✎ What is the impact of memoization on the energy efficiency of mobile web apps?

Markus Funke
2644722
VU Amsterdam
m.t.funke@student.vu.nl

Sven Preng
2614131
VU Amsterdam
s.preng@student.vu.nl

Pjotr Scholtze
2612369
VU Amsterdam
p.scholtze@student.vu.nl

Wouter Kok
2639768
VU Amsterdam
w.j.kok@student.vu.nl

Yaping Ren
2662078
VU Amsterdam
y2.ren@student.vu.nl

ABSTRACT

Context. brief explanation of the motivation for conducting the study
Goal. aim, objects, focus, perspective of the study (based on the GQM)
Method. experimental design, number and kind of objects/subjects, selection criteria, data collection and analysis procedures
Results. main findings
Conclusions. impact of the obtained results

ACM Reference Format:
Markus Funke, Sven Preng, Pjotr Scholtze, Wouter Kok, and Yaping Ren. 2020. ✎ What is the impact of memoization on the energy efficiency of mobile web apps? . 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

Smartphone users are growing worldwide at an exceedingly fast speed. The number has grown by 1 billion in the past five years and is reaching 3.5 billion this year [1]. The market growth stimulated the rapid development of technologies on the mobile end. As a result, more users are accessing internet services with their portable devices.

✎ Mobile device users can easily get access to mobile web apps by only having a browser installed. These mobile web apps create the possibility of writing cross platform apps, which can be written in HTML, CSS, and JavaScript and distributed via the standard web protocols like HTTP. However, there are some signs that mobile web apps' performance is not comparable to native apps, for example, not being able to handle the same graphics in 3d gaming [2]. These performance issues which might be due to inefficiencies, could be related to power consumption.

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

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

With all the merits of ✎ mobile web apps, energy consumption also comes to concern due to the scarcity of battery power on mobile devices. Researching on energy efficiency contrasting different code design patterns could shine light on the trade-offs when making design decisions.

Previous research has shown that a positive correlation between performance and energy consumption exists [3], this raises the question of which previous work on performance improving patterns also improves energy efficiency. One of the possible techniques shown to increase performance is memoization [4], which is a technique from the 1960s [4, 5]. Memoization works by caching the output of a function and returning it if it already knows the return value. An example of a non-memoized function can be seen in fig. ✎ update figure pos 1 and the same function memoized can be seen in fig. 2. Fig. 1 line 1, shows the function declarations which accepts one argument called *n*. Line 2 shows the stopping condition, which is 1 for which it will return 1;. Line 3 shows the recursive part of the calculation, making sure *n* will be multiplied by the result of $n - 1$ called on the same function. Fig 2 shows the memoized function, line 1 shows the function declaration, which accepts two arguments, *n* and *memo*, *n* is the factorial number and *memo* the memoized results. Line 2 shows the stopping condition, which is 1 for which it will return 1;. Line 3 shows ensures the memo is an object. Line 4 checks if the result of the calculation already exists, and line 5 calculate this number. Line 7 returns the result of this part of the calculation.

```
1 function factorial(n) {
2     if (n == 1) return 1;
3     return n * factorial(n - 1);
4 }
```

Figure 1: Non-memoized pure JavaScript function example

Memoizing a function with a state ✎ similar to object-oriented programming is typically more challenging than those without state [6]. An example of functions without a state are pure functions. Pure functions are functions in functional programming, which can be called with zero or more arguments and will always return the same result without any side effects or change in state [7]. Therefore, the focus of this work is on the memoization of pure functions.

```
function factorial(n, memo) {
    if (n == 1) return 1;
    memo = memo || {};
    if (!memo.hasOwnProperty(n)) {
        memo[n] = n * factorial(n - 1);
    }
    return memo[n];
}
```

Figure 2: Memoized pure JavaScript function example

This research **aims to** investigate the impact of memoization on the energy efficiency level of pure JavaScript functions on web Apps. ✎ To accomplish this, we plan to research the influence of memoization on memory usage and energy efficiency level. An empirical study is conducted on the energy efficiency of memoized pure JavaScript functions on different devices with a ✎ custom web App. We avoid manual decision-making and automate the selection of pure JavaScript functions in real-world mobile web apps.

The **main contributions** are: (i) a statistical analysis on the energy efficiency of memoized pure JavaScript functions compared to their not-memoized counterpart, (ii) a package for replicable experiments including the ✎ custom web App, test functions, and the R script used for analysis.

The **target audience** of this work consists of two groups, ✎ mobile web app developers and researchers. The ✎ mobile web app developers can be informed about the impact of memoization patterns on energy consumption despite the space-time trade off for ✎ the mobile web apps. As for researchers, we could provide more insights into the energy efficiency of memoization in the context of ✎ mobile web apps with real-world JavaScript functions.

The experiment in this work is conducted according to the guidelines by Wohlin and colleagues [8].

2 RELATED WORK

Recent research displays increasing interest in energy efficiency and performance on mobile web apps as well as progressive web apps [9, 10]. The current landscape of research in this area will be analyzed in order to advance upon it with accurate and useful experiments.

Prior research has shown a negative correlation between performance scores and energy consumption in mobile web apps [9]. When measured performance scores improve, it tends to lower energy consumption. Therefore it is important to see how memoization impacts the performance of mobile web apps.

There are many ways developers can write better code to increase performance. In a study on optimizations in JavaScript, Selakovic et al. show an extensive list of causes of performance issues from 16 popular client-side and server-side JavaScript projects [11]. From all issues, 13% are related to repeatedly executing the same operations. Memoization can be a solution to this issue.

Moreover, memoization in real-world Java programs has been evaluated by Della et al. [12]. A dynamic analysis tool called MemoizeIt is used to detect Java functions that showed the best opportunity to memoize. The memoized functions showed statistically significant speedups, from which we can hypothesize that memoization might lead to increased performance, which can lead to lowered energy consumption.

In a similar study, Rua et al. suggest that memoization can positively influence energy savings in Android [13]. After manually analyzing Android apps to find pure Java functions, a list of functions was assembled to test. The results indicate a decreased energy consumption in 13 out of 18 tested methods. Our study aims to find the impact of memoization on the energy efficiency of mobile web apps, therefore our scope is different and can help provide additional information.

Lastly, Malavolta et al. did a study on real-world PWAs to see the impact of caching on energy consumption and performance showing positive results [10]. The study indicates that PWAs can reduce energy consumption and boost performance when caching web pages. Noticeably, memoization is a type of caching [13]. It might be interesting to compare the differences between caching in PWAs and caching of mobile web apps using memoization on energy consumption.

3 EXPERIMENT DEFINITION

In this section, the Goal Question Metric (GQM) method from Basili (1992) is used to express the research goal, the questions, and the used metrics [14]. Figure 3 shows the graphical representation of the GQM model and is followed by a textual description in the subsequent sections.

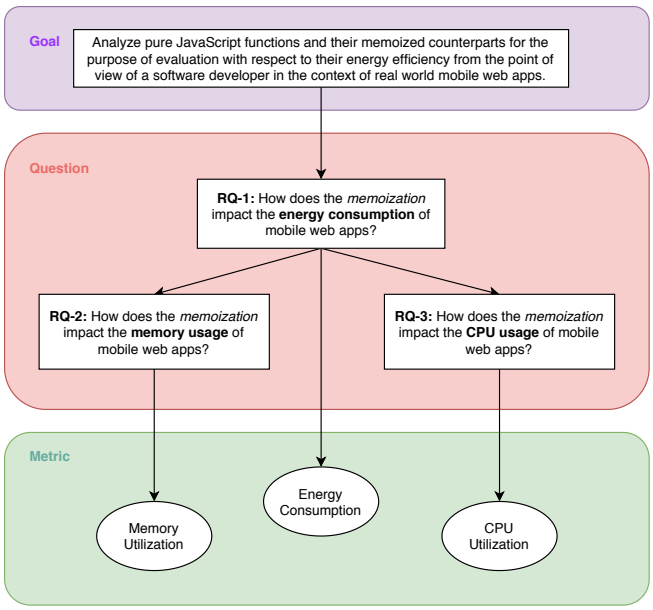

Figure 3: GQM Model

3.1 Goal

Analyze **pure JavaScript functions and their memoized counterparts** for the purpose of **evaluation** with respect to their **energy efficiency** from the point of view of **a software developer** in the context of **real-world mobile web apps**.

3.2 Question

RQ-1: *How does memoization impact the energy consumption of mobile web apps?*
For the experiment, real-world applications are analyzed for their JavaScript functions. To do so, we are going to mine as many JavaScript functions as possible and check these functions for their purity.

RQ-2: *How does memoization impact the memory usage of mobile web apps?*
In extend to the main research question RQ-1, we will also analyze the memory usage of the same mined JavaScript functions.

RQ-3: *How does memoization impact the CPU usage of mobile web apps?*
In extend to the main research question RQ-1, we will also analyze the CPU usage of the same mined JavaScript functions.

3.3 Metric

- **Energy consumption**: We are curious about the energy consumption of memoized functions and functions that are not memoized. Energy consumption can be measured during runtime of the application. The result will be the energy consumption measured in ✎ Joule and is calculated with the ✎ formula $Energy = PowerxTime$.
- **Memory utilization**: As for Memory utilization, memoization makes a difference in memory caching and thus this is a factor that could play a role in energy consumption. Memory utilization can be measured during runtime of the application. The result will be the memory utilization measured in ✎ kilo bytes (KB).
- **CPU utilization**: As for CPU utilization, memoization makes a difference in computation and thus this is a factor that could play a role in energy consumption. CPU utilization can be measured during runtime of the application. The result will be the CPU utilization measured in ✎ percentage (%).

4 EXPERIMENT PLANNING

4.1 Subjects Selection

The JavaScript functions, which are the subjects of the experiment, are mined from real websites. The *Tranco* list is used to get the *1508* most popular sites for this procedure. To save these websites to disk, a *Python* script is used with *Selenium* and *PyAutoGUI* after which the *JavaScript* files are processed with the node library *js-beautify*. The non-pure lines are removed with the node tool *purecheck*. All that remains are *4915* functions. From these functions *50* functions with parameters are hand-picked, and *50* functions with no parameters with no dependencies are randomly selected. Resulting in *100* functions for testing. Memoization is manually applied to these functions with a bit of custom code.

4.2 Experimental Variables

We identified the **memoization status** of JavaScript functions as independent variable. The selected non-memoized JavaScript functions are converted to memoized versions with a JavaScript library. Therefore all the functions are processed with the same method.

For each run of the experiment, we run both the memoized functions and non-memoized functions 1000 times to make sure the memoization takes effect, and record the dependant variables separately. After each function, the application environment (e.g. Google Chrome cache) is cleared.

We identified dependent variables for each of our research questions as follows:

- **Energy consumption**: We use Android runner to profile the energy consumption for the run time of our mobile app on Google Chrome. Android runner can then return data points of battery power with the set intervals with Joules as the measuring unit.
- **Memory usage**: The plugin men-CPU in Android-Runner provides a performance measurement by collecting memory usage via meminfo in ADB's dumpsys.

4.3 Experimental Hypotheses

Two hypotheses are provided to be able to answer the research questions of this study; where B is the battery power, M is the memory usage, and C is the CPU usage:

The first null hypothesis is tested for the first research question (RQ-1) to see if the average battery power consumed is the same at both memoized and non-memoized functions. The alternative hypothesis here is that they are not the same.

$$H_0^1 : \mu_{B_{memoized}} = \mu_{B_{non-memoized}} \quad (1)$$

$$H_a^1 : \mu_{B_{memoized}} \neq \mu_{B_{non-memoized}} \quad (2)$$

The second null hypothesis is tested for the second research question (RQ-2) to see if the average memory usage is the same at both memoized and non-memoized functions. The alternative hypothesis here is that they are not the same.

$$H_0^2 : \mu_{M_{memoized}} = \mu_{M_{non-memoized}} \quad (3)$$

$$H_a^2 : \mu_{M_{memoized}} \neq \mu_{M_{non-memoized}} \quad (4)$$

The third null hypothesis is tested for the third research question (RQ-3) to see if the average CPU usage is the same at both memoized and non-memoized functions. The alternative hypothesis here is that they are not the same.

$$H_0^3 : \mu_{M_{memoized}} = \mu_{M_{non-memoized}} \quad (5)$$

$$H_a^3 : \mu_{M_{memoized}} \neq \mu_{M_{non-memoized}} \quad (6)$$

4.4 Experiment Design

The experiment design is described based on the checklist on chapter 5.9 from Basics of software engineering experimentation [15].

Step 1. Factors
We identified the factor *Pure JavaScript Function*. This will be further discussed in section 5. Beyond, we have identified treatment 1) as memoized and treatment 2) as non-memoized functions.

Step 2. Response variables
The response variables that are likely to be influenced by memoization are the energy consumption during runtime and memory utilization during runtime.

Step 3. Parameters
One parameters that is important for this experiment is the ✎ chrome version of the Google Chrome browser. Another parameter is the operation system of the phone which is ✎ Android version.

Step 4. Blocking variables
For this experiment, we do not have any blocking variables.

Step 5. Replications
✎ 30

Step 6. Experimental design
✎ As statistical test we have chosen the paired t-test. This test was chosen, since our experiment is based on measurements, i.e. energy consumption in Joule and memory usage in ✎ Bytes, and we have two paired sets of measurements which we want to examine according their differences.

Step 7. Experimental objects
The experimental objects are pure JavaScript functions mined from Tranco; a ranked list of sites used for research. This will be further discussed in section 5.

Step 8. Experimental subjects
The authors of the paper will be running the experiments.

Step 9. Data collection process
✎ Ivano: This is about how you are going to measure (i.e., to collect the data about your dependent variables).

4.5 Data Analysis

✎ Ivano: Remember to add also Effect size estimation here. As discussed beforehand, the analyzed subjects are pure JavaScript functions. The data analysis part is also explained in figure 4 in a high level manner.

Exploration.
✎ Ivano: Easy to collect and show data points might be: - lines of code for each function - number of input parameters (remember that memoization is strongly dependent on the parameters of a function since it needs to store them) - total number of functions To better understand the data distribution as a base for the following analysis, we start with boxplots for more intuitive outlier detection. Then histograms for the shape of the gained data. We plan to explore energy consumption, memory usage, and CPU utilization.

Normality assessment.
In the next phase we are going to assess whether the data that we have collected is normally ✎ Ivano: check also the other assumptions that your statistical test will have. distributed. It is important to adjust the statistical methods used based on the normality of the data. We will assess the normality of the gained data by means of a Q-Q plot and a Shapiro-Wilk test. Parametric tests will be followed when we confirmed the normal distribution of data.

Hypothesis testing.
✎ To assess the influence of memoization on the subjects, we are going to use a paired t-test. This test allows us to see if there is a significant difference between the subjects based on the different treatments they got. We are going to filter the data to remove outliers if necessary. That is, only if the data is continuous which is assumed. The observations are independent ✎ cite. Our experiments only have one factor (memoization status of JavaScript functions) and two treatments (memoized functions and non-memoized functions). Therefore we will conduct a paired t-test for each research question to compare the population means before and after memoization. A paired t-test is selected because the measurement of dependant variables is taken before and after memoization. Therefore they are dependant. The paired t-test will show whether there is a significant difference in energy consumption (or memory usage) between the same memoized and non-memoized JavaScript function group.

5 EXPERIMENT EXECUTION

✎ Markus Update

This section gives an overview of the execution of the experiment and examines the different stages. The experiment can be divided into three different stages, i.e., (i) data mining, (ii) experiment execution, and (iii) data analysis. Figure 4 illustrates the experiment in a high level manner and is discussed in the following subsections.

5.1 Data Mining

The data selection is based on the Tranco list[1] with ✎ one million domains which were selected on different criteria [16]. Since this research's subject are JavaScript functions, it is necessary to mine as many JavaScript functions as possible. The complete mining process is conducted by a ✎ virtual Xubuntu Linux machine on a stand-alone PC with a wired internet connection. By making use of Selenium[2], an automated process iterates over all available domains of the Tranco list and stores the complete website with all its resources temporally on disk, such that the website is accessible offline. As described in section 3, this research focuses only on pure JavaScript functions. Therefore, a node.js instance is used to perform several tasks on each saved website. Firstly, all JavaScript functions are extracted from one particular website. Secondly, js-beautify[3] is used to reformat minimized JavaScript files and ensure that they are readable. Lastly, purecheck[4] separates 'normal' JavaScript functions from pure JavaScript functions. The output of the data mining process is a single JavaScript file with pure functions. The procedure filtered from the initial one million domains, ✎ XX usable websites with ✎ XX JavaScript functions in total, whereas ✎ XX actually pure JavaScript functions were and suitable for this experiment.

5.2 Experiment Execution

The main component of the experiment is based on Android Runner (AR) to automatically executing measurement-based experiments on native and web apps are running on Android devices [17]. This study uses AR to collect measures of the (i) energy consumption and (ii) memory usage, as described in section 3. moize[5] is used as

[1] https://tranco-list.eu (Accessed: 2020-09-25)
[2] https://www.selenium.dev (Accessed: 2020-09-25)
[3] https://www.npmjs.com/package/js-beautify - v1.13.0 (Accessed: 2020-09-25)
[4] https://www.npmjs.com/package/purecheck - v1.2.0 (Accessed: 2020-09-25)
[5] https://www.npmjs.com/package/moize - v5.4.7 (Accessed: 2020-09-25)

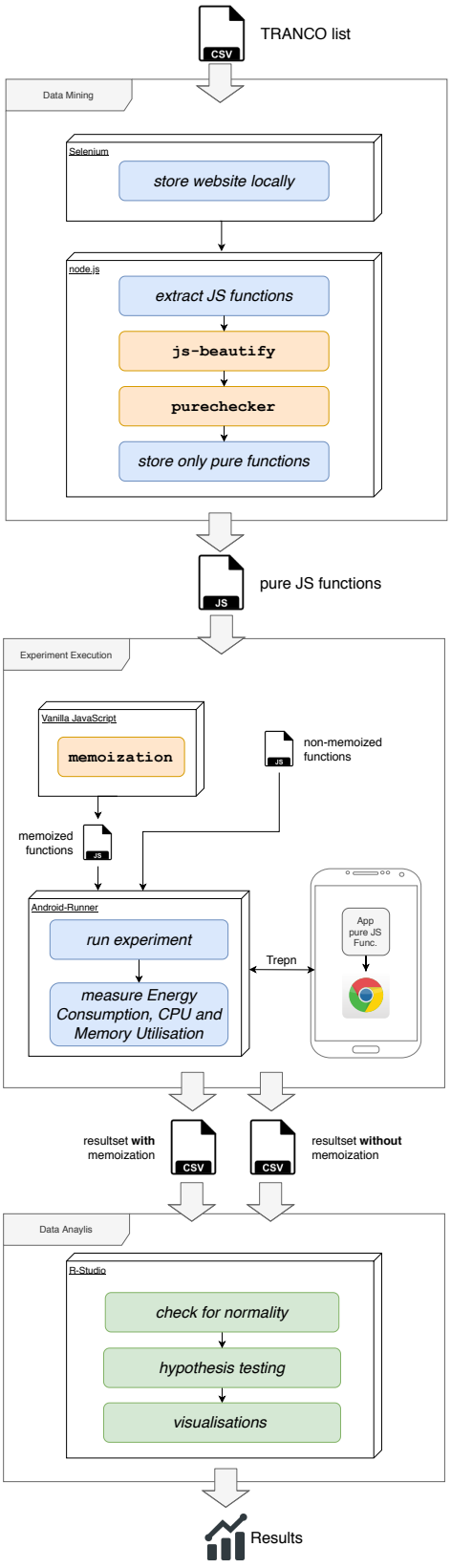

Figure 4: Experiment Architecture and Process

memoization library to handle the transformation of 'normal' pure JavaScript functions into memoized functions. These two groups of measurements serve as input for the AR experiment. The complete experiment is also executed by a stand-alone PC. AR is connected to a laboratory Android phone ✎ XY which runs the JavaScript functions in a dedicated developed web application and is executed by Google Chrome. The output of the experiment execution includes two different result-sets; measures with the applied memoization technique and measures without memoization.

6 RESULTS

6.1 Exploratory Data Analysis

-

6.2 RQ-1: How does memoization impact the energy consumption of mobile web apps

-

6.3 RQ-2: How does memoization impact the memory usage of mobile web apps?

-

6.4 RQ-3: How does memoization impact the CPU usage of mobile web apps

-

Provide:

- descriptive statistics
- hypothesis testing

Provide suitable plots and tables to illustrate your results.
Page limit: Open - go deep as you wish

7 THREATS TO VALIDITY

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

7.1 Internal Validity

✎ Markus

7.1.1 History. Different trials of the experiment performed in different time frames (eg, after holidays vs normal days)

The time you do the experiment, should not have an influence on the experiment/result;

-> the time frame is the same

7.1.2 Maturation. Subjects may react differently over time (eg, learning effect, tiresome, boredome)

-> not applicable, we reset, we split the experiment in memoized, non-memoized; but there is no cache or something since we reset everything, so we clean everything

-> how do we keep our experiment under control; reliable; (closed PC setup without any different usage during the experiment; phone is cleaned; no Wifi or connection since everything is loaded from local file system) -> do we use apps? -> what is our usage scenario? -> reliability: we are doing repetitions over the same combination

of subjects/treatments/objects; we can measure reliability by discussing standard derivation or coiffent derivation the data we are collection trial by trial; if our data are tight falling into a certain level of dispairtion? -> then our data are reliability

7.1.3 Selection. Some subjects may abandon the experiment; Event worse, some specific type of subjects may leave it

-> we are looking at some servers? no; everything is offline; no subjects can be abandon

7.1.4 Reliability of measures. If you repeat the measurement you should get similar results -> same conclusions

-> what are we doing in the project -> do we repeat the experiment? -> if we repeat the experiment do we come to the same results? I think so

7.2 External Validity

7.3 Construct Validity

7.4 Conclusion Validity

Page limit: 1

8 DISCUSSION

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

Elaborate on the obtained results; impact into practice (industry); Make the results actionable (how will app developer John use your results tomorrow?; what researcher Y learned about phenomenon X?); Lessons learnt; No abstract findings and number;

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] P. b. S. O'Dea and A. 20, "Smartphone users worldwide 2020," Aug 2020. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[2] I. Malavolta, "Beyond native apps: web technologies to the rescue!(keynote)," in *Proceedings of the 1st International Workshop on Mobile Development*, 2016, pp. 1–2.

[3] G. Pinto, F. Castor, and Y. D. Liu, "Mining questions about software energy consumption," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 22–31.

[4] M. Hall and J. P. McNamee, "Improving software performance with automatic memoization," *Johns Hopkins APL Technical Digest*, vol. 18, no. 2, p. 255, 1997.

[5] D. Michie, ""memo" functions and machine learning," *Nature*, vol. 218, no. 5136, pp. 19–22, 1968.

[6] H. Rito and J. Cachopo, "Memoization of methods using software transactional memory to track internal state dependencies," in *Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java*, 2010, pp. 89–98.

[7] M. Hofmann, A. Karbyshev, and H. Seidl, "What is a pure functional?" in *International Colloquium on Automata, Languages, and Programming*. Springer, 2010, pp. 199–210.

[8] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2012.

[9] 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," in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 190–199.

[10] I. Malavolta, K. Chinnappan, L. Jasmontas, S. Gupta, and K. A. K. Soltany, "Evaluating the impact of caching on the energy consumption and performance of progressive web apps," in *7th IEEE/ACM International Conference on Mobile Software Engineering and Systems 2020*, 2020.

[11] M. Selakovic and M. Pradel, "Performance issues and optimizations in javascript: an empirical study," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 61–72.

[12] L. Della Toffola, M. Pradel, and T. R. Gross, "Performance problems you can fix: A dynamic analysis of memoization opportunities," ser. OOPSLA 2015. Association for Computing Machinery, 2015, p. 607–622. [Online]. Available: https://doi.org/10.1145/2814270.2814290

[13] R. Rua13, M. Couto13, A. Pinto24, J. Cunha14, and J. Saraiva14, "Towards using memoization for saving energy in android," 2019.

[14] V. R. Basili, "Software modeling and measurement: the goal/question/metric paradigm," Tech. Rep., 1992.

[15] N. Juristo and A. M. Moreno, *Basics of software engineering experimentation*. Springer Science & Business Media, 2013.

[16] V. L. Pochat, T. van Goethem, and W. Joosen, "Rigging research results by manipulating top websites rankings," *CoRR*, vol. abs/1806.01156, 2018. [Online]. Available: http://arxiv.org/abs/1806.01156

[17] 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