

The Impact of De-Cluttering JavaScript Code in Mobile Web Pages with respect to Energy Consumption

Geoffrey van Driessel

2639310

VU Amsterdam

g.r.van.driessel@student.vu.nl

Sophie Vos

2551583

VU Amsterdam

s.o.vos@student.vu.nl

Abijith Radhakrishnan

2667575

VU Amsterdam

a.radhakrishnan@student.vu.nl

Marc Wiggerman

2653796

VU Amsterdam

m.g.wiggerman@student.vu.nl

ABSTRACT

Context.  [at the end](#)


Goal.  [at the end](#)

Method.  [at the end](#)

Results.  [at the end](#)

Conclusions.  [at the end](#)

ACM Reference Format:

Geoffrey van Driessel, Abijith Radhakrishnan, Sophie Vos, and Marc Wiggerman. 2020.  [The Impact of De-Cluttering JavaScript Code in Mobile Web Pages with respect to Energy Consumption](#). In *Green Lab 2020/2021 - Vrije Universiteit Amsterdam, September–October, 2020, Amsterdam (The Netherlands)*. ACM, New York, NY, USA, 3 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

CISCO predicted global internet traffic in 2020 to be equivalent to 95x the volume of the entire global internet in 2005 [1]. Since the end of the second quarter of 2020, mobile devices (excluding tablets) have been a major contributor to global internet traffic. It accounts for 51.53% of global website traffic and has an ascending trend. Corresponding to the increased traffic, the complexity of modern web pages has considerably advanced in the previous decade, leading to bigger web pages that are computationally intense for mobile devices.

The extended complexity in websites can be mainly attributed to the inception of web and JavaScript (JS) frameworks (such as jQuery, Angular, React) [2] which provide swift solutions for web development with function-specific modules and libraries that can fast pace the programming process. The debut of Single Page Applications (SPAs) further paved the way for more frameworks and standardization of web development. Nevertheless, this also led to JS files in websites being bloated with unused and unnecessary

functions from the frameworks and ergo higher computational complexity.

A computationally intensive website leads to higher energy consumption as well as increased Page Load Time (PLT). It has been shown that web pages that have a PLT higher than 3 seconds, have a probability of losing 32% of its visitors [3]. Even if the energy usage of a single smartphone may be considered modest, the overall energy footprint left by mobile devices is far higher [4]. This was further extended by top-end smartphones and 4G network [5]. Google handled this by using lighter variants of web pages through Accelerated Mobile Pages (AMP). Another approach for reducing PLT commonly used by web developers is concatenation, minification, and uglification of different JS files in the web page. Even though this approach provides a slightly faster PLT because of smaller file size [6], JS computations in the browser remain the same. Furthermore, social media websites such as Facebook offers a lighter version of their service (Facebook Lite) which is better suited for low-power devices and more energy efficient.

Another method to reduce PLT by reducing the amount of JS code is introduced by Chaqfeh et al. [7]. Chaqfeh et al. [7] introduced *JS Cleaner*: a classification algorithm that classifies critical, replaceable, and non-critical JS code. Accordingly, the replaceable JS code is replaced with HTML counterparts, and non-critical JS code is eliminated. This process is referred to as ‘de-cluttering’. The authors studied the relation between the de-cluttered web pages and the PLT. They concluded that de-cluttering achieves a 30% reduction in PLT. A different study, conducted by Thiagarajan et al. [8] showed that JS code has a relatively high energy consumption compared to other web elements. Since energy-efficiency contributes to the user experience and impact of increasingly popular mobile web pages, we aim to contribute by investigating the impact of eliminating JS code on energy consumption. Our research focuses on the relation between de-cluttered JS code using *JS Cleaner* on mobile web pages and its impact on energy consumption. Hence, our research question is: “*What is the impact of de-cluttering JavaScript code in mobile web pages using JS Cleaner with respect to the energy consumption?*”.

The remainder of this paper is structured as follows. Section 2 provides an insight into related work, and how this paper is compared to these related works. Next, section 3 defines the experiment using the GQM method.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Green Lab 2020/2021, September–October, 2020, Amsterdam, The Netherlands

© 2020 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

2 RELATED WORK

To the best of our knowledge, this paper is the first one to analyze and compare the energy consumption of mobile web pages using (1) the full JavaScript (JS) code base, and (2) the de-cluttered JS code base. Most of the research up until now focused on improving and analyzing the performance increase with relation to the Page Load Time (PLT). First we describe the work of Obbink et al. [9] who introduced a tool called *Lacuna*. *Lacuna* automatically eliminates dead JS code from web apps, which in effect improves the PLT and energy consumption by reducing the required network and browser interpretation time. Obbink et al. [9] apply different analysis techniques to discover and eliminate dead code from the code base without changing the features of the web app. After testing *Lacuna* on 29 web apps, the results showed that *Lacuna* could be integrated into real-world build systems due to its reasonable execution time.

Second, Chaqfeh et al. [7] present a way to automatically de-clutter JS code using a tool called *JS Cleaner*. The proposed JS de-cluttering tool helps to reduce PLT in web pages without significantly jeopardizing its content and functionality. It follows a rule-based classification algorithm that divides JS into three classes: non-critical, replaceable, and critical. It then deletes non-critical JS from the web pages, interprets replaceable JS components with their corresponding HTML code, and maintains critical JS without any modification. The paper then quantitatively analyzed 500 popular web pages to achieve a 30% reduction in PLT and 50% reduction in the number of requests and web page size using the *JS Cleaner* tool. For qualitative evaluation, the authors recruited 103 users to analyze the difference between original and de-cluttered JS web pages. The de-cluttered web pages maintained the appearance of the actual web pages with a median value of over 93%. Both papers [7, 9] introduce a different way to improve the PLT. However, they do not analyze the change in energy consumption of the original versus the cleaned code base.

Other studies perform a similar research methodology, only to accomplish a different goal. For example, Malavolta et al. [10] performed an experiment to test the performance and energy efficiency of PWAs while tuning the browser’s cache behavior. The study involved measuring the energy consumption in joules and the PLT in milliseconds during the initial load of nine PWAs on a single Android device running in the Firefox browser. The authors concluded that the current results do not show evidence that adding caching to a website improves its energy efficiency. Despite the goal being different from the one presented in our current paper, the experimental setup and execution are similar to the ones presented in the current research.

Another study that has a similar research methodology is the work of Chan-Jong-Chu [11]. The authors conducted an empirical experiment using mobile web apps with the aim to find a correlation between performance and energy consumption. The study has a clear research setup in which the correlation is proved using statistical tests together with a thorough description of the experiment execution. Hence, the study is suitable for replication and therefore has high scientific validity. We provide a similar setup to ensure that the measured difference in energy consumption with and without *JS Cleaner* is statistically significant and the study can be precisely replicated.

Lastly, Thiagarajan et al. [8] studied the energy consumption of downloading and rendering web pages on mobile devices by comparing the different energy usage (in joules) of web elements. This is relevant to our research as we aim to understand the energy consumption of a specific modification in a mobile web page. Thiagarajan et al. [8] aim to advise developers in designing energy-efficient mobile web sites. They concluded that the usage of JS in mobile web pages has a negative effect on energy consumption compared to HTML elements. This is relevant to our study as we measure (a potential) energy gain by eliminating non-critical JS code or replacing the JS elements with HTML code. The research of Thiagarajan et al. [8] differs from our study as they used dedicated hardware to measure the energy consumption whereas we measured the energy consumption using Android Runner Software [12].

3 EXPERIMENT DEFINITION

As described in the related work section, *JS Cleaner* is able to reduce the size of JS files significantly and, therefore, reduce the PLT. However, it has not been shown that this actually reduces the energy consumption. The **goal** of our study is: “*Analyze JS Cleaner for the purpose of evaluation with respect to the energy consumption from the point of view of software developers in the context of mobile web pages*”. This goal has been established using the GQM method [13], the steps are presented in Table 1.

Table 1: GQM

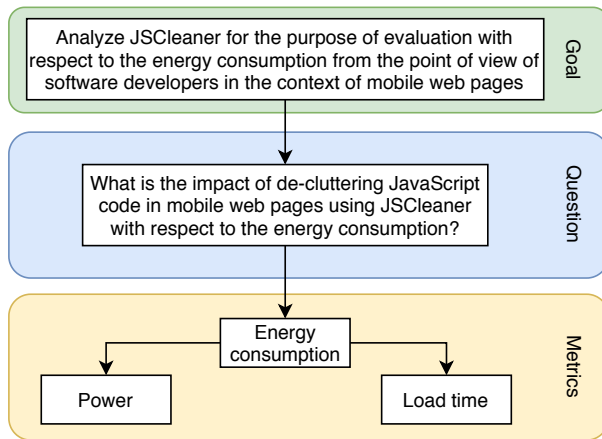
Analyse	JS Cleaner
For the purpose of	evaluating
With respect to the	energy consumption
From point of view of	developer
In the context of	mobile web pages

Following our goal we have identified the main **research question**: [RQ1] “*What is the impact of de-cluttering JavaScript code in mobile web pages using JS Cleaner with respect to the energy consumption?*”. We answer this research question by comparing the energy consumption of 500 web pages compared to their de-cluttered version.

To answer our research question the *energy consumption (joule)* is used as **metric**. The energy consumption is calculated using the measured *power consumption (joule per second)* times the *PLT (seconds)*.

Figure 1 presents a visual representation of the GQM. It hierarchically shows how the goal is obtained using the research question, and how the research question is answered using the metrics.

Figure 1: GQM tree



4 EXPERIMENT PLANNING

4.1 Subjects Selection

4.2 Experimental Variables

4.3 Experimental Hypotheses

4.4 Experiment Design

4.5 Data Analysis

Page limit: 3

5 EXPERIMENT EXECUTION

Report about: how you plan to conduct your experiment, which tools you are going to use, which devices/laptops, figure and description of the overall software/hardware infrastructure you are setting up for the experiment (e.g., who communicates with whom, proxies, network requests, order of actions, etc.).

Page limit: 2

6 RESULTS

Provide:

- descriptive statistics
- hypothesis testing

Provide suitable plots and tables to illustrate your results.

Page limit: Open - go deep as you wish

7 DISCUSSION

Report implications and interpretations of your results (possibly grouped by research question).

Page limit: 1

8 THREATS TO VALIDITY

Report about each type of threat to the validity of the experiment, according to the classification discussed in class.

8.1 Internal Validity

8.2 External Validity

8.3 Construct Validity

8.4 Conclusion Validity

Page limit: 1

9 CONCLUSIONS

One brief paragraph for summarizing the main findings of the report.

One brief paragraph about the possible extensions of the performed experiment (imagine that other 3 teams will be assigned to the extension of your experiment).

REFERENCES

- [1] CISCO. Global - 2020 forecast highlights. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2020_Forecast_Highlights.pdf
- [2] M. Persson, "Javascript dom manipulation performance: Comparing vanilla javascript and leading javascript front-end frameworks," 2020.
- [3] Google. Find out how you stack up to new industry benchmarks for mobile page speed. [Online]. Available: <https://think.storage.googleapis.com/docs/mobile-page-speed-new-industry-benchmarks.pdf>
- [4] TIME. The surprisingly large energy footprint of the digital economy. [Online]. Available: <https://science.time.com/2013/08/14/power-drain-the-digital-cloud-is-using-more-energy-than-you-think>
- [5] X. Chen, Y. Chen, M. Dong, and C. Zhang, "Demystifying energy usage in smart-phones," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–5.
- [6] W. Craig. A modern approach to improving website speed. [Online]. Available: <https://www.webfx.com/blog/web-design/improve-website-speed/>
- [7] M. Chaqfeh, Y. Zaki, J. Hu, and L. Subramanian, "Jscleaner: De-cluttering mobile webpages through javascript cleanup," in *Proceedings of The Web Conference 2020*, ser. WWW '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 763–773. [Online]. Available: <https://doi.org/10.1145/3366423.3380157>
- [8] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery? analyzing mobile browser energy consumption," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 41–50. [Online]. Available: <https://doi.org/10.1145/2187836.2187843>
- [9] N. G. Obbink, I. Malavolta, G. L. Scoccia, and P. Lago, "An extensible approach for taming the challenges of javascript dead code elimination," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2018, pp. 291–401.
- [10] I. Malavolta, K. Chinnappan, L. Jasmontas, S. Gupta, and K. Ali Karam Soltany, "Evaluating the impact of caching on the energy consumption and performance of progressive web apps," 10 2020.
- [11] K. Chan-Jong-Chu, T. Islam, M. M. Exposito, S. Sheombar, C. Valladares, O. Philippot, E. M. Grua, and I. Malavolta, "Investigating the correlation between performance scores and energy consumption of mobile web apps," ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 190–199. [Online]. Available: <https://doi.org/10.1145/3383219.3383239>
- [12] I. Malavolta, E. M. Grua, C.-Y. Lam, R. de Vries, F. Tan, E. Zielinski, M. Peters, and L. Kaandorp, "A Framework for the Automatic Execution of Measurement-based Experiments on Android Devices," in *35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW '20)*. ACM, 2020. [Online]. Available: https://github.com/S2-group/android-runner/blob/master/documentation/A_Mobile_2020.pdf
- [13] M. H. M. C. O. B. R. A. W. Claes Wohlin, Per Runeson, "Experimentation in software engineering." Springer US, 2012.