 1. January 1970 UTC as JS Date Object

- sub-millisecond resolution - provide a monotonic, uniformly increasing timestamp suitable for interval measurements

- performance.now() is a measurement of floating point milliseconds since that particular page started to load -> the performance.timing.navigationStart timeStamp to be specific - number stays relative to the page because you'll be comparing two or more measurements against eachother

- type definition - used to store a time value in milliseconds, measured relative from the time origin, shared monotonic clock, or a time value that represents a duration between two DOMHighResTimeStamps

- type - double - store a time value in milliseconds, accurate to 5 microseconds - describe a discrete point in time or a time interval

- now() - timeOrigin

Navigation Timing - access the complete timing information for navigation of a document - Navigation is about how user agents convert the requested HTML, CSS, and

JavaScript into rendered pixels, which is one of the most critical steps for users to navigate a document

- complete client-side latency measurements

- Navigation timings are metrics measuring a browser's document navigation events - measures the main document's timings - read only - supported in all browsers - measuring the performance of the main page, generally the HTML file via which all the other assets are requested

- navigation timing measures the main document's timings - The real benefit of Navigation Timing is that it exposes a lot of previously inaccessible data, such as DNS and TCP connect times, with high precision (microsecond timestamps), via a standardized performance.timing object in each browser. - Hence, the data gathering process is very simple: load the page, grab the timing object from the user's browser, and beacon it back to your analytics servers! - By capturing this data, we can observe real-world performance of our applications as seen by real users, on real hardware, and across a wide variety of different networks. - Navigation Timing provides performance timers for root documents only

- The largest benefit of navigation timing is that it exposes a lot of timings that lead up to the HTML loading –> This is this famous image - In addition to providing a good start time, it exposes information about any redirects, DNS lookup times, time to connect to the server, and how long it takes the Web server to respond to the request, for every user and for every page the user visits - The measurement points are exposed to the DOM (Document Object Model) through the performance object and make it trivial to calculate load times (or arbitrary intervals, really) from JavaScript.

- Level 1 PerformanceTiming and PerformanceNavigation Interface, Both are collected in Performance Interface, Performance Interface is accessible through window.performance - Level 1: time is measured in milliseconds since midnight of January 1, 1970 (UTC). - Level 2: Navigation Timing 2 specification allows access to timing information related to navigation using sub-milliseconds resolution instead

- Level 1 kept for backwards compatibility - Level 2 is working draft from march 2021 and replaces first version of Navigation Timing - See summary of improvements for differences between level 1 and 2 - Level 2 is Builds on top of Resource Timing 2 and uses high resolution time 2

- level 2 specification defines the PerformanceNavigationTiming interface - we will see later in X which attributes are available

PerformanceNavigationTiming Interface: - extends PerformanceEntry Interface from performance timeline. see attributes section for details - extends PerformanceResourceTiming Interface from resource timing

- window.performance.getEntriesByType("navigation")

Resource Timing - Level 1 is candidate recommendation from march 2017

- Level 2 is working draft from april 2021 - access the complete timing information for resources in a document. - Navigation Timing 2 extends this specification to provide additional timing information associated with a navigation. - mechanisms to provide complete client-side latency measurements within applications

- set of attributes for each loaded resource - access a set of critical network timing attributes for each resource on the page - resources usually refer to HTML documents, XHR objects, links (such as a stylesheet) or SVG elements

- resource timing measures timing for individual resources, the assets called in by the main page, and any assets that those resources request - Many of the measurements are similar to navigation timing

- retrieval and analysis of detailed network timing data regarding the loading of an application's resources - determine, for example, the length of time it takes to fetch a specific resource like request, svg, image, script

- Resource Timing provides similar performance data for each resource on the page, allowing us to gather the full performance profile of the page. - resource timing provides the times for all the assets or resources called in by that main document and the resources' requested resources.

- exposes timing information about every network request the browser had to make to load a page and what triggered each request (whether stylesheet, script, or image)

- PerformanceResourceTiming interface

- performance.getEntriesByType("resource"): array of Resource Timing Objects for each requested resource - The PerformanceResourceTiming interface extends the PerformanceEntry interface in the Performance Timeline - Each of these timestamps is in microseconds, which are provided by the window.performance.now() method in the High Resolution Time specification.

Attributes from Navigation and Resource Timing As both specs are tightly coupled, e.g. resource timings is in the middle of navigation timing, i will cover them both here together. I will go through all exposed attributes and explain them. Then i will show one example of how a vendor is implementing this.

Then i will show which metrics are directly calculated from those attributes.

- We can see that this is the same as explained in CRP

exposed as DOMHighResTimeStamp

from PerformanceEntry Interface: name entryType startTime duration

Navigation Timing	
origin.. (startTime, navigationStart)	...
unloadEventStart	...
unloadEventEnd	...
Resource Timing	

redirectStart	...
redirectEnd	...
fetchStart	...
domainLookupStart	...
domainLookupEnd	...
connectStart	...
connectEnd	...
secureConnectionStart	...
requestStart	...
responseStart	...
responseEnd	...
Navigation Timing continued	
domInteractive	...
domContentLoadedEventStart	...
domContentLoadedEventEnd	...
domComplete	...
loadEventStart	Same as window.onload
loadEventEnd	...
Extended from PerformanceEntry Interface (cite navigation timing 2 spec)	
name	Address of the current document
entryType	"navigation"
startTime	Time value of 0
duration	loadEventStart - startTime
Navigation Timings Outside of Timeline	
type	...
redirectCount	...
Resource Timings Outside of Timeline	
initiatorType	...
nextHopProtocol	...
workerStart	...
transferSize	...
encodedBodySize	...
decodedBodySize	...

Table 2.2: Navigation and Resource Timing Level 2 Attributes

- good explanations of the attributes

Requests and responses: - fetchStart marks when the browser starts to fetch a resource. This is distinct from a request in that it doesn't mark when the browser makes a network request for a resource, but rather when it begins checking caches (e.g., HTTP and service worker caches) to see if a network request is even necessary. - workerStart marks when a request is being fetched from a service worker within a fetch event handler (if appli-

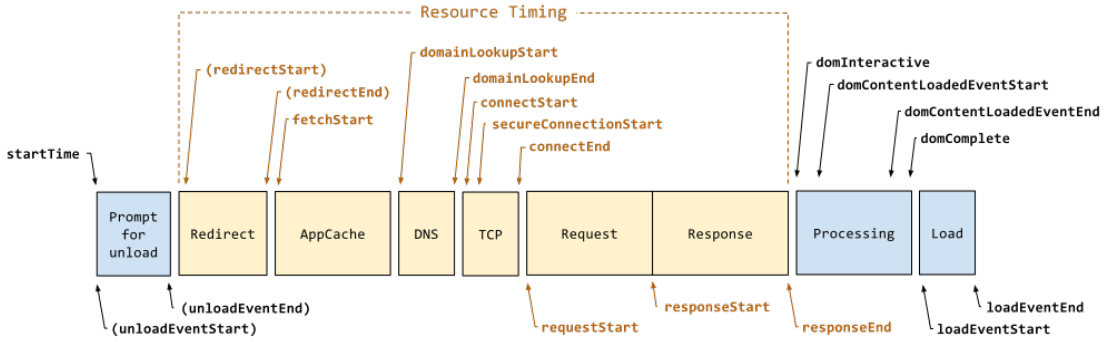


Figure 2.5: Timestamps from Navigation and Resource Timing

cable). This will be always be 0 if a service worker isn't installed for the current page. - requestStart is when the browser issues the network request. - responseStart is when the first byte of the response arrives. - responseEnd is when the last byte of the response arrives.

lets see how chromium is capturing domContentLoaded

Directly from those timestamps many metrics are calculated. Sometimes different vendors give a different name to the same metric. For many metrics there are no official definitions and the values diverge depending on the implementation. I will show how those metrics are being calculated.

- navigationStart is not in level 2 - navigationStart is the same as startTime in level 2 - Difference is that navigationStart is a EPOCH time stamp. The timings have to be calculated as differences to this timestamp - navigationStart just starts with 0, which makes all the other timestamps easier to read - many still use level 1 and navigationStart

Time to First Byte (TTFB)	responseStart - navigationStart
Page Load Time (PLT)	loadEventStart - navigationStart
DNS Lookup Time	domainLookupEnd - domainLookupStart
TCP Handshake / Server Connect Time	connectEnd - connectStart
TLS Handshake Time	requestStart - secureConnectionStart
Request / Server Response Time	responseStart - requestStart
DOM Content Loaded	domContentLoadedEventEnd - domContentLoadedEventStart
Page Download Time	responseEnd - responseStart
Latency	responseStart – fetchStart
DOM Interactive	
DOM Complete	
Redirect	

Table 2.3: Metrics

-

Some other specs here. Check if they are also used for calculating metrics.

User Timing User Timing: - obtain the high precision performance characteristics of an

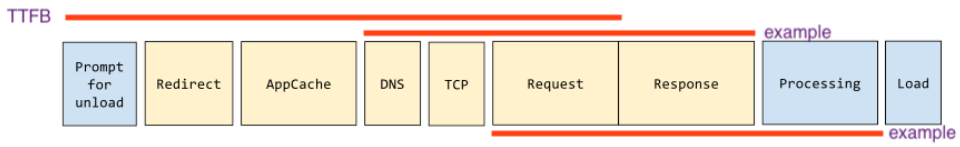


Figure 2.6: Timestamps from Navigation and Resource Timing

- User Timing API: - Not a browser API - Measure events and durations - Keep track of marks (timestamps) - Mark: timestamp - Measure: delta between two timestamps - performance.mark - Performance.measure -> Puts data in Performance Timeline / DevTools

-> Create your own timestamps and measures. -> Also visible in WebPageTest

Performance Timeline Access all different data from other APIs.

Performance Timeline: - Performance Timeline APIs was introduced in 2011 - unified interface to obtain various performance metrics - The Performance Timeline API uses PerformanceEntry.entryType to describe the type of the interface represented by this PerformanceEntry object, which represents performance measurements

- PerformanceEntry object represents performance measurements - PerformanceEntry object: - name - entryType - startTime - duration

Performance Timeline API Performance Entry API

- PerformanceEntry object encapsulates a single performance metric that is part of the performance timeline - A performance entry can be directly created by making a performance mark or measure

Always of type: - PerformanceMark - PerformanceMeasure - PerformanceFrameTiming - PerformanceNavigationTiming - PerformanceResourceTiming - PerformancePaintTiming

Properties: - name - entryType - startTime - duration

Version 1 and 2

- store and retrieve performance metric data - does not cover individual performance metric interfaces

- defines PerformanceEntry.entryType to describes the type of the interface represented by this PerformanceEntry object - PerformanceEntry object represents performance measurements

Types values: - mark (User Timing) - measure (User Timing) - navigation (Navigation Timing) - resource (Resource Timing) - longtask

PerformanceObserver

Performance Observer API

- Performance Timeline API: - window.performance.getEntries() - getEntriesByType(), getEntriesByName() - Capturing different performance metrics - Browser APIs can add data to this timeline

- performance.getEntries(): List of events that happened on that page

=> Window.performance returns Performance object => Performance object exposes: - High Resolution Time API - Navigation Timing API - Resource Timing API - Performance Timeline API - User Timing API

Performance API includes: - Performance Timeline API, the Navigation Timing API, the User Timing API, and the Resource Timing API.

- provides access to performance-related information for the current page - part of the High Resolution Time API, but is enhanced by the Performance Timeline API, the Navigation Timing API, the User Timing API, and the Resource Timing API - can be obtained by calling the window.performance

Performance API -> Includes Performance Timeline API, the Navigation Timing API, the User Timing API, and the Resource Timing API.

- Performance: base interface for these standards - methods and properties are extended by different standards

PerformanceEntry API: - provides for marking and measuring times along the navigation and resource loading process - You can also create marks - performance.getEntriesByType('navigation -> get navigation timings - performance.getEntriesByType('resource') -> get resource timings

- The PerformanceEntry object encapsulates a single performance metric that is part of the performance timeline

Conclusion Provide here a table with all important recommendations and the metrics they expose.

UX / User Perceived, Visual Performance Metrics

Web Vitals a set of browser-based metrics for capturing user-perceived performance. Google heralds the Web Vitals as the gold standard for website performance across a number of services (e.g. PageSpeed Insights, Search Console, TestMySite) and publishes them in the Chrome User Experience Report (CrUX) [7] database. The core metrics will further be used for ranking search results starting in June [6] and even non-AMP pages will be admitted to the Top Stories [4] feature in Search on mobile — given they exhibit top-of-the-line performance according to the Core Web Vitals. Website performance according to the Web Vitals will therefore be critically important for SEO in the upcoming years.

Key questions: Is is usable, is it delightful, ... Types of metrics important metrics custom metrics

Core Web Vitals: First Input Delay, Cumulative Layout Shift, Largest Contentful Paint First Paint, First Contentful Paint: Is it happening? PerformanceObserver First Meaningful Paint, Hero Element: Is it useful? Time To Interactive: Is it usable? Use Polyfill Long Tasks: Is it delightful? PerformanceObserver Total Blocking Time Time To First Byte

Core Web Vitals Most important metrics, Apply to all websites, Measures real user experience, Measurement support for Lab and Field, Concise and clear LCP: Progressive loading. FCP may become a core web vital FID: Interactivity during load CLS: Visual stability Future goals: Better support for Single Page Apps, Input responsiveness, Scrolling and animations Areas of user experience beyond performance: Security, Privacy, Accessibility

Introduction, what is it How to measure How to improve

Google had some issues with capturing LCP in chrome, see twitter Because of this they needed to release a new chrome version CruX also has 2 different lcps now (?) use this story to show that this stuff is far from trivial

Introduction, what is it How to measure How to improve

Introduction, what is it How to measure How to improve

Other UX Metrics Visually complete ?

Speed Index

Custom Metrics

- First, it is important to understand that no single number will answer that question. Even if you have defined exactly what you are trying to measure on your Web site, performance will vary widely across your user base and across the different pages on your site - Nothing beats application-specific knowledge and measurements

- own / user specific metrics, e.g. time to first tweet for twitter

- "there is no one single metric that holds true for every application, which means that we must carefully define custom metrics in each case" - "Custom and application-specific metrics are the key to establishing a sound performance strategy"

2.3.3 The Metrics Taxonomy, Map

DNS resolution / DNS lookup time = GAs Avg. Domain Lookup Time (sec)

Connecting / TCP Handshake time

TLS Handshake time

Waiting / Server Response Time ??

Receiving / Download Time ??

onload Event:

DOMContentLoaded Event

Page Load Time: - Has been the de facto metric of the web performance world - Increasingly insufficient performance benchmark: we are no longer building pages, we are building dynamic and interactive web applications

Start Render - tells you how many seconds it took for the browser to begin rendering content

Time to First Byte:

- First byte that the browser receives - It's a good indicator of how quickly the backend of your site is able to process and send back content

TTFB describes how long the requesting party has to wait from sending the first request packet until receiving the first response data packet

First Paint, Time to First Paint:

- can tell you the first point in time when the user sees something other than a blank white screen - doesn't necessarily mean the user sees anything useful

First Contentful Paint:

First Meaningful Paint:

The First Meaningful Paint is another user-centric metric and represents the point in time at which the largest visual change takes place; the underlying assumption here obviously is that the biggest visual change is relevant to the user, for example because a hero image or a navigation bar appear

Largest Contentful Paint:

- approximation of the First Meaningful Paint that can be captured in the browser

Speed Index:

The Speed Index revolves around visual completeness and represents the average time it takes for website elements to become visible. It works well for websites with a static layout, but is unreliable for websites with moving elements like carousels or videos. Computing the Speed Index for this kind of website requires a custom timer or event to demarcate the point in time at which visual completion is reached

- best effort for representing the user experience in a single number

Above-the-fold time - point in time when the last visual change is made to the visible part of the page - available only in lab environments where the visual progress of a page loading can be recorded

Time to Interactive:

3 Related Work

- Last chapter...
- This chapter: Describe shortly all sections from this chapter
- In the next chapter...
- This chapter should list research which covers and explores questions relevant for this thesis, such as:
  - Metrics: New metrics, meaning of metrics, difficulties of defining metrics, etc.
  - Overview, evaluation and comparison of measurement tools and methods
  - If available: Impact of RUM on performance

3.1 WebPageTest

- Overview
- Configuration
- Private Instances

3.1.1 Overview

- What is it
- Why to use it, Who uses it, how to use it
- Waterfall and Grades
- See in performance tab for details about grades and optimization techniques

3.1.2 Configuration

- Caching, repeat view
- Traffic shaping
- e.g. capture devtools timeline

3.1.3 Private Instances

- Architecture
- AWS
- Docker localhost
- Bulk tests

- A waterfall chart such as Figure 2-2 shows you how much time it takes to request the contents of a page, such as CSS, images, or HTML, and how much time it takes to download this content before displaying it in a browser. - ...

3.1.4 Metrics

- Metrics Categories:
  - High Level Metrics:
    - Document Complete
    - Fully Loaded
    - Load Time
    - First Byte
    - Start Render
    - Requests
    - Bytes In (Page Size)
  - Page-level Metrics:
    - Technical Page Metrics:
      - -> APIs, GA Site Speed Metrics
      - TTFB
      - loadTime
      - docTime
      - ...
    - Visual Metrics:
      - SpeedIndex
      - firstPaint
      - firstContentfulPaint
      - firstMeaningfulPaint
      - ...

        * Javascript and CPU timings
        * Page Information
        * Browser State
        * Lighthouse Summary Metrics
        * Optimization Checks/Grades
        * Instrumented Metrics
        * Test Information
        * Misc
    - Request-level metrics:
        * Request Details
        * Request Timings
        * Request Stats
        * Headers
        * Protocol Information
        * Javascript/CPU details
        * Optimization Checks
        * Misc
- Optimization Grades:
    - Keep-alive Enabled
    - Compress Text
    - Compress Images
    - Cache Static Content
    - Use of CDN
- First View and Repeat View

Name	Description
Successful Tests	Amount of tests who completed successfully

Document Complete	The time from the initial request until the browser fires load event. Also known as the document complete time. This is the time at which the Document Object Model (DOM) has been created and all images have been downloaded and displayed. For most traditional web pages, the load time is a suitable metric for representing how long a user must wait until the page becomes usable. This is the default performance metric on WebPageTest. Also known as Load Time (?). Around this time, the page's script is hard at work in the load-event handler firing off more requests for secondary content. The incomplete nature of this metric is why Fully Loaded was added to the table of metrics from the previous section. window.onload (?). The point where the browser onLoad event fires. The equivalent Navigation Timing event is loadEventStart. Document Complete Time: Amount of time that has elapsed from the initial page request until the browser fires the load event. This is the time at which the Document Object Model (DOM) has been created and all images have been downloaded and displayed.
Fully Loaded	The time from the initial request until WebPageTest determines that the page has finished loading content. The page might have waited for the load event to defer loading secondary content. The time it takes to load the secondary content is accounted for in the Fully Loaded Time. The time (in ms) the page took to be fully loaded — e.g., 2 seconds of no network activity after Document Complete. This will usually include any activity that is triggered by javascript after the main page loads. The point after onLoad where network activity has stopped for 2 seconds. Specific to WebPageTest and not provided by Performance API. Fully loaded waits for 2 seconds of no network activity (and no outstanding requests) after onLoad and then calls it done (only measures to the last activity, doesn't include the 2 seconds of silence in the measurement). Fully Loaded is a measure based on the network activity and is the point after onload when there was no activity for 2 seconds.
First Byte	Time until the server responds with the first byte of the response.
Start Render	Time until the browser paints content to the screen. The time for the browser to display the first pixel of content (paint) on the screen. Time until the browser paints content to the screen. WebPageTest's own metric, determined by programmatically watching for visual changes to the page. Same as First Render?

Bytes In (Doc)	Total size of the Document Complete Requests' response bodies in bytes.
Requests (Doc)	Number of HTTP requests before the load event, not including the initial request.
Load Event Start	Time in ms since navigation started until window.onload event was triggered (from W3C Navigation Timing).
Speed Index	See Speed Index
Last Visual Change	Time in ms until the last visual changed occured. Last change is a completely visual measurement and is the last point in the test when something visually changed on the screen. It could be something as simple as an animated gif or ad even that didn't really cause much CPU work but changed some pixels on the screen. It is only captured when video is recorded because it depends on the video capture to measure it.
Visually Complete	Time in ms when page was visually completed. Is measured from a video capture of the viewport loading and is the point when the visible part of the page first reached 100% "completeness" compared to the end state of the test.

Table 3.1: Your caption here

3.2 Google Analytics

- Custom metrics with Google Web Vitals as example
- Show how to include GA script (analytics.js, gtag, Tag Manager, etc.)
- Show some real life examples how script code is included into page, e.g. from Amazon, Otto etc

Real User Measurement (RUM) with Google Analytics

3.2.1 The Tracking Script

- Show multiple code examples
- Explain whats going on: script tag, create script element etc.
- Maybe also show Hotjar example to see that they are similar

- this async pattern is used so that all browsers will load it async - we can just use async attribute for newer browsers...

3.2.2 Site Speed Metrics

Show with analytics.js that it is indeed those navigation timing api calculations.

Ec = function (a)...

GA does not really provide any UX metrics! The site speed metrics are all from navigation timing api which are measurements from the browser.

GA Site Speed Metrics (description from `https://support.google.com/analytics/answer/2383341?hl=en&ref_topic=1282106`)

`https://stackoverflow.com/questions/18972615/how-do-the-metrics-of-google-anal`

Name	Description
Page Load Sample	The number of pageviews that were sampled to calculate the average page-load time.
Speed Metrics Sample	The sample set (or count) of pageviews used to calculate the averages of site speed metrics. This metric is used in all site speed average calculations, including avgDomainLookupTime, avgPageDownloadTime, avgRedirectionTime, avgServerConnectionTime, and avgServerResponseTime.
DOM Latency Metrics Sample	Sample set (or count) of pageviews used to calculate the averages for site speed DOM metrics. This metric is used to calculate ga:avgDomContentLoadedTime and ga:avgDomInteractiveTime.
Page Load Time (sec)	The average amount of time (in seconds) it takes that page to load, from initiation of the pageview (e.g., click on a page link) to load completion in the browser.
Domain Lookup Time (sec)	The average amount of time spent in DNS lookup for the page.
Page Download Time (sec)	The time to download your page.
Redirection Time (sec)	The time spent in redirection before fetching the page. If there are no redirects, the value for this metric is expected to be 0.
Server Connection Time (sec)	The time needed for the user to connect to your server.
Server Response Time (sec)	The time for your server to respond to a user request, including the network time from the user's location to your server.
Document Interactive Time (sec)	The average time (in seconds) that the browser takes to parse the document (DOMInteractive), including the network time from the user's location to your server. At this time, the user can interact with the Document Object Model even though it is not fully loaded.
Document Content Loaded Time (sec)	The average time (in seconds) that the browser takes to parse the document and execute deferred and parser-inserted scripts (DOMContentLoaded), including the network time from the user's location to your server. Parsing of the document is finished, the Document Object Model is ready, but referenced style sheets, images, and subframes may not be finished loading. This event is often the starting point for javascript framework execution, e.g., JQuery's onready() callback, etc.

3.2.3 Comparison GA and WPT Metrics

- We can show that above relations are true with experiments
- Load test page on a specific day only once and save timings exposed by perfor-

Navigation Timing API	WPT	GA
loadEventStart - navigationStart	Document Complete, Load Event Start	pageLoadTime
domainLookupEnd - domainLookupStart	DNS lookup, dns_ms	domainLookupTime
connectEnd - connectStart	connect_ms	serverConnectionTime
responseStart - requestStart	..	serverResponseTime
responseEnd - responseStart	..	pageDownloadTime
fetchStart - navigationStart	..	redirectionTime
domInteractive - navigationStart	..	domInteractiveTime
domContentLoadedEventStart - navigationStart	domContentLoadedEventStart	domContentLoadedTime

mance.timing object (from console)

- Calculate differences corresponding to the table
- Get GA data for that day and save it

3.3 Research

- Research exists about topics like: ....
- Here i will provide a list of in my eyes relevant papers, summaries them and discuss why this is important for my research

3.3.1 some title for first category

2014 Singal I. - Describes history of web analytics and tools - Provides definitions and taxonomy for metrics - Describes log file vs page tagging - Describes KPIs

II. - Lit. overview for KPIs and Web Metrics - Lit. overview for "Trust" - Lit. overview for "Fuzzy" -> What are does categories?

III. - Some other literature worth mentioning

IV. - Describes 8 open challenges for researchers

2015 Bekavac - Two parts: - 1: Some general overview of web analytics, tools and metrics, KPIs etc - 2: Empirical study about employees satisfaction of used web analytics tools

1: - 9 web business models and 5 common goals - Hypothesis: Web analytics tools track and improve a user's satisfaction with web-based business models. - Web anayltics defintion. Log files vs Site Tagging - Web Analytics process - Tools: 5 categories, Process of selecting tool, Table with features of different tools - Web metrics categories, Table with business models and their KPIs

2: - Which tools are used for which purpose / Activity - Users satisfaction

3.3.2 Research about Tools

Kaushik 2007 - Provides 3 questions which help to choose web analytics tools

2011 Nakatani - Gives some arguments why web analytics is important for business - Provides different categorizations for web analytics tools - Gives pros and cons of log file analysis and page tagging - Provides tool selection method based on AHP (Analytic Hierarchy Process)

"Web analytics tools collect click-stream data, track users navigation paths, process and present the data as meaningful information. - Categorizations: 1: By 4 different data collection methods 2: SaaS vs in-house 3: mobile vs non-mobile 4: Time lag

2016 Kaur - free vs paid - real time vs long term - hosted vs in-house - data portability - free / open source tools - proprietary tools - Service Hosted Software - GA most popular one

3.3.3 Research about Metrics

- Dont know:

4 Approach

- Last chapter...
- This chapter: Describe shortly all sections from this chapter
- In the next chapter...
- In this chapter the practical work should be documented and explained
- Elaboration of how the practical work could help answer the research question
- Discussion of real-life setup and how experiments approach it

4.1 Empirical Research Methods

- Overview of methods
- reproduceability etc.
- validity
- Justification why following approaches are conducted as controlled experiments
- Change something: Delete this item again

4.1.1 Controlled Experiment

- Short overview about controlled experiments in computer science
- Design: Show test setup image: Independent and dependent variables
- Hypothesis testing

4.1.2 Test Setup

- What is test object (website)
- What are dependent variables: Performance metrics
- What are independent variables: Specific changes in test object (see next chapter)
- Kohavi 2016: Sample size, collect right metrics, track right users, randomization unit

Variable	Values
Position	top-head, bottom-head, bottom-body
Attribute	no attribute, async, defer
Other Script	false, true

Measure effects: Dependent Variables

- Performance metrics from Lab and Field, see terms and definitions
- But also quality of RUM data. Because we could have a nice performance but RUM will be of bad quality.

Test object / HTML Template

- Depending on different approaches / Ideas (see next chapter), template looks different
- But general structure stays the same and independent variables can be defined
- Here we show different independent variables and variants

Lab and Field

- I want to collect Lab and field data for dependent variables for comparison
- This setup is a special case because lab bots (e.g. WPT) simulate at the same time real users for RUM data

4.1.3 Independent Variables within template

- IV 1 POSITION: Position of included analyitcs script. Values: top-head, bottom-head, bottom-body
- IV 2 ATTRIBUTE: Attribute of included analyitcs script: no-attribute, async, defer
- IV 3 OTHER SCRIPT: Other tracking script included
- Other IVs not included but worth mentioning

I will compare the values from one independent variable only. Therefore, when comparing the values of one independent variable, i need to set a default value for the other independent variables. The default values are:

Position: top-head Attribute: no attribute Other Script: false

Listing 4.1: Position 1

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Google Analytics -->
        <script></script>
        <!-- End Google Analytics -->

        <title>
        <meta>
        <link>
        <script>
    </head>

    <body>
        ...
    </body>
</html>
```

Listing 4.2: Position 2

```
<!DOCTYPE html>
<html>
    <head>
        <title>
        <meta>
        <link>
        <script>

        <!-- Google Analytics -->
        <script></script>
        <!-- End Google Analytics -->
    </head>

    <body>
        ...
    </body>
</html>
```

Listing 4.3: Position 3

```
<!DOCTYPE html>
<html>
    <head>
        <title>
        <meta>
        <link>
        <script>

    </head>

    <body>
        ...
        <!-- Google Analytics -->
        <script></script>
        <!-- End Google Analytics -->
    </body>
</html>
```

Other IVs not included but worth mentioning

- More or less infinite number of independent variables
- Again the big and important fact that each website is different

it looks like attribute on inline script is ignored... "Inline JavaScript script tags ignore the defer or async attribute" so those variables will not make any sense? Do i see this in the data? As i can see how the GA is included, it will create a async attribute to fetch the analytics.js

4.2 Test Object: HTML Template / Test website ideas

- Several ideas are proposed
- Each idea has pro and contra: each idea should be discussed of its usefulness, advantages and disadvantages

4.2.1 WordPress

- Show usage of WordPress with some statistics: Why is it so verbreitet
- Explain Plugin system
- Explain Setup on localhost with wocommerce and GA plugin
- Elaborate why this idea was not used

4.2.2 Plain / Skeletal Website

- Idea: Lab environment to have control over all and see effects of changing independent variables
- Problem: Too far away from reality
- Use this as the simplest test possible, not even POC (POC is http archive site)

4.2.3 HTTP Archive inspired website

- Idea: Get correct page weight
- POC: Show that changing independent variables X affect result

- Chart with changes on time: apps are growing

4.2.4 Mirroring a complete e-commerce website

- Which website / shop to clone? Show some statistics about biggest e-commerce websites in germany

Listing 4.4: Attribute 1

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Google Analytics -->
        <script></script>
        <!-- End Google Analytics -->

        <title>
        <meta>
        <link>
        <script>
    </head>

    <body>
        ...
    </body>
</html>
```

Listing 4.5: Attribute 2

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Google Analytics -->
        <script async></script>
        <!-- End Google Analytics -->

        <title>
        <meta>
        <link>
        <script>
    </head>

    <body>
        ...
    </body>
</html>
```

Listing 4.6: Attribute 3

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Google Analytics -->
        <script defer></script>
        <!-- End Google Analytics -->

        <title>
        <meta>
        <link>
        <script>
    </head>

    <body>
        ...
    </body>
</html>
```

Listing 4.7: Other Script 1

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Google Analytics -->
        <script></script>
        <!-- End Google Analytics -->









        <title>
        <meta>
        <link>
        <script>
    </head>

    <body>
        ...
    </body>
</html>
```

Listing 4.8: Other Script 2

```
<!DOCTYPE html>
<html>
    <head>
        <!-- Google Analytics -->
        <script></script>
        <!-- End Google Analytics -->

        <!-- Other Script-->
        <script></script>
        <!-- End Other Script -->

        <title>
        <meta>
        <link>
        <script>
    </head>

    <body>
        ...
    </body>
</html>
```

Otto Re-write this to otto start page clone chapter

Manual adjustments: - Move everything to test folder because top domain is /otto

What did not work (mostly 404s): - user-set-consent-id-cookie: Cookie with name consentId is not set, user-set-consent-id-cookie returns therefore 404 - subscribeToNewsletterSnippetContent: Change path did not work... - amount.json: Not found, also wl_miniWishlistAmount in local storage does not created - a_info: Mock a_info response json does not work...

- footer - userTiming

WPT RV is returning empty csv when 404s are encountered. Therefore i mock the missing ressources so that WPT can run bulk tests successfully.

- mock image sprite_all_1ba408b2.png

- create empty file called user-set-consent-id-cookie

- change path for subscribeToNewsletterSnippetContent: This will remove the cookie banner... but then WPT works

- Idea: Close to reality as possible
- Problems when mirroring a website
- Elaborate why this idea of mirroring complete website was not used
- I used mock of start page of otto, which works fine
- Compare original otto website with mock

Comparison to original webpage

- Remove GA again from mock, so that mock and original are as similar as possible
- Run the same lab test on both pages: WPT and mabye lighthouse
- Compare both results and explain differences

- Setup: Run WPT on mock and on original website - WPT config: - Browser: Chrome - Number of test runs: 3 - FV and RV - Capture Video - Capture DevTools Timeline - Bulk testing: 100x

Diagrams with FV and RV for both cases:

Technical: - First Byte - Bytes In (Doc) - Requests (Doc)

VIsual: - Document Complete - Speed Index

Problem with Repeat View - Problem with RV, Caching: Otto sets request headers to cache-control: no-cache which means that RV basically downloads all resources again. The mock is hosted on Github, where the cache-control header is set to ... It is not possible to change the github request headers. We can modify the http request headers via html, but this is not a clean solution. Therefore I use a different e-commerce website which does not shut down caching so that the RV results are more similar.

Ideally I would host the mock website on a similar infrastructure as the original site with the same webserver configuration. This is for a masters thesis not feasible.

Zalando Idea: It looks like zalando page does not has that many cache-control headers, therefore it may be easier to clone so that RVs are more similar.

Comparison Diagrams with fixed traffic shaping:

4.3 Test Runs

- This section covers all conducted test runs
- Explain test configuration: how many runs, dependent and independent variables, etc.

4.3.1 WPT Configurations

General Settings

Table 4.1: Test Runs [Sch99]

Configuration Setting	Options	GA
Test Location	Test Location	.
Browser	Firefox, Chrome	.
Connection	LAN	.
Number of Tests to Run	1 to 9	..
Repeat View	First View and Repeat View, First View Only	.
Capture Video	True or False	..
Keep Test Private	True or False	...
Label	Any String	...
Advanced Tab	...	...
Chromium Tab	...	...
Auth, Script, Block, SPOF, Custom Tabs	...	...
Bulk Testing Tab	List of URLs	...

Explanations First View: "First View refers to the cold cache setup in which nothing is served locally" Repeat View: "Repeat View refers to the warm cache containing everything instantiated by the first view" (2016 Using WPT p. 62)

Capture Video: ...

For one test, we have actually six times that the website gets loaded and tested. For e.g. 500 URLs in the bulk test list, we have a total of $500 \times 6 = 3000$ page hits.

Configuration 1

Configuration 2 Emulate Mobile Browser

Figure 4.1: Number of tests to run: 3, First View and Repeat View

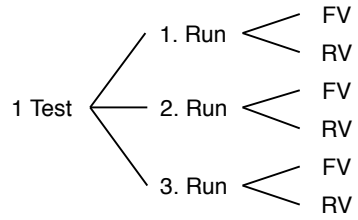


Table 4.2: Configuration 1

Configuration Setting	Option
Test Location	Test Location
Browser	Chrome
Connection	LAN
Desktop Browser Dimensions	default (1366x768)
Number of Tests to Run	1
Repeat View	First View and Repeat View
Capture Video	True
Keep Test Private	False
Label	none
Advanced Tab	Nothing selected
Chromium Tab	Capture Dev Tools Timeline selected
Auth, Script, Block, SPOF, Custom Tabs	Nothing
Bulk Testing Tab	Test URL x times according to test plan

Traffic Shaping

- Important to have stable and realistic network condition
- Chromes tool is not the best for this
- Private WPT Instance docker on mac does not allow traffic shaping functionality from WPT
- I use Network Link Conditioner from Apple to slow down the whole machine. See in same blogpost that Patrick highly recommends this
- WPT also slows down their whole machines
- IN general internet connection is very unstable. If i run network link conditionier with e.g. DSL each speedtest gives different results. And other test platforms such as fast.com gives also different result.
- as long as internet connection is stable along all tests, it should not make a big difference because i compare the different variants. Therefore internet connection will fall out of the equation

- i will use the durchschnitt in germany which seems to be 40 mbit per second. or actually i use LTE profile from network conditioner which is 50 mbit per second

4.3.2 Test Object (Website) Variations

as described before, i will compare the values within one independent variable. This is needed in order to compare the impact of the different values within one IV. For example, i want to measure if there is a difference in performance between the different script attributes. To measure this, i set the default values for the other IVs and vary the values for the IV attribute

Positions: 1: Top of head element 2: just before closing head element 3: just before closing body element

Variants

Variant	Position	Attribute	Other Scripts
Variant P1	top-head	none	no
Variant P2	bottom-head	none	no
Variant P3	bottom-body	none	no
Variant A1	top-head	none	no
Variant A2	top-head	async	no
Variant A3	top-head	defer	no
Variant OS1	top-head	none	no
Variant OS2	top-head	none	yes

Table 4.3: Your caption here

I will not compare variants which are not from the same subgroup, e.g. Variant A2 will not be compared to Variant OS2. Because the first row of the variants table also includes the default values for Attribute and Other Scripts, VP1 is equal to VA1 and VO1.

With the defined IVs and variants, I can create the test objects, that is the index.html files with the corresponding setup. Because its easier to differentiate i will create for the three equal variants nevertheless own index files.

For each test variant, I will create a concrete test artefact, which is a modified index.html. This index.html needs to be uploaded to the webserver before starting with the tests.

All variants will have the same name which is index.html. This is the default file which will be delivered by the webserver when accessing root path of webpage.

Variants to measure: ————————

- Original website - Mock without GA - Position 1 - Position 2 - Position 3 - Attribute 1

- Attribute 2 - Attribute 3 - Other Script True - Other Script False

4.3.3 Test Plan. Generate the data

The Google Analytics code is more or less fixed and there are no configurations. It would be possible to change config of script, e.g. change sample rate, track other metrics etc. But it is not possible to change default tracking behaviour (?)

How the script is included into the file should reflected withing Website Variations

I will use only one WPT Configuration for all tests. Other WPT config can be used in future work, e.g. emulate mobile device.

Table 4.4: Test Runs [Sch99]

Variant	Traffic Shaping	Runs	Date
V-P1	DSL	500	2021-05-07
V-P2	DSL	500	2021-05-07
V-P3	DSL	500	2021-05-07
V-A1	DSL	500	2021-05-07
V-A2	DSL	500	2021-05-07
V-A3	DSL	500	2021-05-07
V-OS1	DSL	500	2021-05-07
V-OS1	DSL	500	2021-05-07

Pre-step: Compare Mock website with and without GA included The comparison between mock and original is part of chapter Test Object

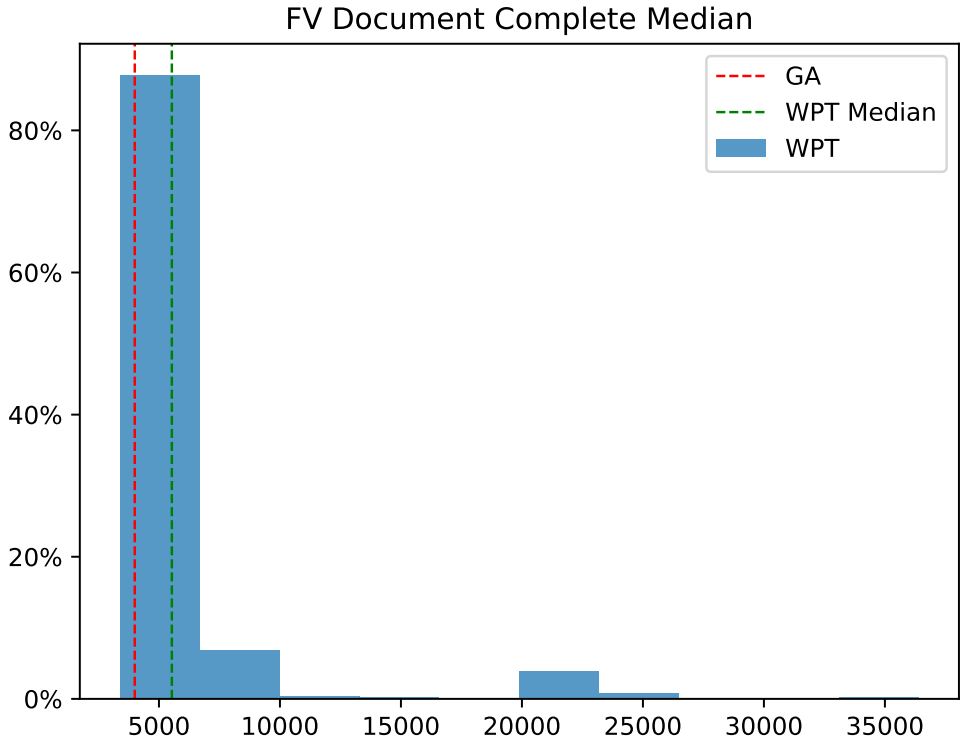
4.3.4 Test Protocol

- Deploy variant of index.html by pushing to GitHub
- Start Network Link Conditioner with specified config on local machine
- Test internet speed with speedtest-cli
- Start local WPT server and agent
- Configure WPT according to specified setup and add list of urls to bulk test interface
- Run test
- When finished, download summary csv file
- On GA helper site, fetch and download data for the current day

4.3.5 Tool support for diagrams and data analysis

- python
- Matplotlib
- seaborn library

5 Evaluation



- Last chapter...
- This chapter: Describe shortly all sections from this chapter
- In the next chapter...

5.1 Test Results

5.1.1 Metrics for Evaluation

Page Weight: Measured by WPT: - bytes - bytes uncompressed - Requests

Technical Timings / API: Measured by WPT and GA: - page load time - domain lookup time - page download time - redirection time - server connection time - server response time - Dom interactive time - Dom content loaded time

Visual Metrics / Web Vitals: Measured by WPT TODO Measure also with GA / own script ??: - CLS - FCP - FMP - LCP - SI

- Visually complete ? - Time to Interactive ? Is this the same as DOM interactive time ? Core Web Vital FID ?? -> Can not be measured without real users... From WPT bulk section. Also include this somewhere for comparison ?: - Filmstrips ? - Waterfall ? - Visual Progress ? - Layout Shifts ?

5.1.2 Original vs Mock Plain

5.1.3 Mock Plain vs Position 1 (which is default position of GA: Check this again!)

TODO rename this like with GA true false ?

5.1.4 Position 1 vs 2 vs 3

5.1.5 Attribute 1 vs 2 vs 3

5.1.6 Other script True vs False

5.2 General

- For each attempt, describe: Threats to validity, generalizability

generalizability: meine Daten zeige nur für Chrome, MacBook, diese Geschwindigkeit etc. Und auch nur für diese Test-Website Die Schwierigkeit der Generalisierbarkeit ist eines der grössten Probleme bei dieser Fragestellung

5.3 Plain / Skeletal Website

- Information gained from this experiment
- Limitations and questions which can not be answered with this approach

5.4 Mirroring

5.5 HTTP Archive inspired website

- Information gained from this experiment
- Meaning and interpretation of the collected data
- Limitations and questions which can not be answered with this approach

5.6 WebPageTest Bulk Tests

- Bulk testing is a feature for private instances only
- Misuse this feature to test the same website X times

5.6.1 Bulk Test Overview: Description of test result page

- Each test has Test ID: YYMMDD_random_random
- Test results after bulk test available under `http://localhost:4000/result/{testID}/`
- For each test run, following data is available:
  - Link to test results: Test result page as same as for single test run
  - Median load time (First view)
  - Median load time (Repeat view)
  - Median Speed Index (First View)
  - Raw page data (file: [TestID_summary.csv]
  - Raw object data (file: [TestID_details.csv])
  - Http archive (.har) (file: json)
- Average First View Load Time
- Average Repeat View Load Time
- Combined Raw: Page Data (file: [TestID_summary.csv])
- Combined Raw: Object Data (file: [TestID_details.csv]). For 100 test runs, this file is appr. 20 MB, 24432 rows, 76 columns.
- Aggregate Statistics (file: [TestID_aggregate.csv])

5.6.2 Summary File for one Test

- Contains 6 rows: 3 test runs: for each test runs 1x first view and 1x repeat view
- Rows 1, 3, 5 contain FV, rows 2, 4, 6 contain data for RV

5.6.3 Aggregate Statistics File

- Contains aggregated data from bulk test
- One row for each test run: For 100 URLs in bulk test will be 100 rows in csv
- Each metric is available with Median, Average, Standard Deviation, Min, Max

- Metrics are available once from FV and once for Repeat View
- Metrics:
  - Successful Tests
  - Document Complete
  - Fully Loaded
  - First Byte
  - Start Render
  - Bytes In (Doc)
  - Requests (Doc)
  - Load Event Start
  - Speed Index
  - Last Visual Change
  - Visually Complete
- => For metric details, see Terms and Definitions

5.6.4 Compare Section

WPT has a feature to compare multiple tests. Accessible under compare URL: `http://localhost:4000/video/compare.php?tests={TestID},{TestID},...`

The compare page contains:

- Film strip
- Waterfall diagram
- Visual Progress diagram
- Timings diagram:
  - Visually Complete (First View Visually Complete Median)
  - Last Visual Change
  - Load Time (onload)
  - ...
- Cumulative Layout Shift diagram
- Requests diagram
- Bytes diagram
- Visually complete

- Last Visual Change
- Load Time (onload)
- Load Time (Fully Loaded)
- DOM Content Loaded
- Speed Index
- Time to First Byte
- Time to Title
- Time to Start Render
- CPU Busy Time
- 85% Visually Complete
- 90% Visually Complete
- 95% Visually Complete
- 99% Visually Complete
- First Contentful Paint
- First Meaningful Paint
- Largest Contenful Paint
- Cumulative Layout Shift
- html Requests
- html Bytes
- js Requests
- js Bytes
- css Requests
- css Bytes
- image Requests
- image Bytes
- flash Requests
- flash Bytes

- font Requests
- font Bytes
- video Requests
- video Bytes
- other Requests
- other Bytes

5.7 Internal, external validity

- At this point, i have the data collected and can analyse it
- The quality and quantity of the data needs to be discussed
- Quality: There are chances that some data are malformed, e.g. because internet connection was bad, etc.
- Quantity: Is the amount of data sufficient to make the evaluation generalisable

6 Future Work

- Last chapter...
- This chapter: Describe shortly all sections from this chapter
- In the next chapter...

6.1 Limitations of this thesis

- Discussion of unobserved topics
- Discussion of possible next steps

6.2 Other measurement tools and metrics

- List of tools and metrics worth investigating

6.2.1 Google Analytics 4

6.3 Speed Kit

6.4 PWAs, AMPs, Service Workers, Caching, HTTP2 etc.

- Overview of other web technologies and how they could be relevant for further research

7 Conclusion

- Last chapter...
- This chapter: Describe shortly all sections from this chapter
- Scope and contribution of this thesis
- Short summary of each chapter:
  - Problem statement and why it is worth to examine research question
  - Terms and definitions
  - (Related work)
  - Approach and evaluation of practical work
  - Future work

- Several topics wurden bearbeitet in this thesis, such like mocking a website for testing purposes, literature review, metrics taxonomy, and the main part which is an experiment

8 Appendix

8.1 WebPageTest Bulk Tests

8.1.1 Single Test Raw page data

WPT Metrics from summary file

Name	Description
minify_total	Total bytes of minifiable text static assets.
responses_200	The number of responses with HTTP status code of 200, OK.
testStartOffset	...
bytesOut	The total bytes sent from the browser to other servers.
gzip_savings	Total bytes of compressed responses.
requestsFull	...
start_epoch	...
connections	The number of connections used.
base_page_cdn	The CDN provider for the base page.
bytesOutDoc	Same as bytesOut but only includes bytes until the Document Complete event. Usually when all the page content has loaded (window.onload).
result	Test result code.
final_base_page_request_id	...
basePageSSLTime	...
docTime	Same as loadTime.
domContentLoadedEventEnd	Time in ms since navigation started until document DOMContentLoaded event finished.
image_savings	Total bytes of compressed images.
requestsDoc	The number of requests until Document Complete event.
firstMeaningfulPaint	...
score_cookies	WebPageTest performance review score for not using cookies on static assets.
firstPaint	RUM First Paint Time, the time in ms when browser first painted something on screen. It's calculated on the client for browsers that implement this method.
score_cdn	WebPageTest performance review score for using CDN for all static assets.
optimization_checked	Whether or not optmizations were checked.

score_minify	WebPageTest performance review score for minifying text static assets.
gzip_total	Total bytes of compressible responses.
responses_404	The number of responses with HTTP status code of 404, not found.
loadTime	The total time taken to load the page (window.onload) in ms.
URL	The tested page URL.
score_combine	WebPageTest performance review score for bundling JavaScript and/or CSS assets.
firstContentfulPaint	...
image_total	Total bytes of images.
score_etags	WebPageTest performance review score for disabling *ETag*s.
loadEventStart	Time in ms since navigation started until window.onload event was triggered (from W3C Navigation Timing).
minify_savings	Total bytes of minified text static assets.
score_progressive_jpeg	WebPageTest performance review score for using progressive JPEG.
domInteractive	...
score_gzip	WebPageTest performance review score for using gzip compression for transferring compressable responses.
score_compress	WebPageTest performance review score for compressing images.
domContentLoadedEventStart	Time in ms since navigation started until document DOMContentLoaded event was triggered (from W3C Navigation Timing).
final_url	...
bytesInDoc	Same as bytestIn but only includes bytes until Document Complete event.
firstImagePaint	...
score_keep-alive	WebPageTest performance review score for using persistent connections.
loadEventEnd	Time in ms since navigation started until window.onload event finished.
cached	0 for first view or 1 for repeat view.
score_cache	WebPageTest performance review score for leveraging browser caching of static assets.
responses_other	The number of responses with HTTPS status code different from 200 or 404.
main_frame	...
fullyLoaded	The time (in ms) the page took to be fully loaded — e.g., 2 seconds of no network activity after Document Complete. This will usually include any activity that is triggered by javascript after the main page loads.
requests	List of details of all requests on tested page.

final_base_page_request	...
TTFB	Time to first byte, which is the duration in ms from when the user first made the HTTP request to the very first byte of the page being received by the browser.
bytesIn	The amount of data that browser had to download in order to load the page. It is also commonly referred to as the page size.
osPlatform	...
test_run_time_ms	...
tester	The ID of tester that performed the page test.
browser_version	The browser version.
document_origin	...
document_URL	...
date	Time and date (number of seconds since Epoch) when test was complete.
PerformancePaintTiming.first-paint	...
osVersion	...
domElements	The total number of DOM elements.
browserVersion	The browser version.
fullyLoadedCPUms	CPU busy time in ms until page was fully loaded.
browser_name	The browser name.
PerformancePaintTiming.first-contentful-paint	...
base_page_cname	...
eventName	...
os_version	...
base_page_dns_server	...
fullyLoadedCPUpct	Average CPU utilization up until page is fully loaded.
domComplete	...
base_page_ip_ptr	...
document_hostname	...
lastVisualChange	Time in ms until the last visual changed occured.
visualComplete	Time in ms when page was visually completed.
render	The first point in time (in ms) that something was displayed to the screen. Before that user was staring at a blank page. This does not necessarily mean the user saw the page content — it could just be something as simple as a background color — but it is the first indication of something happening for the user.
SpeedIndex	The SpeedIndex score.
visualComplete85	Time in ms when page was visually completed 85%.
visualComplete90	Time in ms when page was visually completed 90%.
visualComplete95	Time in ms when page was visually completed 95%.
visualComplete99	Time in ms when page was visually completed 99%.
LargestContentfulPaintType	...
LargestContentfulPaintNodeType	...

chromeUserTiming.navigationStart	...
chromeUserTiming.fetchStart	...
chromeUserTiming.responseEnd	...
chromeUserTiming.domLoading	...
chromeUserTiming.markAsMainFrame	...
chromeUserTiming.domInteractive	...
chromeUserTiming.domContentLoadedEventStart	...
chromeUserTiming.domContentLoadedEventEnd	...
chromeUserTiming.firstPaint	...
chromeUserTiming.firstContentfulPaint	...
chromeUserTiming.firstImagePaint	...
chromeUserTiming.firstMeaningfulPaint	...
chromeUserTiming.firstMeaningfulPaintCandidate	...
chromeUserTiming.domComplete	...
chromeUserTiming.loadEventStart	...
chromeUserTiming.loadEventEnd	...
chromeUserTiming.LargestContentfulPaint	...
chromeUserTiming.LargestTextPaint	...
chromeUserTiming.CumulativeLayoutShift	...
run	The run number.
step	...
effectiveBps	Bytes per seconds, i.e.: total of bytes in / total time to load the page.
effectiveBpsDoc	Same as effectiveBps but until Document Complete event.
domTime	The total time in ms until a given DOM element (specified via domelement parameter when running a test) was found on the page.
aft	Above the Fold Time (no longer supported). The time taken to load everything in the viewport above the fold.
titleTime	Total time in ms until page title was set on browser.
domLoading	...
server_rtt	...
smallImageCount	...
bigImageCount	...
maybeCaptcha	...
bytes.html	...
requests.html	...
bytesUncompressed.html	...
bytes.js	...
requests.js	...
bytesUncompressed.js	...
bytes.css	...
requests.css	...
bytesUncompressed.css	...
bytes.image	...
requests.image	...

bytesUncompressed.image	...
bytes.flash	...
requests.flash	...
bytesUncompressed.flash	...
bytes.font	...
requests.font	...
bytesUncompressed.font	...
bytes.video	...
requests.video	...
bytesUncompressed.video	...
bytes.other	...
requests.other	...
bytesUncompressed.other	...
id	...
chromeUserTiming.InteractiveTime	...

Table 8.1: Your caption here

8.1.2 Single Test Raw object data

8.1.3 Single Test Http archive (.har)

8.1.4 Combined Test Raw page data

8.1.5 Combined Test Raw object data

8.1.6 Combined Test Aggregate data

Bibliography

[Abe] ABERDEENGROUP: *The Performance of Web Applications. Customers Are Won or Lost in One Second.* https://info.headspin.io/hubfs/Analyst%20Reports/5136-RR-performance-web-application.pdf, Abruf: 2021-06-06

[Akaa] AKAMAI: *Akamai Online Retail Performance Report: Milliseconds Are Critical.* https://www.akamai.com/de/de/about/news/press/2017-press/akamai-releases-spring-2017-state-of-online-retail-performance-repor jsp, Abruf: 2021-06-06

[Akab] AKAMAI: *Performance Matters. 9 Key Consumer Insights.* https://www.akamai.com/us/en/multimedia/documents/content/akamai-performance-matters-key-consumer-insights-ebook.pdf, Abruf: 2021-06-06

[BAB+20] BHATTI, Anam ; AKRAM, Hamza ; BASIT, Hafiz M. ; KHAN, Ahmed U. ; NAQVI, Syeda Mahwish R. ; BILAL, Muhammad: E-commerce trends during COVID-19 Pandemic. In: *International Journal of Future Generation Communication and Networking* 13 (2020), Nr. 2, S. 1449–1452. – ISSN 2233–7857 IJFGCN

[BGP15] BEKAVAC, Ivan ; GARBIN PRANIČEVIĆ, Daniela: Web analytics tools and web metrics tools: An overview and comparative analysis. In: *Croatian Operational Research Review* 6 (2015), Oktober, Nr. 2, S. 373–386

[CA11] COHEN-ALMAGOR, Raphael: Internet History. In: *International Journal of Technoethics* 2 (2011), April, Nr. 2, 45–64. http://dx.doi.org/10.4018/jte.2011040104. – DOI 10.4018/jte.2011040104. – ISSN 1947–3451, 1947–346X

[CKR] CROCKER, Cliff ; KULICK, Aaron ; RAM, Balaji: *Real-User Monitoring at Walmart. SF & SV Web Performance Group.* https://www.slideshare.net/devonauerswald/walmart-pagespeedslide, Abruf: 2021-06-06

[CP09] CROLL, Alistair ; POWER, Sean: *Complete Web Monitoring*. 1st ed. Beijing; Cambridge [Mass.] : O'Reilly, 2009

[Dev] DEVICEATLAS: *Android v iOS market share 2019*. https://deviceatlas.com/blog/android-v-ios-market-share, Abruf: 2021-06-06

[For] FORRESTER: *The Total Economic Impact Of Accelerated Mobile Pages*. https://amp.dev/static/files/The_Total_Economic_Impact_Of_AMP.pdf, Abruf: 2021-06-06

[GA] GOOGLE ; AWWWARDS: *Speed Matters. Designing for Mobile Performance*. https://www.awwwards.com/brain-food-perceived-performance/, Abruf: 2021-05-06

[Ges21] GESSERT, Felix: *Mobile Site Speed and the Impact on E-Commerce*. https://youtu.be/RTt1RfMUvOQ. Version: 05.06.2021. – code.talks 2019

[Gri13] GRIGORIK, Ilya: *High Performance Browser Networking*. Beijing ; Sebastopol, CA : O'Reilly, 2013

[Hei20] HEINEMANN, Gerrit: *Der neue Online-Handel: Geschäftsmodelle, Geschäftssysteme und Benchmarks im E-Commerce*. Wiesbaden : Springer Fachmedien Wiesbaden, 2020

[Her19] HERMOGENO, Darwin L.: E-Commerce: History and Impact on the Business and Consumers. In: *International Journal of Engineering Science* 9 (2019), Nr. 3

[IKOK10] ISLAM, A.K.M. N. ; KOIVULAHTI-OJALA, Mervi ; KÄKÖLÄ, Timo: A lightweight, industrially-validated instrument to measure user satisfaction and service quality experienced by the users of a UML modeling tool. In: *AMCIS 2010 Proceedings* 8 (2010)

[Jan09] JANSEN, Bernard J.: Understanding User-Web Interactions via Web Analytics. In: *Synthesis Lectures on Information Concepts, Retrieval, and Services* 1 (2009), Januar, Nr. 1, S. 1–102

[KL17] KOHAVI, Ron ; LONGBOTHAM, Roger: Online Controlled Experiments and A/B Testing. In: *Encyclopedia of Machine Learning and Data Mining*. Boston, MA : Springer US, 2017, S. 922–929

[KO20] KUMAR, Vikas ; OGUNMOLA, Gabriel A.: Web Analytics for Knowledge Creation: A Systematic Review of Tools, Techniques, and Practices. In: *International Journal of Cyber Behavior, Psychology and Learning* 10 (2020), Januar, Nr. 1, S. 1–14

[Lin] LINDEN, Greg: *Make Data Useful. Standford Data Mining Class CS345A, 2006.* http://glinden.blogspot.com/2006/12/slides-from-my-talk-at-stanford.html, Abruf: 2021-06-06

[LO20] LANG, Gil ; OTTEN, Steffen: *Erfolgreicher Online-Handel für Dummies*. Weinheim : Wiley-VCH, 2020

[MAC] MEDER, Sam ; ANTONOV, Vadim ; CHANG, Jeff: *Driving user growth with performance improvements*. `https://medium.com/pinterest-engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7`, Abruf: 2021-06-06

[Mar15] MAREK, Kate: *Library technology reports: expert guides to library systems and services. Using Web Analytics in the Library. Volume 47, Number 5*. 2015

[May] MAYER, Marissa: *Conference Keynote, Web 2.0, 2006*

[Mor18] MORYS, André: Mit A/B-Tests die Optimierungsideen validieren. In: *Die digitale Wachstumsstrategie*. Wiesbaden : Springer Fachmedien Wiesbaden, 2018, S. 97–119

[Mos] MOSES, Lucia: *How GQ cut its webpage load time by 80 percent*. `https://digiday.com/media/gq-com-cut-page-load-time-80-percent/`, Abruf: 2021-06-06

[NC11] NAKATANI, Kazuo ; CHUANG, Ta-Tao: A web analytics tool selection method: an analytical hierarchy process approach. In: *Internet Research* 21 (2011), Januar, Nr. 2, S. 171–186

[SB19] STONE, Brad ; BEZOS, Jeffrey: *Der Allesverkäufer: Jeff Bezos und das Imperium von Amazon*. 2. erweiterte Neuausgabe. Frankfurt : Campus Verlag, 2019

[SBR+19] STEIREIF, Alexander ; BÜCKLE, Markus ; RIEKER, Rouven ; ERTLER, Bernhard ; SKIBBA, Janina: *Handbuch Online-Shop: Strategien, Erfolgsrezepte, Lösungen*. 2., aktualisierte und erweiterte Auflage. Bonn : Rheinwerk Verlag, 2019 (Rheinwerk Computing)

[Sch99] SCHWARZ, Kerstin: *DISDBIS*. Bd. 64: *Das Konzept der Transaktionshülle zur konsistenten Spezifikation von Abhänigigkeiten in komplexen Anwendungen*. Infix Verlag, St. Augustin, Germany, 1999

[SKG+] SCHELHOWE, Luetke ; KAGAWA, Shuhei ; GRUDA, Thorbjoern ; CYBULSKI, Jeff ; MARTIN, David: *Loading Time Matters*. `https://engineering.zalando.com/posts/2018/06/loading-time-matters.html`, Abruf: 2021-06-06

[SKS14] SINGAL, Himani ; KOHLI, Shruti ; SHARMA, Amit K.: Web analytics: State-of-art & literature assessment. In: *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*. Noida : IEEE, September 2014, S. 24–29

[SSMG17] SANTOS, Valdeci Ferreira d. ; SABINO, Leandro R. ; MORAIS, Greiciele M. ; GONCALVES, Carlos A.: E-Commerce: A Short History Follow-up on Possible Trends. In: *International Journal of Business Administration* 8 (2017), Oktober, Nr. 7, 130. http://dx.doi.org/10.5430/ijba.v8n7p130. – DOI 10.5430/ijba.v8n7p130. – ISSN 1923–4015, 1923–4007

[Unb] UNBOUNCE: *Think Fast: The 2019 Page Speed Report Stats & Trends for Marketers*. https://unbounce.com/page-speed-report/, Abruf: 2021-05-06

[Wik16] WIKIPEDIA: *Wissenschaftliche Arbeit*. https://de.wikipedia.org/w/index.php?title=Wissenschaftliche_Arbeit&oldid=156007167. Version: 2016, Abruf: 2016-07-21

[WK09] WAISBERG, Daniel ; KAUSHIK, Avinash: Web Analytics 2.0: Empowering Customer Centricity. (2009)

[ZP15] ZHENG, Guangzhi ; PELTSVERGER, Svetlana: Web Analytics Overview. In: KHOSROW-POUR, Mehdi (Hrsg.): *Encyclopedia of information science and technology*. Information Science Reference, 2015, Kapitel 756, S. 7674–7683

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den _____ Unterschrift: _____