The Impact of Cluttered JavaScript Code in Mobile Web Apps with respect to Energy Consumption

Geoffrey van Driessel
2639310
VU Amsterdam
g.r.van.driessel@student.vu.nl

Abijith Radhakrishnan
2667575
VU Amsterdam
a.radhakrishnan@student.vu.nl

Sophie Vos
2551583
VU Amsterdam
s.o.vos@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 Cluttered JavaScript Code in Mobile Web Apps with respect to Energy Consumption . In *Green Lab 2020/2021 - Vrije Universiteit Amsterdam, September–October, 2020, Amsterdam (The Netherlands).* ACM, New York, NY, USA, 6 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. Mobile devices account for 51.53% of global web traffic and has an ascending trend [1]. Corresponding to the increased traffic, the complexity of modern mobile web apps has advanced in the previous decade, leading to larger mobile web apps that are computationally intense for mobile devices [2].

The increased complexity in mobile web apps can be mainly attributed to the inception of web and JavaScript (JS) frameworks (such as jQuery, Angular, React) [3] which provide swift solutions for web development with function-specific modules and libraries that can fast pace the development 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 mobile web apps being bloated with unused and unnecessary functions from the frameworks and ergo higher computational complexity [4].

A computationally intensive website leads to higher energy consumption [5] as well as increased Page Load Time (PLT) [6]. It is shown that mobile web apps that have a PLT higher than 3 seconds, have a probability of losing 32% of its visitors [7]. Even if the energy usage of a single smartphone may be considered modest, the overall energy footprint left by mobile devices is far higher [8]. This is further extended by top-end smartphones and the 4G network [9]. Google handles the increasing energy consumption and PLT of mobile web apps by using lighter variants of the mobile web apps through Accelerated Mobile Pages (AMP) [10]. Another approach for reducing PLT commonly used by web developers is concatenation, minification, and uglification of different JS files in the mobile web apps. Even though this approach provides a slightly faster PLT because of a smaller file size [11], JS computations in the browser remain the same. Furthermore, social media mobile web apps, such as Facebook, offer a lighter version of their service (Facebook Lite[1]) which is better suited for low-power devices.

Another method to reduce PLT by reducing the amount of JS code is introduced by Chaqfeh et al. [6]. Chaqfeh et al. [6] introduced *JSCleaner*: 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 cluttered mobile web apps and the PLT. Their results showed a 50% reduction in requests and a 30% reduction in page size of the de-cluttered mobile web app compared to the original/cluttered mobile web app. They concluded that de-cluttering achieves a 30% reduction in PLT. A different study, conducted by Thiagarajan et al. [12], 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 apps, we aim to contribute by investigating the impact of cluttered JS code on energy consumption. In our study, we use JSCleaner to de-clutter JS code in mobile web apps. Our research question is: "*What is the impact of cluttered JavaScript code in mobile web apps with respect to the energy consumption?*". We answer this research question by

[1]https://www.facebook.com/lite

measuring and comparing the energy consumption of cluttered and de-cluttered mobile web apps.

The remainder of this paper is structured as follows: Section 2 provides an insight into related work. Next, Section 3 defines the experiment using the GQM method. Afterwards, we outline our experiment planning in Section 4 and describe the experiment execution in Section 5.

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 apps using (1) the cluttered 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 PLT. First, we describe the work of Obbink et al. [13], who introduced a tool called *Lacuna*. Lacuna automatically eliminates dead JS code from mobile web apps, which in effect improves the PLT and energy consumption by reducing the required network and browser interpretation time. Obbink et al. [13] apply different analysis techniques to discover and eliminate dead code from the code base without changing the features of the mobile web app. After testing Lacuna on 29 mobile 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. [6] present a way to automatically de-clutter JS code using a tool called JSCleaner. The proposed JS de-cluttering tool helps to reduce PLT in mobile web apps without significantly jeopardizing its content and functionality. It follows a rule-based classification algorithm that divides JS code into three classes: non-critical, replaceable, and critical. It then deletes non-critical JS code from the mobile web apps, interprets replaceable JS components with their corresponding HTML code, and maintains critical JS code without any modification. The study then quantitatively analyzed 500 popular mobile web apps and reports an empirical evaluation in which a 30% reduction in PLT and a 50% reduction in the number of requests and size using the JSCleaner tool is presented. A qualitative evaluation followed in which the authors recruited 103 users to analyze the difference in functionality between cluttered and de-cluttered mobile web apps. The de-cluttered mobile web apps maintained the appearance of the cluttered mobile web apps with a median value of over 93%. Both papers [6, 13] introduce a different way to improve the PLT. However, they do not analyze the change in energy consumption of the cluttered versus the de-cluttered code base.

Other studies perform a similar research methodology, only to accomplish a different goal. For example, Malavolta et al. [14] performed an experiment to test the performance and energy efficiency of Progressive Web Applications (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 study, the experimental setup and execution are similar to the ones presented in our study.

Another study that has a similar research methodology is the work of Chan-Jong-Chu [15]. 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. We provide a similar setup to ensure that the measured difference in energy consumption of cluttered and de-cluttered mobile web apps is statistically significant and the study can be precisely replicated.

Lastly, Thiagarajan et al. [12] studied the energy consumption of downloading and rendering mobile web apps 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 app. Thiagarajan et al. [12] aim to advise developers in designing energy-efficient mobile web apps. They concluded that the usage of JS code in mobile web apps 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 main difference between the study of Thiagarajan et al. [12] and our study is that the work of Thiagarajan et al. is more descriptive (they are trying to understand a phenomenon), whereas we aim at discovering a specific relationship (i.e., the one between being cluttered and consuming different amounts of energy).

3 EXPERIMENT DEFINITION

As described in the related work section, JSCleaner reduces 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 cluttered JavaScript code 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 apps*". This goal has been established using the GQM method [16], the steps are presented in Table 1.

Analyze	cluttered JavaScript code
For the purpose of	evaluating
With respect to the	energy consumption
From point of view of	software developers
In the context of	mobile web apps

Table 1: Goal

Following our goal, we have identified the **research question (RQ)**: "*What is the impact of cluttered JavaScript code in mobile web apps with respect to the energy consumption?*". We answer this research question by comparing the energy consumption of popular mobile web pages compared to their de-cluttered version. The mobile web apps are de-cluttered by JSCleaner.

To answer our research question the *energy consumption (in joules)* is used as **metric**. The energy consumption is calculated using the measured *power consumption (in watts)* times the *PLT* *(in seconds)*. Furthermore, we measure the *CPU load (%)* and *memory usage (MB)* to understand possible fluctuations in the energy consumption.

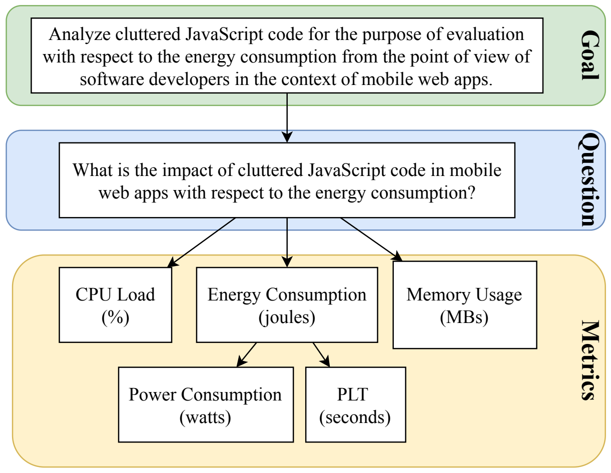
Analyze cluttered JavaScript code for the purpose of evaluation with respect to the energy consumption from the point of view of software developers in the context of mobile web apps.
Goal
What is the impact of cluttered JavaScript code in mobile web apps with respect to the energy consumption?
Question
CPU Load (%)
Energy Consumption (joules)
Memory Usage (MBs)
Power Consumption (watts)
PLT (seconds)
Metrics

Figure 1: The GQM Tree

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.

4 EXPERIMENT PLANNING

In this section, we explain how the experiment is laid out. This is done in the following steps: context, subjects, experimental variables, hypothesis, experiment design, and data analysis.

4.1 Context Selection

To characterize the experiments of our study, the four dimension method presented by Wohlin et al. [17] is used.

The first dimension regards the choice between on-line and off-line experiments. As the experiments conducted in our study use a local Android device loading pre-developed mobile web apps, the off-line dimension is selected.

The second dimension represents the choice of using students versus professionals as subjects. This dimension is not suitable for our study as we use mobile web apps as subjects. No humans are the subject of the performed experiments.

The third dimension determines if the nature of the experiments are real or toy problems. As the mobile web apps used in this study are collected from real mobile web apps, the experiments and results presented in this paper can be regarded as covering real problems. Moreover, the outcome of our study could also directly impact real problems, i.e., developers could immediately tackle the energy efficiency of their real mobile web apps using a de-cluttering tool.

The fourth and final dimension defines whether the experiments are specific versus general. Our study can be regarded as general as it uses diverse and frequently visited mobile web apps to conduct the experiments. No specific type of mobile web app is selected.

4.2 Subjects Selection

The subject selection is done by a python script ✎ link to public GitHub that reads a CSV file containing 93 mobile web apps and their corresponding size. The mobile web apps are a duplicate of the set used by Chaqfeh et al. [6] containing 93 popular mobile web apps. The set is obtained from The Majestic Million[2] and includes a wide variety of mobile web apps such as: news sites, education, sports, entertainment, travel, and government web apps. This ensures the generalizability of our sample to the real-world population of mobile web apps. The mobile web apps are available on a proxy server provided by Chaqfeh et al. [6]. The proxy server contains both the cluttered mobile web apps and the mobile web apps de-cluttered by JSCleaner.

To perform the analysis required for the feature selection, each mobile web app is downloaded locally using *Resources Saver Extension* for *Google Chrome*[3] after which its size is measured and a *Lines Of Code* analysis is performed using *CLOC*[4]. The mobile web apps with a size of 0 bytes, duplicate mobile web apps, and mobile web apps which – according to CLOC – do not contain JS, HTML or CSS code are discarded. This deletion process results in 70 mobile web apps which are suitable for analysis.

After this cleaning step, the mobile web apps are divided into three groups based on the size of the cluttered mobile web app. These groups are created to control the effect of size on the energy consumption. The groups are created using three quantiles and contain the following size ranges:

Small	0.0 MB - 2.9 MB	(25 mobile web apps)
Medium	3.1 MB - 5.9 MB	(22 mobile web apps)
Large	6.1 MB - 46 MB	(23 mobile web apps)

After this division, the script randomly selects 10 mobile web apps from each group using a pseudo-random number generator[5] with a seed of 42. This process completes with a total of 20 mobile web apps (each app is used twice during the experiments, once as the cluttered version and once as the de-cluttered version). The selected mobile web apps and their information is presented in Table 2.

4.3 Experimental Variables

In our study, the *cluttering state* is regarded as the independent variable. The independent variable is both cluttered and de-clutterd once for each selected subject. The *energy consumption (in joules)* is regarded as the dependent variable which is measured in the performed experiments. The energy consumption is calculated by multiplying the *power consumption (in watts)* and the *PLT (in seconds)*.

To reduce the effect on the results of the page size, the size categories introduced in Section 4.2 are used as a blocking variable. Next, to reduce the effect of co-factors such as the browser type, network type, and device on the results, these co-effects are fixed to a single value. Additionally, the *CPU load (%)* and *memory usage*

[2]https://majestic.com/reports/majestic-million
[3]https://github.com/up209d/ResourcesSaverExt
[4]https://github.com/AlDanial/cloc
[5]https://docs.python.org/3/library/random.html

ID	URL	Cluttered Size	De-clutterd Size	Category	OG LOC JS	OG LOC HTML	OG LOC CSS	DC LOC JS	DC LOC HTML	DC LOC CSS
0	www.sougou.com	520KB	1.3MB	Small	4031	257	15918	4031	342	16070
1	fc2.com	560KB	444KB	Small	4505	325	2210	2130	319	2210
2	pixabay.com	1.3MB	592KB	Small	9582	786	4724	5097	343	4351
3	www.cricbuzz.com	1.4MB	336KB	Small	24960	1893	3867	1	1962	3874
4	utk.edu	1.5MB	1.1MB	Small	11624	921	8506	11914	382	8506
5	www.office.com	1.7MB	1.0MB	Small	13450	1254	8119	7629	766	8119
6	www.colorado.edu	1.9MB	1.2MB	Small	11822	887	24218	10429	779	4178
7	stackexchange.com	2.1MB	1.9MB	Small	6519	1565	18531	4206	1493	18531
8	www.rt.com	2.9MB	1.7MB	Small	36237	3426	21329	9285	3306	21497
9	www.utexas.edu	2.9MB	1.7MB	Small	20826	1179	12204	4614	1046	1040
10	www.mediafire.com	3.1MB	2.1MB	Medium	36130	2680	464	19638	2817	1768
11	archive.org	3.2MB	2.5MB	Medium	51263	2117	18343	33970	2062	18343
12	www.mdpi.com	3.4MB	1.3MB	Medium	17074	4948	16062	16663	1824	14733
13	www.boston.com	3.6MB	2.3MB	Medium	36772	14830	5727	458	13810	5788
14	www.typeform.com	3.7MB	1.7MB	Medium	43485	4138	1044	4574	3967	88
15	mail.ru	3.8MB	760KB	Medium	50290	6576	7626	3200	6019	6303
16	vk.com	4.0MB	1.4MB	Medium	53866	916	20620	4013	239	16127
17	www.nokia.com	4.3MB	2.0MB	Medium	34518	3073	18619	5590	2542	1612
18	www.jhu.edu	4.5MB	2.4MB	Medium	18575	3392	25608	6690	3601	626
19	www.exoclick.com	5.5MB	4.0MB	Medium	21551	754	21893	1	798	21893
20	prezi.com	6.1MB	8.0MB	Large	29179	854	16877	11018	641	16396
21	kakaku.com	6.2MB	4.5MB	Large	76646	8370	6548	44184	5113	6548
22	github.com	6.5MB	5.7MB	Large	22363	1142	36722	22462	1107	36730
23	www.whatsapp.com	7.2MB	6.6MB	Large	158180	3962	51324	158180	2216	35910
24	wetransfer.com	7.4MB	6.5MB	Large	108487	981	15724	95836	198	15045
25	na.op.gg	7.5MB	5.2MB	Large	94260	4716	38907	47446	3509	38907
26	www.smzdm.com	8.8MB	11MB	Large	59827	8959	7733	13580	7506	7685
27	vimeo.com	9.2MB	1.2MB	Large	78081	876	9922	26484	179	4113
28	www.tistory.com	9.6MB	9.4MB	Large	23836	495	7023	20546	440	7006
29	www.worldbank.org	11MB	9.6MB	Large	56972	64834	63298	67144	9044	63228

Table 2: The subjects used during the experiments (OG = Original, DC = De-Cluttered, LOC = Lines of Code).

(MB) are measured to explain possible fluctuations in the energy consumption during the experiments.

4.4 Experimental Hypotheses

To test the research question, we have defined the following hypotheses: the null hypothesis tells that the average energy consumption for cluttered mobile web apps is equal to the average energy consumption of de-cluttered mobile web apps:

$$H_0 : \mu_{E_{cluttered}} = \mu_{E_{de-cluttered}}$$

where E is the measured energy consumption and the subscript specifies the treatment. The alternative hypothesis states that the average energy consumption for cluttered mobile web apps is not equal to the average energy consumption of de-cluttered mobile web apps:

$$H_a : \mu_{E_{cluttered}} \neq \mu_{E_{de-cluttered}}$$

4.5 Experiment Design

The experiment is performed according to the following method: for each selected subject (which are described in Section 4.2) we have each treatment applied. This entails that there is a cluttered and de-cluttered version of each subject. In this way, each block of the mobile web app size is automatically balanced (crossover design). Both versions (cluttered and de-cluttered) of the mobile web app are loaded (in random order) on a smartphone to measure the power consumption and PLT. Additionally, the CPU load and memory usage are measured to possibly further explain fluctuations in the energy consumption. Our experiment design has the structure: 1 factor - 2 treatments [17]. The cluttering state is the factor and <*cluttered, de-cluttered*> are the treatments. The energy consumption is the measured outcome of the experiment. We repeat each trial (a combination of subject and treatment) 15 times to mitigate the effect of measurement errors. To summarize, the energy consumption of each mobile web app is measured 15 times for the cluttered version and 15 times for the de-cluttered version. Hence, as there are 10 subjects in each of the three blocks <*small, medium, large*>, there are: 3 (blocks) x (10 (cluttered mobile web apps) + 10 (de-cluttered mobile web apps)) x 15 (trial repetitions) = 900 measurements.

4.6 Data Analysis

First of all, the resulting data is visualized and explored using multiple histograms and box-plots. Afterwards, we check for normality using Q-Q-plots and the Shapiro-Wilks test with $\alpha = 0.05$ to decide which statistical test can be applied to test our hypothesis.

Given the results provided by the two previous steps, we apply two-way ANOVA in the case the data is normally distributed. ANOVA is used as we are dealing with categorical factors (mobile web app size) and a numerical dependent variable (energy consumption). We use the two-way variant because the second factor we want to account for is the block <*small, medium, large*>. If the data is not normally distributed, we use Friedman's Two-Way Analysis as non-parametric test. We also have to test whether there is an interaction effect between the page size and the cluttered state, this is also done by applying ANOVA with the significance level of $\alpha = 0.05$.

For further analysis of the fluctuations in the energy consumption, we use a multiple regression test with the CPU load and memory usage as numerical independent variables and the energy consumption as dependent variable.

5 EXPERIMENT EXECUTION

In our experiment, we use an Android smartphone to load the mobile web apps. The smartphone used for the experiment is the Samsung Galaxy J7 Duo and has an octa-core (2x2.2 GHz Cortex-A73 and 6x1.6 GHz Cortex-A53) processor with 4 GB of RAM. This smartphone runs Android version 8.0.0. To mitigate the effects of background processes, we disable services we do not use such as Bluetooth, Location Services (GPS), and NFC. To ensure that the experiment stays running while minimizing the impact of the display on the measured energy consumption, the screen brightness is minimized while ensuring that the phone cannot go to sleep mode. Furthermore, we disabled all push notifications and non-critical apps.

The smartphone is connected to a laptop using Android Debug Bridge (adb)[6]. The computer that we use to manage the experiment has an Intel i7 CPU (8th generation, 8 cores, and 4.2 GHz maximum clock speed) with 16 GB of RAM. Adb is a tool that allows communication between a development machine and an Android device. This communication entails information of the experiment configurations from the laptop to the smartphone and measurement data from the smartphone to the laptop. The Android smartphone and laptop are both connected to the same LAN. All the devices are in close proximity to the WiFi router (802.11ac standard and 5 GHz frequency band) to ensure minimum latency. The Android smartphone uses the Google Chrome web browser version✎ X to access the internet and load the cluttered and de-cluttered mobile web apps. The researchers of the JSCleaner paper [6] cached the cluttered and de-cluttered mobile web apps, which they used for their research, on a proxy server and shared these details with us. In order to access the proxy server, the mitmproxy[7] certificates should be installed on the smartphone. Mitmproxy is an HTTPS proxy that allows (among other features) interactive HTTPS requests.

In order to manage the execution of our experiment, we use Android Runner (AR)[8]. AR is a tool that facilitates the execution of measurement-based experiments on native apps and mobile web apps running on Android devices [18]. AR allows us to fill in the experiment configurations and setup after which it communicates this to the smartphone. After the experiment is conducted, AR stores the recorded measurement data on the laptop. AR utilizes Monkey Runner[9] to implement an API that grants control over the Android smartphone. We use Monkey Runner to construct a log of smartphone input commands. This log can be used to automate the interactions with the smartphone. Lastly, we use Trepn[10] to measure the battery power (watts), memory usage (MB), and CPU load (%). Trepn is a profiler and plugin of AR. Trepn utilizes the Trepn Android app[11] which we downloaded and ran on the smartphone.

A high-level view of our experiment execution is visualized in Figure 2. Both the *Android Smartphone* and the *Laptop* are connected through the same *LAN*. The *LAN* is connected to a *modem* that sends requests to the *Proxy Sever* that has access to the cached *cluttered and de-cluttered mobile web apps*. The *Android Smartphone* and the *Laptop* are connected through *adb*. The *Laptop* runs the *Android Runner* software which manages the experiment. *Android Runner* allows us to define the experiment setup. *Android Runner* utilizes the *Monkey Runner* software which allows to store phone interactions so that the experiment can be fully automated. Moreover,

[6]https://developer.android.com/studio/command-line/adb
[7]https://docs.mitmproxy.org/
[8]https://github.com/S2-group/android-runner
[9]https://developer.android.com/studio/test/monkeyrunner
[10]https://github.com/S2-group/android-runner/tree/master/AndroidRunner/Plugins/trepn
[11]https://github.com/S2-group/android-runner/blob/master/AndroidRunner/Plugins/trepn/com.quicinc.trepn.apk

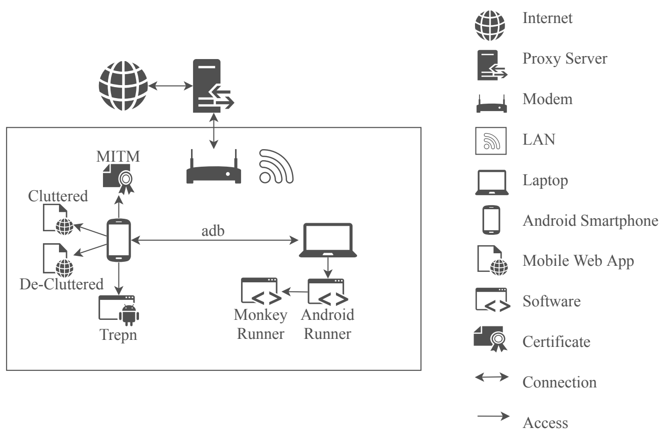

Figure 2: Experiment Execution

Android Runner records the measurement through the *Trepn* profiler. The *Android Smartphone* has the Android native *Trepn* application installed. This application performs the actual measurements of the battery power, memory usage, and CPU load. Furthermore, the *Android Smartphone* has the *MITM Certificate* installed to access the *Cluttered and De-Cluttered Mobile Web Apps* through the *Proxy Server*.

In each trial, AR ensures that the Google Chrome browser is opened on the smartphone. Afterwards, the URL of a mobile web app is inserted and loaded. Trepn measures the battery power, memory usage, and CPU load of this process. Next, this data is communicated through adb and stored using AR. After each run, the Google Chrome browser is cleared and closed. Then, the smartphone remains idle for 1 minute to ensure that all processes are finished. To guarantee the intrinsic variability of the experiment, each trial is repeated 15 times. We repeat this measurement for 10 randomly selected cluttered mobile web apps and their de-cluttered version.

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] T. Dale. Javascript frameworks and mobile performance. [Online]. Available: https://tomdale.net/2015/11/javascript-frameworks-and-mobile-performance/

[3] M. Persson, "Javascript dom manipulation performance: Comparing vanilla javascript and leading javascript front-end frameworks," 2020.

[4] J. Ambriz. Front-end frameworks: Solutions or bloated problems? [Online]. Available: https://www.toptal.com/javascript/are-big-front-end-frameworks-bad

[5] R. N. Mayo and P. Ranganathan, "Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down," in *International Workshop on Power-Aware Computer Systems*. Springer, 2003, pp. 26–40.

[6] 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

[7] 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

[8] 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

[9] X. Chen, Y. Chen, M. Dong, and C. Zhang, "Demystifying energy usage in smartphones," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2014, pp. 1–5.

[10] Google. Amp on google. [Online]. Available: https://developers.google.com/amp

[11] W. Craig. A modern approach to improving website speed. [Online]. Available: https://www.webfx.com/blog/web-design/improve-website-speed/

[12] 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

[13] 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.

[14] 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.

[15] 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

[16] M. H. M. C. O. B. R. A. W. Claes Wohlin, Per Runeson, "Experimentation in software engineering." Springer US, 2012.

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

[18] 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