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].

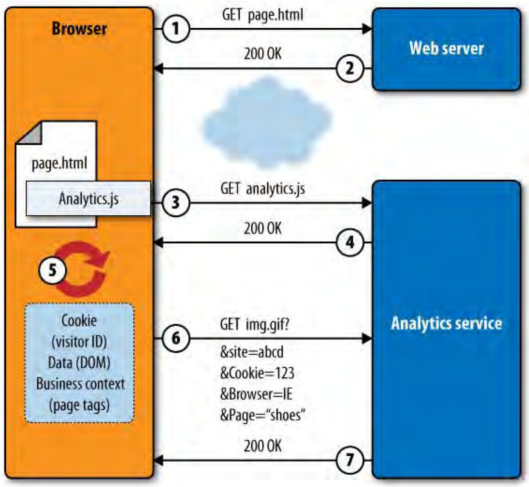


Figure 1.1: Page Tagging

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, 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
- Making the site usable as soon as possible

[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]

- 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

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)

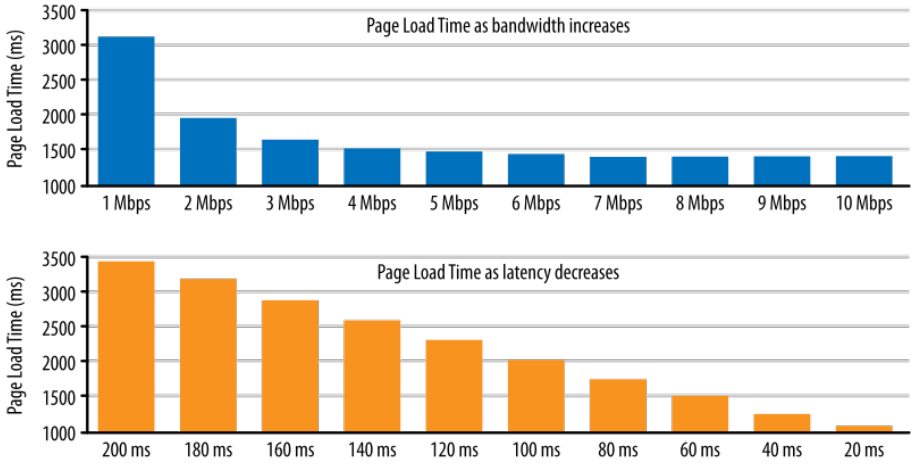


Figure 2.1: Latency vs Bandwidth

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."

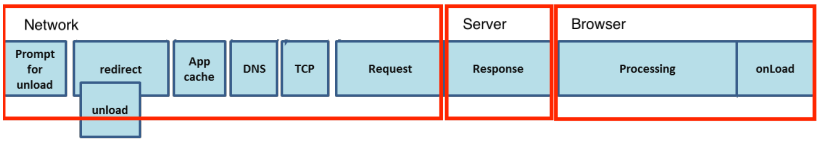


Figure 2.2: Timing Overview

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 called the Critical Rendering Path (CRP). The CRP is the last part of the navigation process as seen in image X.

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. In short, the task of the CRP is to build a render tree, who then can be layout for the desired screen and pixels can be painted on it, see image X.

Once the HTML is received, the browser starts with parsing the HTML and translate it into the DOM. The HTML references CSS and JS files.

The browser will request those external resources when he sees them.

When the The content of the CSS files will be parsed to the CSSOM. JS needs also to be executed. Render Tree: Once DOM and CSSOM are available, the Render Tree is being created. Layout: When the Render Tree is available, Layout is happening. Paint: Finally, pixels can be printed on the screen.

Some requests are blocking, which means the parsing of the rest of the HTML is halted until the imported asset is handled. The browser continues to parse the HTML making requests and building the DOM, until it gets to the end, at which point it constructs the CSS object model. With the DOM and CSSOM complete, the browser builds the render tree, computing the styles for all the visible content. After the render tree is complete, layout occurs, defining the location and size of all the render tree elements. Once complete, the page is rendered, or 'painted' on the screen.

The critical rendering path 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.

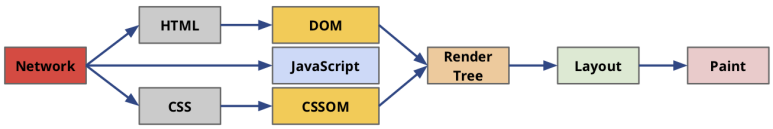


Figure 2.3: Critical Rendering Path

The individual steps are discussed below.

From HTML to DOM In this step, the goal is to convert the content of the received HTML file into a Document Object Model, or DOM. DOM is a tree structure and internal representation of the HTML for the browser. DOM is specified in the living standard. This specification is new. The Parsing of the HTML into the DOM is defined in the HTML standard "The DOM tree captures the properties and relationships of the document markup, but it doesn't tell us how the element will look when rendered. That's the responsibility of the CSSOM." The general process is translating Bytes -> characters -> tokens -> nodes -> object model.

DOM constructoin is the very first step after HTML is received. Subsequent are the creaton of CSSOM which is discussed below, which then leads to the render tree, which will be rendered to the screen.

Therefore, HTML is considered to be a render blocking resource.

Browser will begin parsing as soon as data is being received. That means the DOM is created incrementally which is beneficial for performance.

As we have learned that DOM tree generation is incremental which means as the browser reads HTML, it will add DOM elements to the DOM tree

At some point, the process will encounter a linked and external resource, such as a Cascading Style Sheer or JavaScript. Immediately when external references stylesheet is arrived (<link href..>) browser requests it, that is dispatching HTTP GET request.

as the parser finds any links for CSS or Javascript, it immediatelly sends a request for them, after that it also sends requests for all the other assets found in the rest of the page The sequence is: Build DOM, build CSSOM, Run JS

How CSS and JS is being handled is part of next sections.

- Critical: Resources that delay the first render of the page are considered to be critical. - Non-critical: resources that are loaded after the initial page render are considered to be non-critical, that is, they dont delay the initial render of the page - HTML to be one of our render-blocking, critical resources, we cant render any content if we havent parsed it yet

- HTML and CSS are render-blocking resources

- If these resources are at the bottom of the page, they would be the last things to be parsed, eliminating any blocking that may occur early on. - Non-critical resources are those that provide contributions to secondary/tertiary functionality or styling for the content on your page, e.g., a calendar widget on the sidebar below-the-fold.

- DOM is also exposed, and can be manipulated through various APIs in JavaScript Optimisation: Preload scanner: - This process occupies main thread while browser is building DOM Tree - parse through the content available and request high priority resources like CSS, JavaScript, and web fonts. - will retrieve resources in the background so that by the time the main HTML parser reaches requested assets, they may possibly already be in flight, or have been downloaded

Time it takes for first render is captured in metric TIme to first render, see section X.

From CSS to CSSOM CSSOM is standardized here

- The CSS bytes are converted into characters, then tokens, then nodes, and finally they are linked into a tree structure known as the "CSS Object Model" (CSSOM):

- Parsing of HTML can continue when a CSS file is encountered

Chapter 2 - Request for stylesheet will block parsing

- <link rel="stylesheet" href="a.css">: Browser waits on this line, until file is received

The CSSOM contains all the styles of the page

A CSS is does not block parsing.

Creation of CSSOM happens after DOM is completed.

- DOM and CSSOM are independent data structures.

- CSSOM: Characters -> Tokens -> Nodes -> CSSOM

- CSSOM construction usually very fast

- If we encounter a CSS style sheet (embedded or linked), start building the CSSOM - Turning CSS into the CSSOM is considered to be a “render-blocking” stage just like building the DOM out of our HTML. - If it just went ahead and rendered to pixels without waiting for the CSSOM we’d see a flash of unstyled content (ugly!) for a moment while the CSSOM was parsing. - browser blocks rendering until it has parsed all CSS - Any CSS that is not necessary for the first load can be considered “non-critical”

- As soon as the parser reaches the line with <link rel="stylesheet" href="style.css">, a request is made to fetch the CSS file, style.css - The DOM construction continues, and as soon as the CSS file returns with some content, the CSSOM construction begins

- Partial CSS Tree not possible, because rules can be overwritten - CSS can not be parsed incrementally due to the cascading nature of CSS - Browser needs entire file beforehand (not like with HTML which can be parsed incrementally, e.g. when not the whole file is available) - Then CSSOM is constructed

Is not being created incremental: The incremental processing features don’t apply to CSS like they do with HTML, because subsequent rules may override previous ones. Cascading

- Unlike the DOM tree, CSSOM tree construction is not incremental and must happen in a specific manner. - browsers can’t build the CSSOM tree incrementally as it reads the CSS content. The reason for that is, a CSS rule at the end of the file might override a CSS rule written at the top of the file

- Browser blocks page rendering until it receives all of the CSS -> CSS is render blocking

-> Rendering is blocked on CSS - Browser needs all of the CSS before putting anything on the screen - CSS is truly critical

CSS is a render blocking resource, as rendering of the page can only happen when CSSOM and therefore CSS is available.

- CSS blocks rendering: -> Rules can be overwritten, so that content can’t be rendered until CSSOM is complete

By default, CSS is treated as a render blocking resource, which means that the browser

won't render any processed content until the CSSOM is constructed.

Is render blocking: the browser blocks page rendering until it receives and processes all of the CSS CSS is render blocking because rules can be overwritten, so the content can't be rendered until the CSSOM is complete.

- CSS is a render-blocking resource - Once the browser makes a request to fetch an external stylesheet, the Render Tree construction is halted.

- When parsing is finished, At this point DOM and CSSOM are existing - Redner Tree is built, out of DOM and CSSOM

What happens with JavaScript - When parser is at <script> tag: - Will block rendering of the page - Stop the parser - Fetch the file - Execute JS - Only then proceed

- DOM construction can't proceed until JavaScript is fetched - DOM construction can't proceed until JavaScript is executed - Sync script will block the DOM + rendering of your page:

Scripts on the other hand, which dont have the attribute async or defer, will block the parsing process. This means that excessive scripts can be a significant bottleneck

The JS can change the DOM

- scripts which are not async or defer block parsing -> excessive scripts can be a significant bottleneck

- Javascript is a powerful tool that can manipulate both the DOM and CSSOM, so to execute Javascript, the browser has to wait for the DOM, then it has to download and parse all the CSS files, get to the CSSOM event and only then finally execute Javascript. - When the parser finds a script tag it blocks DOM construction, then waits for the browser to get the file and for the javascript engine to parse the script, this is why Javascript is parser blocking.

- JS Blocking Operation: <script src=""></script>: Will be executed directly when received

- Parser blocking JS - Internal: script elements are parser-blocking. - Every external file requests such as image, stylesheet, pdf, video, etc. do not block DOM construction (parsing) except script (.js) file requests. - When the browser encounters a script element, if it an embedded script, then it will execute that script first and then continue parsing the HTML to construct the DOM tree. - So all embedded scripts are parser-blocking

- If the script element is an external script file, the browser will start the download of the external script file off the main thread but it will halt the execution of the main thread until that file is downloaded. That means no more DOM parsing until the script file is downloaded - Once the script file is downloaded, the browser will first execute the downloaded script file on the main thread (obviously) and then continue with the DOM parsing. If the browser again finds another script element in HTML, it will perform the same operation

- All normal scripts (embedded or external) are parser-blocking as they halt the con-

struction of DOM.

- If we encounter a JS block (not designated as async) while building the DOM, wait for CSSOM construction, stop DOM construction and parse/execute the code. The reason for this is because JS execution can modify the DOM and access/modify the CSSOM.

- you can modify the content and styling of a page using JavaScript - you can remove and add elements from the DOM tree, and you may modify the CSSOM properties of an element via JavaScript as well - whenever the browser encounters a script tag, the DOM construction is paused - The entire DOM construction process is halted until the script finishes executing. - This is because JavaScript can alter both the DOM and CSSOM. - . Because the browser isn't sure what this particular JavaScript will do, it takes precautions by halting the entire DOM construction altogether. - The DOM construction is halted until the script's execution is complete - If you extract the inline script to an external local file, the behavior is just the same. The DOM construction is still halted - If the network is slow, and it takes thousands of milliseconds to fetch app.js, the DOM construction will be halted for the thousands of milliseconds as wel

- But script tags without async or defer attribute block rendering and pause the parsing of HTML. - Though the browser's preload scanner hastens this process.

- JavaScript files added to the <head> section of the document are treated as render blocking resources by default. - You can remove them from the critical rendering path by placing the <script> tags right before the closing </body> tag instead of the <head> section. - In this case, they only begin to download after the entire HTML has been downloaded.

- By default, every script is a parser blocker! The DOM construction will always be halted.

Chapter 10: - Script execution can issue a synchronous doc.write and block DOM parsing and construction - DOM construction cannot proceed until JavaScript is executed, and JavaScript execution cannot proceed until CSSOM is available

- JS can change styling of page -> Browser blocks JS until CSS is resolved

- So, what happens when the parser encounters a script tag but the CSSOM isn't ready yet? - Well, the answer turns out to be simple: the Javascript execution will be halted until the CSSOM is ready. - So, even though the DOM construction stops until an encountered script tag is encountered, that's not what happens with the CSSOM. - With the CSSOM, the JS execution waits. No CSSOM, no JS execution

Chapter 10: - JavaScript can also block on CSS

- Directly use async attribute: - Is a promise that script will not use document.write - Therefore parser can continue - Download JS while continue with DOM parsing - Once its available execute it

Async scripts don't block DOM construction and don't have the need to wait for the CSSOM event - Async: - <script src="" async></script> - Loads script asynchronously to the page - DOM parsing continues

- asynchronous scripts are independent of the DOM. - On the other hand, the async attribute informs the browser that a script is completely independent of the page. - It will download in the background as a non-render blocking resource, just like deferred scripts - However, unlike deferred scripts, async scripts don't follow the document order, so they will execute whenever they finish downloading — which can happen at any time - The async attribute is recommended for independent third-party scripts, such as ads, trackers, and analytics scripts. For example, Google Analytics recommends adding the async attribute to support asynchronous loading in modern browsers.

- async: When DOM parser encounters an external script element with async attribute, it will not halt the parsing process while the script file is being downloaded in the background. But once the file is downloaded, the parsing will halt and the script (code) will be executed. - All async scripts (AKA asynchronous scripts) do not block parser until they are downloaded. - As soon as an async script is downloaded, it becomes parser-blocking.

- If you add the async keyword to the script tag, the DOM construction will not be halted. The DOM construction will be continued, and the script will be executed when it is done downloading and ready.

- async attribute not applicable on inline scripts

- deferred scripts respect the document order - The defer attribute instructs the browser to download the script in the background so it won't block the rendering of the page - The deferred script executes once the DOM is ready but before the DOMContentLoaded event fires - Deferred scripts follow the document order, just like nondeferred, default scripts. - You can't add defer to inline scripts; it only works with external scripts that specify the script's location using the src attribute. - The defer attribute is recommended for scripts that need the DOM, but you want to begin to download them before the document loads, without making them a render blocking resource. - You should also use defer rather than async if the document order is important — for instance, when consecutive scripts depend on each other.

- defer: We also have a magical defer attribute for the script element which works similar to the async attribute but unlike the async attribute, the script doesn't execute even when the file is fully downloaded. All defer scripts are executed once the parser has parsed all HTML which means the DOM tree is fully constructed. Unlike async scripts, all defer scripts are executed in the order they appear in the HTML document (or DOM tree). - However, all defer scripts (AKA deferred scripts) are non-parser-blocking script as they do not block the parser and execute after the DOM tree is fully constructed.

Building the Render Tree HTML and CSS are both render blocking, as they prohibit the rendering of the render tree.

- DOM + CSSOM: Content and styles - Only captures visible content - e...g when in CSS Display: none -> Children also skipped, because style cascades down - The hidden element will be present in the DOM but not the render tree

- The DOM and CSSOM trees are combined to form the render tree. - Render tree contains only the nodes required to render the page.

Render: - Build Render Tree - Create render tree out of DOM and CSSOM, aka computed style tree - Tree is used to compute layout - The render tree holds all the visible nodes with content and computed styles

Render Tree: - First, the browser removes all non-visible elements. This includes elements such as <head>, <script>, and <meta>, as well as HTML elements that have the hidden attribute

Layout - Position of the elements on the page - Calculating positions and dimensions

- Layout viewport size - <meta name="viewport" content="width=device-width"> - Default width: 980px, used when meta tag is not set - Layout triggered by device orientation, window resize, adding or removing from DOM tree, etc.

4. Layout: - Run layout on render tree - Compute geometry of each node - Exact size and location of each object - Subsequent changes are called reflow, e.g. when image is received later without specify its size

- Layout determines the size and location of everything on the page. Once layout is determined, pixels are painted to the screen. - After the render tree is complete, layout occurs, defining the location and size of all the render tree elements - Layout is dependent on the size of screen. The layout step determines where and how the elements are positioned on the page, determining the width and height of each element, and where they are in relation to each other. - This width of the device impacts layout - The viewport meta tag defines the width of the layout viewport, impacting the layout. - Without it, the browser uses the default viewport width, which on by-default full screen browsers is generally 960px. On by-default full screen browsers, like your phone's browser, by setting <meta name="viewport" content="width=device-width" - Layout happens every time a device is rotated or browser is otherwise resized. - Layout performance is impacted by the DOM – the greater the number of nodes, the longer layout takes. - Any time the render tree is modified, such as by added nodes, altered content, or updated box model styles on a node, layout occurs

- Layout computes the exact position and size of each object.

- Layout is where the browser figures out where elements go and how much space they take up. - When calculating layout, the browser has to start at the top of the render tree and move downward, since each element's positioning, width, and height is calculated based off of the positioning of its parent nodes

Paint - Paint nodes on screen -> First Meaningful Paint - Time to Interactive

- Once layout is determined, pixels are painted to the screen. - The last step is painting the pixels to the screen. Once the render tree is created and layout occurs, the pixels can be painted to the screen - Onload, the entire screen is painted. After that, only impacted

areas of the screen will be repainted, as browsers are optimized to repaint the minimum area required

- Paint: It's important to remember that some CSS properties can have a larger impact on the page weight than others (for example, a radial-gradient is much more complex to paint than a simple color).

- Above the fold: "Above-the-Fold" refers to the area that the visitor normally sees on a website before scrolling down to see the rest of the content.

2.1.3 Conclusion Technical Background

2.2 Measurement Methods

- synthetic monitoring - RUM - other methods briefly described

- Explanation and comparison of synthetic and real-user monitoring with concrete examples
- Short overview of other measuring methods such as log analysis or surveys

synthetic: - lab environment: geography, network, device, browser, etc. - control variables to identify performance issues. this does not reflect real world and real user experience - automated - simulate user paths - traffic is generated artificial and is not by real users - can also be used for live system monitoring - fairly easy to implement, inexpensive

RUM: - measure from real users machine - part of page tagging (same technique with including some JS) - measures actual use cases

- Synthetic and Real-User Performance Measurement

- Log file - Synthetic: will be discussed in chapter X - RUM which will be covered in chapter X - CrUX - Surveys

- Pingdom - GTMetrix - Website Grader - Site Speed checker

Ch 6 Measuring and Iterating on Performance

- Browser Tools - Synthetic Testing - Real User Monitoring - Changes over Time

2.2.1 Synthetic Monitoring

- What is it
- How does it work
- Application, real life scenario
- Examples:
  - WebPageTest
  - Google Lighthouse
  - Other solutions

2.2.2 Real-User Monitoring

- What is it
- How does it work
- Application, real life scenario
- Examples:
  - Google Analyitcs
  - CrUX
  - SpeedKit
  - Other solutions

2.2.3 Other methods

Reports such as CruX or http archive surveys log files

- Log file - Synthetic: will be discussed in chapter X - RUM which will be covered in chapter X - CrUX - Surveys

2.3 Metrics

- 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

Check out glossary: https://developer.mozilla.org/en-US/docs/Glossary

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

First Paint:

First Contentful Paint:

First Meaningful Paint:

Largest Contentful Paint:

Speed Index:

Time to Interactive:

Performance APIs: - Many Web APIs available - Performance API is one API within Web APIs collection: Includes other APIs - Navigation Timing API: Famous image: Exposes metrics related to navigation events

Tools and metrics: - 2 categories: Tools for measuring (reporting) and tools for improving - Reporting: - PageSpeedInsights - WebPageTest - Network: - Network Panel in DevTools

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

Navigation Timing API Navigation Timing API Level 2

Performance Timeline API

Performance Entry API

PerformanceNavigationTiming

User Timing API

Resource Timing API

Performance Observer API

User Timing API Navigation Timing API: Level 1 (performance.timing), Level 2 (PerformanceNavigationTiming) ? Network Information API Resource Timing API Paint Timing API High Resolution Time API Performance Timeline API Performance Observer API Long Tasks API Element Timing API Event Timing API Server Timing API

2.3.1 Introduction

- Metrics jungle, difficulty of taxonomy
- Performance vs UX

2.3.2 "Non-Performance" metrics

- User engagement: session length, bounce rate, etc.
- Business KPIs: Cart size, conversion rate, etc.
- QA metadata: Page views, JS errors, etc.
- Hit
- Click-Through

- Page View
- Visit
- Visitor / Unique Visitor
- Referrer
- Conversion Rate
- Abandonment Rate
- Attrition
- Loyalty, Frequency and Recency
- Measuring Reach: ...
- Measuring Acquisition: ...
- Measuring Conversion: ...
- Measuring Retention: ...
- Basic metrics (see table): basic metrics are meaningless
- Advanced metrics: Customer lifecycle analysis, customer behaviour analysis
- Types: Counts, Rations, KPIs
- Definitions for all terms, like Page view, unique visitor, etc.
- Importance of setting goals
- Conversion Rate
- Kennzahlen für Websites nach Typ: ROI-Ebene, Online-Shop, ...
- Conversion Rates, pages that visitors abandon most
- Click throughs
- UGC (User generated content)
- Subscriptions, Signups
- Referring URL
- Visitor Motivaton, VOC: Voice of the Customer
- Ad and campaign effectiveness

- Findability and Search Effectiveness
- Trouble Ticketing and Escalation
- Loyalty: Ratio of new to returning visitors; average time between visits; time since last login; rate of attrition or disengagement

p.15 "whether your business benefited in some way from their visits."
The percentage of visitors that your site converts to contributors, buyers, or users is the most important metric you can track -> Conversion Rate
p. 74 Page View, first useful web analytics metric

- 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

- GA basic metrics: Visits, Bounce Rate, Page views,pages per visit, avg time on site, percentage new visits

- Erfolg messen und bewerten
- Traffic:
  - Page Impression / Page View
  - Visit
  - Visitor / Unique visitor
- Bounce rate

- Conversion rate
- CTR: Click-through-rate
- Session length
- Good metrics should be: Uncomplex, Relevant, Timely, Instantly Useful
- Basic metrics: Visits, bounce rate, page views, pages/visits, avg time, % new visits
- Guidance Performance Indicator (GPI) metric
- Visit count: page view, visit, unique visitor
- Visit duration: time on page, time on site.
- Bounce rate and exit rate.
- Besucheranalyse: Wie viele Besucher?, Anzahl Besucher mit Mobilgerät, Demographische Daten (Geschlecht, Altersgruppe)
- Seitenanalyse: Was machen die Besucher im Shop?, Zielseite / Startseite: Erste Seite, die ein Besucher angeschaut hat, Ausstiegseite
- E-Commerce-Analyse: Transkations-daten aus Shop, Funnel-Analyse
- Types: Anzahl, Relations, Werte
- Content: Where, Who, How, What
- Hits
- Page Views
- Visits / Sessions
- Visitor / Unique Visitor

2.3.3 Performance Metrics

- Introduction to the Web Performance Working Group
- Overview of Browser APIs and the data they expose: High Resolution Time API, Navigation Timing API, etc.
- If possible make one deep dive into one API: What exactly gets measured? Maybe check out html standard, v8 or chromium implementation, etc.

Standards and APIs, Browser metrics, standards

- Web Performance Working Group
- User Timing API
- Navigation Timing API: Level 1 (performance.timing), Level 2 (PerformanceNavigationTiming) ?
- Network Information API
- Resource Timing API
- Paint Timing API
- High Resolution Time API
- Performance Timeline API
- Performance Observer API
- Long Tasks API
- Element Timing API
- Event Timing API
- Server Timing API

Navigation Timing API

- Show image of navigation timings
- Explain one or two events directly with specification: navigationStart, domInteractive, etc.

Google metrics? User-centric / UX / visual

Web Vitals

- 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
- Introduction, what is it
- How to measure
- How to improve
- Introduction, what is it
- How to measure
- How to improve

Others

- Visually complete ?

Speed Index

2.3.4 WebPageTest 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 2.1: Your caption here

2.3.5 Google Analytics 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.

2.3.6 Comparison

- We can show this 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 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. - ...

3.2 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 code example
- 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...

3.3 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

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

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>
```

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>
```

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

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

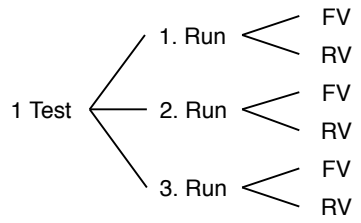
Table 4.1: Test Runs [Sch99]

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: ...

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



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

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

Configuration 2 Emulate Mobile Browser

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

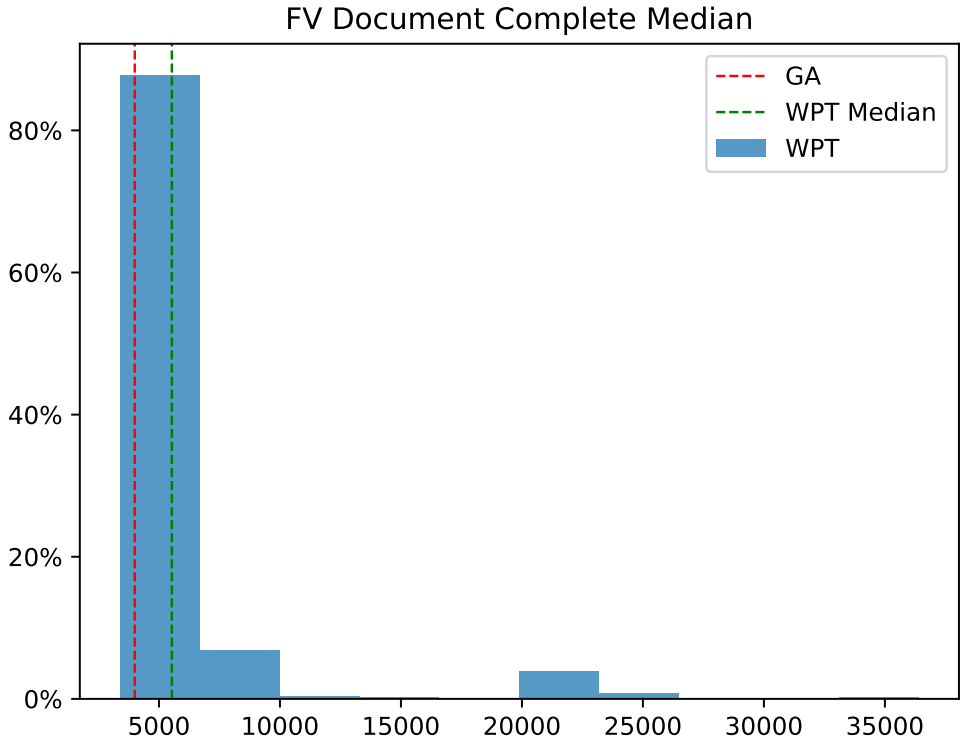
4.3.4 Test Protocol

- 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: _____