
Bogdan Andrei  
Vrije Universiteit Amsterdam  
The Netherlands  
b.andrei@student.vu.nl

Ivano Malavolta  
Vrije Universiteit Amsterdam  
The Netherlands  
i.malavolta@vu.nl

ABSTRACT

Context. Many mobile web apps are using CSS vendor prefixes today. The presence of CSS prefixes might impact the energy consumption and performance of mobile web apps.

Goal. The goal of this research is to empirically assess the impact of using CSS prefixes on the energy consumption and performance of mobile web apps.

Method. To assess the impact of CSS prefixes, we mine 25 real mobile web apps and create two variants for each of them, one with CSS prefixes and one without. Then, we load the mobile web apps on two Android devices across three different browsers (i.e., Google Chrome, Mozilla Firefox, and Opera). For each run of the experiment, we measure the energy consumption, page load time, and CPU usage of the targeted browser.

Results. CSS prefixes do not lead to an observable impact on the energy consumption of the measured Web apps, while their absence leads to a statistically significant increase of CPU usage in all browsers. The presence of CSS prefixes has a statistically significant negative impact on the loading time in all browsers and devices.

Conclusions. Assuming that the targeted mobile web app loads properly without CSS prefixes, we advise web developers to remove CSS prefixes if their goal is to reduce page load time. We collect and discuss strategies about how to safely remove CSS prefixes without incurring compatibility issues. The choice of whether to use CSS vendor prefixes should be made with consideration of the targeted devices and browsers, especially when dealing with energy consumption and CPU usage.

ACM Reference Format:

1 INTRODUCTION

The development of front-end technologies has been flourishing since W3C (World Wide Web Consortium) released HTML5 in 2008 and a major update as well as “W3C Recommendation” status in 2014. From coding languages such as HTML, and JavaScript to web frameworks such as Angular, Vue, and React, developers have been trying to optimize the technologies to keep pace with the times. In recent years, climate change has become a huge topic in the IT industry, and the performance and energy efficiency of hardware and software are more important than ever. For front-end developers, a lot of them contribute their effort in optimizing server implementation, caching, code quality, etc., because they are the larger and more complicated problems; which may result in greater optimization [1].

However, a technology as simple as Cascading Style Sheets (CSS) can also have a significant impact on web performance. When the browser starts processing and rendering CSS after HTML is parsed, a crucial step is to compute the values based on HTML and CSS properties to generate the final view of the web application. Vendor prefixes are part of CSS files, they function as a bridge to provide a method for different browsers to add support for new or experimental CSS features before those features are fully supported in all browsers. The main reason browsers use these prefixes is that it would take time to pass new features to W3C for standardization, and the companies wish to proceed with their experiments and test the new features right away due to development time constraints and other practical matters. CSS vendor prefixes being used today include: -webkit- (Chrome, Safari), -moz- (Firefox), -ms- (Edge, Internet Explorer), and -o- (Opera) [2]. Below we provide an example of the same CSS statement with various types of CSS vendor prefixes.

```css
-width: 100%;
-webkit-width: 100%;
-moz-width: 100%;
-ms-width: 100%;
-o-width: 100%;
transition: all 2s ease;

In addition to the original code, there are more almost identical lines of code added to it. Not only the CSS code becomes more verbose and difficult to maintain for developers, but also it adds up more computations for the rendering process when the web app attempts to resolve conflicting CSS declarations and process final CSS values before displaying the resulting web app.

The goal of this study is to assess the performance and energy consumption when loading web apps either with or without CSS vendor prefixes. To achieve this goal, we dive deeper into the exploration by utilizing an Android-based mobile device as well as a Raspberry Pi Model 4B to measure the load time and energy consumption differences of the three most commonly used browsers (Chrome, Firefox, and Opera) when loading various popular web apps; we will discuss in detail in Section 4. Other operating systems, for instance, iOS, will not be discussed in our work. We will also study related literature for inspiration and possible connections.
The main results of this study indicate that the presence of CSS prefixes has a negative impact on the loading time of the measured Web apps across all three browsers for both Android devices. Differently, the presence of CSS prefixes did not lead to a statistically significant impact in terms of energy consumption. Also, the presence of CSS prefixes tends to increase the CPU usage of the Web apps (except for Firefox and Opera running on the Google Pixel 5 device). Therefore, while CSS prefixes can be optimized for adapting websites to different browsers, developers might consider abandoning such a technique if the optimization of loading time must be prioritized in their project.

The main contributions of this study are (i) an empirical assessment of the impact of using CSS prefixes on the energy consumption and performance of mobile Web apps executed on 2 Android devices and 3 different browsers, (ii) a discussion of the obtained results, and (iii) the full replication package [3] of the study containing raw data, source code, and scripts for data analysis.

2 RELATED WORK

To the best of our knowledge, this is the first study focussing on the overhead of CSS prefixes.

Adrian et al. [4] proposed a tool named WebChar for analyzing the performance and hidden energy problems of HTML and CSS with several browsers. WebChar uses HTML and CSS content to build a static page feature model of browser-related performance and energy consumption. The timing and power data of the Web browser are measured when each page is loaded. The HTML files and associated CSS abstract syntax trees are transformed into features, the model is trained, and a simulated annealing method is used to find high-energy consumption feature vectors. By analyzing 200 websites in two systems, a netbook, and an Android device, the experiments validated performance flaws in 8 components, such as images and CSS descendant selectors. Although they focus on CSS stylesheets, they still do not address browser prefixes.

Yun Ma et al. [5] explored the difference between the performance of 328 services on the Android platform and native apps by running selected web pages on Chrome and measuring the device’s traffic using the Wireshark tool. This experiment set four research metrics to quantify the performance, including the number of requests, response time, data consumption, and energy consumption. The results show that in more than 31% of cases, web pages perform better. Besides, in their opinion, the performance of the application is also related to the user experience. Only the four previously mentioned factors were considered. However, they did not focus on measuring the performance of vendor prefixes, our experiments aim to analyze the energy-performance considered objective metrics. The CPU performance of many mobile devices differs from these studies in that we focus on the possible changes in energy consumption and performance due to CSS, leaving aside factors such as network latency. Our research mainly focuses on evaluating the performance and energy consumption of vendor prefixes on three web browsers (Chrome, Firefox, and Opera) in a stable network environment.

3 EXPERIMENT DEFINITION

The goal of this study is to Analyze the impact of the vendor prefixes to evaluate the impact on energy consumption and performance from the point of view of Web developers in the context of Android mobile web browsing. To achieve this goal, we answer the following research questions:

RQ1 - What is the impact of the vendor prefixes on the energy consumption of Android mobile web browsing?

RQ2 - What is the impact of the vendor prefixes on the performance of Android mobile web browsing?

The measurement of energy consumption is done by using software estimates of Android devices’ energy consumption in Joule (J), during the experiments. When browsing specific websites using an Android mobile device on 3 different browsers (Chrome, Firefox, and Opera), the battery stats will be recorded using the BatteryManager plugin of Android Runner [9].

To measure the performance of a web app, its page load time is calculated in milliseconds (ms), as well as, the CPU usage, in percentage (%) when accessing a web page using on 3 different browsers (Chrome, Firefox, and Opera).

Since the main goal is to evaluate the impact of vendor prefixes on the energy consumption and performance of Android mobile web browsing, the main target audience of this study consists of...
Web developers, who might have an interest in knowing the exact influence of using vendor prefixes is when they are implementing them in their websites.

4 EXPERIMENT PLANNING

4.1 Subjects Selection and Context

A sample of 25 websites is selected from the Tranco list\(^1\) to evaluate the energy consumption of the vendor prefixes technology in three different browsers. To successfully clone a website, two important aspects must be met: the website’s index.html file must not be directly connected to a database and upon downloading process, no "failed to download resources" must be present in the log of the ResourceSaver\(^2\). The 25 web apps selected as subjects for our experiment cover as many areas as possible, because the browsing preference of everyone’s daily life is not the same. Our subjects belong to the following categories: general (5 subjects, e.g., Wikipedia, Espn), e-commerce (5 subjects, e.g., Target, Shopify), consumer services (5 subjects, e.g., FourSquare, Trulia), communities (5 subjects, e.g., Facebook, YouTube), finance (5 subjects, e.g., Bloomberg, Forbes). After this step, the selected sites are technically analyzed for whether they use CSS vendor prefixes or not, i.e., if they have at least one `<style>` tag in their index.html file.

In this experiment, testing on different devices is considered first, because of the different Android versions of each device and technical variability. To obtain accurate data for comparison in the results of the tests on different devices, we use only two Android devices. Indeed, a wide range of Android devices in use in the market, for example, mobile devices and tablets that have Android as their operating system are very different. Even though different mobile device manufacturers now use the same version of Android, these small differences can lead to uncertainty in experimentation due to the depth of customization of the system. The most important point is that even if the same version of Android is used, different mobile devices have different hardware such as CPU, RAM, ROM, etc. which have an impact on the results. At the same time, with the current development of mobile network technology, the results of experiments with and without 5G-enabled mobile devices will also have a significant impact. Therefore it is decided to use two Android devices: a Galaxy J7 Duo with Android version 8.0.0 and a Google Pixel 5 with Android version 11.0.0.

As of August 2023, the market share of browsers on mobile platforms worldwide is as follows: Chrome (58.16%), Safari (35.49%), Samsung Internet (4.02%), Opera (1.09%), YaBrowser (0.43%) and Others (0.81%) including Firefox (0.47%)\(^3\). Firefox had a 3.15% market share across all operating platforms in the past two years. In addition, inspired by the work of de Macedo et al. [10] where they compared the energy consumption of Chrome and Firefox as the two most popular browsers, we decided to also compare these two browsers. Besides, since our topic is related to vendor prefixes, as mentioned in Section 1, Opera has different vendor prefixes from Chrome and Firefox, and although not as large as Chrome and Firefox, Opera has its user group, for which we think Opera is also valuable to be tested.

4.2 Experimental Variables

First, to conduct a controlled experiment, we divide the experiment into two cases, using vendor prefixes and removing vendor prefixes. Therefore, the use of vendor prefixes becomes an independent variable that we selected. At the same time, the type of browser becomes a blocking factor, with three treatments: Chrome, Firefox, and Opera.

Table 1: Dependent variables

<table>
<thead>
<tr>
<th>Dependent variables</th>
<th>Description</th>
<th>RQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption (e)</td>
<td>Energy consumption in Joule(J) by the mobile browser running on the mobile device for completely loading a web app.</td>
<td>RQ1</td>
</tr>
<tr>
<td>Loading time (lt)</td>
<td>Time until the web app is loaded in its entirety in seconds.</td>
<td>RQ2</td>
</tr>
<tr>
<td>CPU usage (cu)</td>
<td>The percentage of time that the CPU is being used to completely load the web app.</td>
<td>RQ2</td>
</tr>
</tbody>
</table>

Table 1 lists the three dependent variables we defined based on the goal of this study. They provide an objective picture of the impact of vendor prefixes on the energy consumption and performance of mobile web pages.

4.3 Experimental Hypotheses

The project’s purpose is to analyze the impact of the implementation of vendor prefixes. The vendor prefixes technique is the independent variable, while the blocking factors are the browser type and the device model. We consider the influence of the different prefixes based on each of the dependent variables in Table 1. In this experiment, we compute the means of the web pages \(\mu_{ijk}\), with \(i \in \{0, 1\}\) being a boolean indicating whether the vendor prefixes technique was applied to this sample (0 for the absence of CSS prefixes and 1 for the presence of CSS prefixes), \(j \in \{c, f, o\}\) represents the means for each browser Chrome, Mozilla Firefox and Opera respectively, and \(k \in \{e, lt, cu\}\) represents each of the dependent variables where: \(e\) represents energy consumption, \(lt\) represents loading time, and \(cu\) represents CPU usage.

Since there are no relevant studies on the energy consumption of vendor prefixes on mobile devices, we tested the null hypothesis that the energy consumption average of the vendor prefixes technique is the same, and the alternative hypothesis is shown below:

\[
H_{0,e} : \mu_{0jk} = \mu_{1jk}, \forall j \in \{c, f, o\}, k = e \quad (1)
\]

\[
H_{a,e} : \mu_{0jk} \neq \mu_{1jk}, \forall j \in \{c, f, o\}, k = e \quad (2)
\]

As the vendor prefixes technique has grown in popularity in recent years[11], we assume that vendor prefixes would improve the value of the ‘lt’ and ‘cu’ variables, and we execute one-sided
statistical tests on the hypotheses. The null hypothesis and the alternative hypothesis are shown below:

\[ H_{0,jk} : \mu_{0,jk} = \mu_{1,jk}, \forall j \in \{c, f, o\}, k \in \{lt, cu\} \]  
\[ H_{a,jk} : \mu_{0,jk} < \mu_{1,jk}, \forall j \in \{c, f, o\}, k \in \{lt, cu\} \]  

4.4 Experiment Design

We load each subject in three different browsers (Chrome, Firefox, and Opera) on two Android devices and measure its load time and energy consumption with and without vendor prefixes. Therefore, for those 25 selected websites from the Tranco list and similar lists, we have 450 runs for each round of experiments. This study considers 25 subjects (websites) with two factors: the presence of CSS prefixes (two treatments: present and absent) and browsers (three treatments: Chrome, Firefox, and Opera), and need to gather three metrics in parallel, each repeated a total of 18 times (two treatments for CSS prefixes multiplied by three treatments for browsers multiplied by three repetitions per subject).

Energy consumption and load time cannot be considered as fixed values. They are influenced by many factors, such as network bandwidth, the number of requests to the server at that time, and CPU usage. To minimize the impact of these factors, we take the following measures: (i) after each experiment’s run, the cache of the browser is cleared; (ii) the experiment is conducted in a stable WiFi network; (iii) the devices are not moved until the end of the experiment; (iv) there are no other Android applications such as games, mobile browsers, or social media apps running in background during the experiment.

4.5 Data Analysis

**Data exploration:** Using descriptive statistics histograms and box plots to get an indication of the energy consumption, load time, and CPU usage. Checking the scale of the data, its symmetry, and whether there are extremes.

**Checks for normality:** According to the histograms, if the overall shape is bell-shaped, Q-Q plots can be produced first to reveal whether the data (approximately) follows a normal distribution, i.e., the plotted points should follow on a straight line. Further, the Shapiro-Wilk normality test is applied, and the output p-value > 0.05 implies that the distribution of the data is not significantly different from a normal distribution.

**Paired t-test:** For the results that meet the above normality test, we perform a paired t-test, where the performance and energy consumption of mobile web pages with and without vendor prefixes applied are paired as two sets of data. We need to first do a subtraction to calculate the difference, then calculate the mean and standard deviation of this difference, and finally, compare the mean of the difference with “0”. If the null hypothesis is rejected, that is p-value < 0.05, it means that the difference between the mean of the difference and “0” is statistically significant, which means that there is a difference between the two sets of data, thus confirming that vendor prefixes affect the dependent variables we have selected.

**Wilcoxon signed rank test:** If the results do not meet the normality test. We use the Wilcoxon signed rank test to compare those two sets of data. This test still uses the p-value to determine the difference in the data. If p-value < 0.05, we can say there is a difference between the two sets of data, and prove that vendor prefixes have an impact on energy consumption and performance of mobile web browsing.

5 EXPERIMENT EXECUTION

5.1 Setup

To conduct our experiment defined in Section 3, for every web app, two versions are needed, for example: the Omegle web page with vendor prefixes and without vendor prefixes. Furthermore, the selected web pages are cloned locally using ResourceSaver plugin for Google Chrome. Afterwards, every clone is manually checked for errors. The process of checking for errors is composed of two steps: first is to consult the ResourceSaver’s logs when downloading a website, second is to check the local server’s console for errors when that website is hosted locally.

Then, for each saved CSS file used for that website and for each inline CSS code written inside an HTML file, the vendor prefixes are added or removed using two tools. In case the vendor prefixes are added, Autoprefixer tool will insert them into the web pages’ CSS files. In case the vendor prefixes are removed, the CSS Devendorizer tool will delete all the vendor prefixes from the web pages’ CSS files. This combined forms the input for the experiment. After saving the selected web pages to a local storage, we proceed to further experiment setup.

As mentioned previously and described in the experiment’s setup visualization presented in Figure 1, the experiment is executed on two Android mobile devices: a Samsung Galaxy J7 Duo with 3GB of RAM, a Mali-G71 GPU and an Exynos 7885 (14 nm) chipset and a Google Pixel 5 with 8GB of RAM, an Adreno 620 GPU, and a Qualcomm SM7250 Snapdragon 765G 5G (7 nm) chipset. Both devices have their screen brightness set to 100% (maximum) and without any other applications running in their background, as mentioned in Section 4.4. For the sake of power resources and monitoring, both of them are connected on a Raspberry Pi Model 4B. To begin the experiments, a JavaScript script for measuring load time is added to the subjects’ directory, BatteryManager, mem-CPU, and Android Runner are installed on the device to measure load time, energy consumption, CPU usage, and monitoring/controlling the hardware. Also, thanks to the configurations in Android Runner, battery charging is disabled when mobile phones are running experiments for more accurate results. For other required software, Chrome (version 106.0.5249.79), Firefox (version 105.0.2), and Opera (version 71.3.3718.67322) will also be installed on both devices to provide platforms for experiments. Since all these browsers’ settings function differently, we eliminate the differences to the minimum by disabling light theme and privacy settings, disabling all cookies, and turning off tracking protection.

After software requirements have been met, we configure the hardware (the Raspberry Pi Model 4B) with an Android Runner config.json file; here, the time interval between each run is set to

---

5. This is used as an example of the current implementation. For example, the Chrome web page with vendor prefixes and without vendor prefixes.

6. This is used as an example of the current implementation. For example, the Mozilla Firefox web page with vendor prefixes and without vendor prefixes.

7. This is used as an example of the current implementation. For example, the Opera web page with vendor prefixes and without vendor prefixes.

8. This is used as an example of the current implementation. For example, the Autoprefixer tool.

9. This is used as an example of the current implementation. For example, the Google Chrome web page with vendor prefixes and without vendor prefixes.

10. This is used as an example of the current implementation. For example, the Google Pixel 5 web page with vendor prefixes and without vendor prefixes.

11. This is used as an example of the current implementation. For example, the Raspberry Pi Model 4B web page with vendor prefixes and without vendor prefixes.
1 minute to ensure that the hardware has time to be optimized and cache can be cleared in time. Furthermore, in config.json file, the experiment repetitions’ number is set to 3 for each website, per browser, per device to ensure the accuracy of the collected metrics (see Table 1). Android Runner has its own Python scripts responsible for launching on the devices the selected web pages.

5.2 Measurement
Android Runner [9] is used to orchestrate the experiment the experiment. Dependent variables that are described in Section 4.2 are measured using different tools and plugins. For measuring energy consumption, BatteryManager Android Runner’s plugin is used. For measuring the load time, a script that is developed in JavaScript is used. This script is used for capturing the initial loading time of a web page. Afterward, the JavaScript scripts send the result to the local server. Finally, for measuring CPU Usage, mem-CPU Android Runner’s plugin is used, as mentioned previously. At the beginning of each run, the local Flask server is started. To obtain an accurate loading time for each web page and make sure that the data is not altered by the previous run, the browser’s cache will be cleared after each trial.

6 RESULTS
6.1 Data exploration
Tables 2 and 3 show the collected measures gathered from our experiment with and without vendor prefixes for each Android device, browser, and dependent variable; en stands for energy consumption in joules, load stands for loading time measured in milliseconds, and cpu represents CPU usage as a percentage.

By looking at the median row for the Samsung Galaxy J7 Duo data (see Table 2), it can be seen that after the removal of vendor prefixes, the energy consumption of loading the selected web pages using Chrome and Opera slightly increased, whereas it slightly decreased when using Firefox. For loading time, all browsers have shorter loading times without vendor prefixes. For CPU usage, all browsers use more CPU when CSS vendor prefixes are removed.

When looking at the median row for the Google Pixel 5 data (see Table 3), after removing CSS vendor prefixes, the energy consumption in Chrome decreased, while it increased in Opera and it stayed the same in Firefox. All three browsers have shorter loading times without CSS prefixes and all three browsers tend to use slightly more CPU without CSS prefixes.
Table 2: Descriptive statistics for the Samsung Galaxy J7 Duo

<table>
<thead>
<tr>
<th>Device Model</th>
<th>Samsung Galaxy J7 Duo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without CSS Prefixes</td>
</tr>
<tr>
<td>Browser</td>
<td>Chrome</td>
</tr>
<tr>
<td>Variable</td>
<td>en (J)</td>
</tr>
<tr>
<td>Mean</td>
<td>8.180</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>3.482</td>
</tr>
<tr>
<td>Max</td>
<td>15.700</td>
</tr>
<tr>
<td>75% Quantile</td>
<td>10.100</td>
</tr>
<tr>
<td>Median</td>
<td>7.120</td>
</tr>
<tr>
<td>25% Quantile</td>
<td>5.530</td>
</tr>
<tr>
<td>Min</td>
<td>4.250</td>
</tr>
</tbody>
</table>

Table 3: Descriptive statistics for the Google Pixel 5

<table>
<thead>
<tr>
<th>Device Model</th>
<th>Google Pixel 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Without CSS Prefixes</td>
</tr>
<tr>
<td>Browser</td>
<td>Chrome</td>
</tr>
<tr>
<td>Variable</td>
<td>en (J)</td>
</tr>
<tr>
<td>Mean</td>
<td>4.705</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>4.564</td>
</tr>
<tr>
<td>Max</td>
<td>21.776</td>
</tr>
<tr>
<td>75% Quantile</td>
<td>4.751</td>
</tr>
<tr>
<td>Median</td>
<td>3.536</td>
</tr>
<tr>
<td>25% Quantile</td>
<td>2.407</td>
</tr>
<tr>
<td>Min</td>
<td>1.809</td>
</tr>
</tbody>
</table>

CSS Prefixes

<table>
<thead>
<tr>
<th>Device Model</th>
<th>Google Pixel 5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With CSS Prefixes</td>
</tr>
<tr>
<td>Browser</td>
<td>Chrome</td>
</tr>
<tr>
<td>Variable</td>
<td>en (J)</td>
</tr>
<tr>
<td>Mean</td>
<td>6.170</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>2.259</td>
</tr>
<tr>
<td>Max</td>
<td>12.115</td>
</tr>
<tr>
<td>75% Quantile</td>
<td>6.827</td>
</tr>
<tr>
<td>Median</td>
<td>5.497</td>
</tr>
<tr>
<td>25% Quantile</td>
<td>4.868</td>
</tr>
<tr>
<td>Min</td>
<td>2.171</td>
</tr>
</tbody>
</table>

The inspected information from the previously described figures means that the collected data for energy consumption and loading time is probably not normally distributed, but it may be normally distributed for CPU usage.

Moreover, CSS prefixes seem to have an impact on energy consumption (see Figures 2 and 5) and CPU usage (see Figures 4 and 7). To better investigate the possibly implied tradeoff between energy consumption and performance, we apply a non-parametric test such as the Spearman’s rank correlation coefficient between the measured energy consumption and CPU usage (with alpha=0.05). As Table 4 shows, only for Google Pixel 5 in the scenario of "with CSS prefixes" for Chrome browser the energy consumption and...
CPU usage are correlated (p-value < 0.05), for all the other data, the energy consumption and CPU usage are not correlated (p-values > 0.05). Moreover, a 'rho' value greater than '0' (rho > 0) indicates a positive correlation, i.e., energy consumption increases with CPU usage; a 'rho' value lower than '0' (rho < 0) indicates a negative correlation, meaning that the energy consumption increases without the CPU usage. For Samsung Galaxy J7 Duo, since all p-values are higher than the alpha value, the correlation between energy consumption and CPU usage does not reach statistical significance. For Google Pixel 5, the CPU usage is lower than the alpha value (p-value = 0.001 < 0.05), meaning that the correlation between energy consumption and CPU usage is significant and the energy consumption increases with the CPU usage since the rho value is greater than '0' (rho = 0.617 > 0).

### 6.2 Normality checks

As described in Section 4.5, normality checks are necessary to determine the statistical tests to be applied when answering our research questions. For normality testing, the first step is achieved by observing the violin plots presented in Section 6.1. Then, we apply the Shapiro-Wilk test. For Samsung Galaxy J7 Duo, only energy usage for Opera without CSS prefixes is normally distributed. But for all other metrics, p-values surpass the alpha threshold, showing a non-normal distribution. For Google Pixel 5, the data for the energy consumption for Chrome with CSS prefixes is the exception, showing adherence to a normal distribution. Nevertheless, similar to the Samsung Galaxy J7 Duo, for all other metrics, the p-values surpass the alpha threshold, signifying a deviation from a normal distribution.

The final step is to generate the Q-Q plots for all dependent variables for each device. The Q-Q plot analysis for energy consumption of Opera without CSS prefixes on the Samsung Galaxy J7 Duo provides informal evidence, which is consistent with the formal findings from the Shapiro-Wilk test, the same consistency also applies for the rest of the data. However, this alignment between informal and formal evidence does not hold for the Google Pixel 5 device. Despite the Q-Q plot for energy consumption of Chrome with CSS prefixes on the Google Pixel 5 indicating a departure from a normal distribution, the Shapiro-Wilk test results for the same data set suggest otherwise, making it challenging to definitively conclude whether the data follows a normal distribution in this case.

With the checks described above, we can reasonably conclude that for all dependent variables only the energy usage of Opera without CSS prefixes for Samsung Galaxy J7 Duo follows a normal distribution, while all the other data is not normally distributed. The results of the Shapiro-Wilk test and the generated Q-Q plots are included in the replication package of this study.

### 6.3 Hypothesis testing

As described in the data analysis steps in Section 4.5, the initial plan was to use the Paired T-Test test on the data that fits the normal distribution and the Wilcoxon signed rank test on the data that does not fit the normal distribution. However, after the results of the normality checks in Section 6.2 and the conclusions that were drawn, we decided to use the Wilcoxon signed rank test. Table 5 and 6 present the obtained p-values for the data collected from the Samsung Galaxy J7 Duo and Google Pixel 5, respectively.
Table 6: Wilcoxon signed rank test’s p-values for all hypotheses for Samsung Galaxy J7 Duo

<table>
<thead>
<tr>
<th>Browser</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption</td>
<td>0.79147</td>
<td>0.52491</td>
<td>0.69151</td>
</tr>
<tr>
<td>Loading time</td>
<td>2.664e-05</td>
<td>4.541e-05</td>
<td>6.556e-06</td>
</tr>
<tr>
<td>CPU usage</td>
<td>0.00557</td>
<td>0.00882</td>
<td>0.01246</td>
</tr>
</tbody>
</table>

Table 6: Wilcoxon signed rank test’s p-values for all hypotheses for Google Pixel 5

<table>
<thead>
<tr>
<th>Browser</th>
<th>Chrome</th>
<th>Firefox</th>
<th>Opera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption</td>
<td>0.00963</td>
<td>0.75099</td>
<td>0.65284</td>
</tr>
<tr>
<td>Loading time</td>
<td>4.172e-06</td>
<td>5.960e-06</td>
<td>2.562e-06</td>
</tr>
<tr>
<td>CPU usage</td>
<td>0.93417</td>
<td>0.03507</td>
<td>0.06670</td>
</tr>
</tbody>
</table>

Based on the analyzed data, there is insufficient evidence to reject the hypothesis regarding energy consumption. All the p-values for this metric are greater than our alpha value (0.05). The sole exception to this pattern is observed in the energy consumption of Chrome on the Google Pixel 5 (see Table 6), where the removal of CSS prefixes leads to statistically significant energy savings.

Regarding loading times, we reject the null hypothesis for both Android devices and all browsers. This indicates that there is a statistically significant difference between the loading times of web apps with and without CSS prefixes across all Android devices and browsers. However, as shown in Figures 4 and 7, the direction of such difference is inconsistent across Android devices and browsers. Specifically, on the Samsung Galaxy J7, Chrome and Firefox tend to have shorter loading times with CSS prefixes, but not on Opera, while on the Google Pixel 5 all browsers tend to have shorter loading times without CSS prefixes.

Regarding CPU usage, we reject the null hypothesis for the runs executed on the Samsung Galaxy J7 across all browsers. Differently, the data for the runs executed on the Google Pixel 5 exhibits a statistically significant difference only for Chrome. In general, as shown in Figures 4 and 7, CPU usage tends to be consistently higher in subjects without CSS prefixes.

7 DISCUSSION

To comprehend the implications of the experiment results, the interpretations are explained by answering the research questions in this section.

RQ1: What is the impact of the CSS vendor prefixes on the energy consumption of Android mobile Web browsing? The energy consumption of Android mobile Web browsing is analyzed by investigating the initial load time in milliseconds (ms), and CPU consumption in percentage (%). For the Samsung Galaxy J7 Duo (see Table 2) as well as for the Google Pixel 5 (see Table 3), several key findings can be summarized. For loading time, the loading time of Web apps without CSS prefixes exhibits a statistically significant difference across all browsers and devices and it is consistently in favour of Web apps without CSS prefixes; in other words, in our experiment Web apps without CSS prefixes loaded faster than those with CSS prefixes. A different pattern emerges when looking at CPU usage. The CPU usage of Web apps without CSS prefixes is statistically higher than those with CSS prefixes across all devices and browsers; in other words, in our experiment Web apps without CSS prefixes consume more CPU than those with CSS prefixes.

Several factors can explain the impact of CSS vendor prefixes on loading time and CPU usage:

1. Browser Parsing and Interpretation: as described in Section 1, when CSS vendor prefixes are used, Web browsers must parse and interpret multiple versions of the same CSS property, one for each vendor prefix. This can lead to longer loading times and more CPU cycles.

2. Redundant Code: CSS vendor prefixes involve duplicating code for the same styling properties. This duplication increases the size of a CSS file, leading to higher loading times, especially on slower network connections.
(3) **Rendering Differences**: browsers interpret and render CSS properties with vendor prefixes slightly differently. This can lead to inconsistencies in how Web pages appear across various devices and browsers, requiring additional CPU cycles to resolve rendering discrepancies.

(4) **Limited Browser Support**: over time, as browsers standardize and adopt CSS features, they often drop support for vendor prefixes\(^{12}\). Code with vendor prefixes may become obsolete, but developers may still need to support older browser versions that use these prefixes, resulting in unnecessary performance overhead\(^{13}\).

We advise Web developers to consider the following strategies to mitigate the impact of CSS vendor prefixes on the performance of a Web app:

1. To use feature detection and progressive enhancement to deliver specific styles only to browsers that require CSS prefixes.\(^{14}\)
2. To try to minimize the use of CSS prefixes by focusing on standardized CSS features and properties that are generally supported.\(^{15}\)
3. To prioritize testing on a variety of browsers and devices to identify and address rendering inconsistencies caused by vendor prefixes. BrowserStack\(^{16}\) and Bitbar\(^{17}\) are two platforms that allow developers to test their Web apps on a wide range of browsers.
4. To regularly update and clean up CSS code to remove old prefixes and optimize styles for performance.\(^{18}\)

The experiment results provide valuable information on how CSS prefixes can affect the performance of Web apps, and the research might motivate developers to think about how they can find the right balance between performance and user experience; since utilizing CSS prefixes, the technology enables Web apps to adapt different browsers; but on the other hand, it forces the Web apps to load much slower and some Web apps can even function improperly due to the missing CSS properties. Furthermore, the data collected for the energy consumption metric in this experiment serves as an initial starting point for future research. Subsequent investigations can explore further the analysis of how CSS prefixes affect energy consumption, considering various devices and browsers to gain a more comprehensive understanding of their impact.

8 **THREATS TO VALIDITY**

8.1 **Internal Validity**

The cache of the browser is automatically cleared at each run to avoid any influence of caching factors on the results. Also, since we choose websites that include some of the most popular social media sites, accepting settings such as marketing cookies when entering a site can affect the results of repeated tests on the same site. Therefore, in all three browsers we tested, we disable cookies to avoid the effect of website cookies on our results. Most notifications are also disabled in the phone’s settings, as pop-up notifications from other applications or running in the background can make the results inaccurate. Charging is also disabled during the runs to prevent inaccurate energy consumption.

We also fully charged the smartphones’ batteries before each run. Considering that the phones’ light sensor may automatically adjust the screen brightness according to the ambient light, that many brands of mobile phone screens may have different screen materials, and that the dark mode may also significantly affect the energy consumption results, we decided to set the screen brightness to 100% in the phone settings to avoid any influence on the results.

8.2 **External Validity**

The subjects of the experiment are selected from the Tranco list. Five different areas are chosen to be closer to users’ browsing habits and to increase the diversity of the subjects. During our tests, we also found that not all websites are optimized accordingly for different browsers. Only a few of the sites we evaluated are optimized for Firefox and Opera, compared to the majority of those that are for Chrome. This represents a blocking factor, thus, only three browsers out of 200 browsers\(^{19}\) were considered for this experiment.

Another blocking factor is represented by the type of Android mobile device used. In this experiment, two such devices are used. The Galaxy J7 Duo model was released in 2018\(^{20}\) while the Pixel 5 model was released in 2020\(^{21}\). This means two years of technological advancement and technical differences between them.

8.3 **Construct Validity**

In the preceding sections, we explain the external settings in an attempt to minimize the effect of external impression factors on the experimental results. If the external influences cannot be changed, we ensure that they are equal during the experimental testing. Using the same settings for each test ensures that the external influences for each subject are the same. We utilize to save the websites to be browsed locally to ensure that any external network impacts, such as network packet loss, network delays, and any other potential surprises that may arise, will not alter the findings of the experiment. We go over the setup in great depth in Section 5.1.

The use of the local web app was another element that might have an impact on the experiment’s outcomes, so we decided to adjust the balance in that regard. We examined the per-page load time while evaluating various websites and three different browsers. We added JavaScript to the localized web pages in advance of the tests to display the load time, allowing us to precisely gauge how long it takes the page to load. We will retest to validate the validity of the experiment in the following studies if the same site loads noticeably differently on various platforms. We conducted several experiments, and we believe we gathered enough information to evaluate the effectiveness of each browser. For the sake of this investigation, we only test browsers on the Android platform, but

---

\(^{12}\) https://ishadeed.com/article/cross-browser-development/

\(^{13}\) https://css-tricks.com/is-vendor-prefixing-dead/

\(^{14}\) https://medium.com/@AlexanderObregon/cross-browser-compatibility-tips-for-making-your-html-css-code-work-on-different-browsers-4a97a611c3b4

\(^{15}\) https://robertnyman.com/2012/02/16/thoughts-on-the-css-prefix-situation/

\(^{16}\) https://www.browserstack.com/

\(^{17}\) https://smartbear.com/product/bitbar/


\(^{19}\) https://en.wikipedia.org/wiki/List_of_web_browsers

\(^{20}\) https://www.gsmarena.com/samsung_galaxy_j7_duo-9153.php

\(^{21}\) https://www.gsmarena.com/google_pixel_5-10386.php
future research may compare other collection platforms as well, such as iOS.

8.4 Conclusion Validity
We demonstrated the correctness of our results in the preceding sections by repeatedly carrying out the same test experiments. We have increased dependability by better organizing measuring instruments and minimizing contextual interference in the measurement environment, as can be observed in previous experiment planning. As can be observed, the data sample produces findings that are not exactly positively distributed, hence expanding the data sample is a potential future step. More precise indicator factors may be taken into account in this study as well as in follow-up research to provide more precise assessments.

9 CONCLUSIONS AND FUTURE WORK
This paper investigates the effects of the CSS prefixes technique on Android mobile web applications, both in terms of performance and energy consumption. We compared 25 websites with and without the CSS prefixes technology on three browsers with different type prefixes, including Google Chrome, Firefox, and Opera, on two Android mobile devices: Samsung Galaxy J7 Duo and Google Pixel 5. In this experiment, we considered three dependent variables: (i) energy consumption, (ii) loading time, and (iii) CPU usage. From the experimental data, we can conclude that the usage of CSS prefixes has a significant (but conflicting) impact on the performance of the measured Web apps across all three browsers and both Android devices; specifically, if on the one hand, the presence of CSS prefixes tends to lead to higher loading times, on the other hand, it leads to lower CPU usage. On the other hand, the presence of CSS prefixes has little influence on the energy consumption of the measured Web apps. Therefore, while the CSS prefixes can be optimized for adapting websites to different browsers, developers might consider abandoning such a technique if shorter loading times must be prioritized in their projects.

Future work involves expanding the sample first. From the p-values for all variables using the Shapiro-Wilk test, it is visible that the CPU usage data is only normally distributed for Opera for Samsung Galaxy J7 Duo, probably due to the fewer repetitions, small sample size of the websites and variable assets size of the web pages, including links, images, etc. The experiment could be enhanced by increasing the sample size or by uniformly checking the complexity of the websites. In addition, adding dependent variables to the experiment is another extension aspect, for example, loading time can be specified as First Contentful Paint (FCP), Time To First Byte (TTFB), Largest Contentful Paint (LCP), and Speed Index (SI)[12]. All these methods are expected to improve the reliability of future replications of this experiment.

Acknowledgements. This project is partially supported by the European COST programme under action identifier CA19135 (CERCI-RAS).

REFERENCES
[12] J. Obikoya, "Load time vs render time. find out how these metrics affect your website."