An empirical study on the performance and energy costs of ads and analytics in mobile web apps

Christos Petalotis, Luka Krumpak, Maximilian Stefan Floroiu, Larêb Fatima Ahmad, Shashank Athreya, Ivano Malavolta

Vrije Universiteit Amsterdam, The Netherlands

ABSTRACT

Context: As the use of mobile devices has increased immensely through the years, the presence of analytics and advertisements on web and native applications has become prevalent. However, serving ads and analytics comes with costs, as they are associated with additional code and network requests to execute properly. Subsequently, more computing resources are used, having an impact on the energy consumption and the performance of web applications. Previous work has focused only on native Android applications, has used different metrics for performance, or has focused on other aspects of web applications.

Goal: This paper aims to investigate the costs of including advertisements and analytics in web applications. This is done in terms of energy consumption and performance. For energy, the consumption is measured in Joules. For performance, the following metrics are used: first contentful paint and full page load time. The results of this study could influence the decisions of web developers and web browser vendors related to ads and analytics usage, while providing the foundation for further research on this topic.

Method: To collect reliable and population-representative results, the research focused on 9 popular web applications included in the Tranco list. Energy consumption and performance metrics were gathered for 3 versions of each web application — original version with ads and analytics, without ads, and without analytics. A cross-over paired comparison design is conducted. Multiple executions of each run were performed in random order to ascertain rigorous measures. The experiment is carried out on an Android tablet using two browsers, Google Chrome and Opera.

Results: Ads significantly impact the energy consumption of mobile web apps for both browsers, with a large effect size; analytics have a significant impact on the energy consumption of Chrome (with a medium effect size), but not on Opera. In terms of performance, both ads and analytics do not significantly impact the first contentful paint metric on both browsers; differently, both ads and analytics significantly impact the full page load time of the mobile web apps on both browsers, but with a small effect size.

Conclusions: This study provides evidence that both ads and analytics can have a significant impact on the energy consumption and performance of mobile web apps loaded either on Opera or Chrome. Depending on the requirements of the mobile web app, it is advisable to limit both ads and analytics in a mobile web app in order to reduce its energy consumption and improve its full page load time. Special attention should be paid to the presence of ads since they resulted to be the most impactful in terms of energy consumption.

1. Introduction

Today, there are currently 6.64 billion unique smartphone users in the world who can take advantage of a small, smart handheld device.¹ This means at least 83.4% of the world’s population owns a smartphone. Over the past decade, the value and importance of a smartphone have increased due to consumers’ access to a variety of services: banking, payment, e-mailing, web browsing, and so forth. This has led to dramatic changes in the field of advertising and analytics due to different media consumption patterns, along with technological advancements and the growth of digital media [1]. As a result, today, the presence of ads and analytics spans various parts of the digital world, from simple websites to advanced mobile applications.

¹ Corresponding author.
E-mail addresses: c.petalotis@student.vu.nl (C. Petalotis), l.krumpak@student.vu.nl (L. Krumpak), m.floroiu@student.vu.nl (M.S. Floroiu), l.f.ahmad@student.vu.nl (L.F. Ahmad), s.m.athreya@student.vu.nl (S. Athreya), i.malavolta@vu.nl (I. Malavolta).
Digital advertisement is a form of advertising on websites or applications through a range of different media formats, including text, image, video, and audio. Such formats are often seen on social media platforms which regularly display sponsored content. However, ads are also prevalent within e-commerce, streaming, or even educational websites. More websites are able to implement some form of digital advertising, and this becomes especially easy with tools such as Google AdSense\(^2\) or Amazon Ads\(^3\) which allows a website owner to quickly implement any form of digital advertisement.

However, advertisement comes with a drawback. The ads are required to be loaded on top of the web application source code. To measure the performance of a digital ad, one might consider the clicks, impressions and click-through rate. Such performance parameters require several analytics tools to help verify the integrity of the data. Thus, such a process might affect the performance and energy consumption of a mobile device.

In addition to the analytics tools used for digital advertisements, web analytics can also help analyze website traffic, user behaviour, bounce rates, goal conversions, and real-time visitor count. Tracking these attributes of a web application requires operations to run during the entire visit of a user to the web app, leading to increased energy consumption on the user’s device. This behaviour can also affect the performance of a web application since such logic needs more resources to run. Some of the most popular analytic tools include Google Analytics, Adobe Analytics, and Mix Panel. Regarding Google Analytics, 55.49\% of all websites use its service, while 84\% of all websites that use traffic analytics tools are using Google Analytics.\(^4\)

The goal of this research is to analyse the performance and energy costs of ads and analytics on mobile web apps. The results of this research are intended to provide developers, browser vendors, and researchers with insights into the energy consumption and performance drain caused by running such software on consumer mobile devices.

In order to achieve the above-mentioned goal, we select 9 web applications and execute them on an Android mobile device. Three versions of each of these applications are analysed — the original version of the mobile web app, containing any existing code that enables ads and analytics, a second version where ads are removed, and a third version where analytics are removed. To collect these measures and quantify the desired quality attributes of web apps, we use various metrics. More specifically, the energy consumption of the mobile web app is measured in Joules, using a specialized tool called Android Runner \(^2\) and its BatteryStats plugin. In order to measure the performance difference of mobile web apps running with and without ads and analytics-related code, we prompt to gauge the initial load time of a web app in terms of First Contentful Paint (FCP) and Full Page Load Time. These metrics are selected to represent the performance of a web application in our study, as they are user-centric metrics that greatly impact the experience of users \(^3\). For these measurements, the performance profiler, PerfumeJS, of Android Runner is utilized. It is important to note that in this study we focus on devices running the Android operating system (OS) and the obtained results are valid exclusively for mobile Web apps loaded on browsers running on Android devices.

The main results of our study can be summarized as follows: (i) ads significantly impact the energy consumption of mobile web apps for both browsers, with a large effect size; (ii) analytics have a significant impact on the energy consumption of Chrome (with a medium effect size), but not on Opera; (iii) both ads and analytics do not significantly impact the first contentful paint metric on both browsers; and finally, both ads and analytics significantly impact the full page load time of the mobile web apps on both browsers, but with a small effect size.

The target audience of this paper includes web developers who design, engineer, and develop mobile web apps, but also browser vendors and researchers. Our results are important for web developers since they provide evidence about the existing energy and performance issues that ads and analytics might cause to their final product, as they may hinder the user experience and lead to increased battery usage. This can be a serious concern for end users who rely on their mobile devices for various everyday tasks, especially considering the reduced battery sizes of mobile devices. The findings of this study could also prove useful to browser vendors since they provide evidence about the impact that ads and analytics might have on performance and energy consumption on two different browsers. Moreover, we aim at providing hard evidence that researchers can use to conduct further studies on this topic and expand the knowledge of the community on the potential performance and energy issues caused by ads and analytics on mobile web apps.

For the sake of independent verification and replication of the study, we provide a complete replication package\(^5\) containing the raw data, the source code of our pipeline for executing the experiment, and the scripts for analysing the data.

In summary, the main contributions of this paper are:

- an empirical evaluation of the impact of ads and analytics on the energy consumption and performance of mobile web apps loaded on two different Android browsers (i.e., Google Chrome and Opera);
- a discussion of the obtained results and their implications for web developers, browser vendors, ads/analytics platform vendors, search engine vendors, and researchers;
- a complete replication package of the study containing its results, raw data, and data analysis scripts.

2. Related work

In this section, we summarize relevant scientific papers that also look at either ads or analytics as part of mobile (web) apps, highlight the differences between these other studies and ours, and indicate how our experiment complements the discussed related work.

Gui et al. carried out an experiment on the hidden costs of advertisements on native apps from the Google Play store \(^6\). The authors report on a mobile device’s network usage, energy consumption, CPU performance and memory usage, when native apps are run with and without advertisements. Additionally, they analysed how much code is related to ads in the history of the apps’ updates, and how ads influence user experience based on app ratings. In order to remove ads from applications, Gui et al. translated each app’s APK-file into Java bytecode using the dex2jar API framework, such that they could identify the ad-related code and modify it. The results of the experiment, performed on 21 different apps, claim that the median increase was 56\% in CPU usage, 22\% in memory usage, 15\% in energy usage and 97\% in network usage, when ads were enabled compared to when ads were disabled. The experiment described in this report differs from Gui et al.’s experiment firstly, as this paper concerns mobile web apps, not native apps. Secondly, our experiment also looks at other metrics, such as the first contentful paint\(^6\) and the full page load time\(^6\) of web applications, in addition to energy consumption.

Varvello and Livshits performed an empirical evaluation of the energy consumption of Android browsers under different configurations, including the presence of ad-blocking features \(^7\). The study targeted

\(^2\) https://www.google.com/adsense/start/.
\(^3\) https://advertising.amazon.com/.
\(^4\) https://www.renolon.com/google-analytics-statistics/.
\(^6\) https://web.dev/fcp/.
\(^7\) https://www.stackpath.com/edge-academy/what-is-page-load-time/.
15 Android browsers with/without adblocking, dark mode, power savings mode, two Android devices, different network conditions, and varying levels of workloads (from 10 up to 600 websites and up to 100 concurrent browser tabs). Among the various results of their study, Varvello and Livshits provided evidence that browsers with ad-blocking features (e.g., Opera, Brave, Firefox Focus) are more energy efficient than browsers without those features (e.g., Chrome, Baidu, Firefox), offering battery savings of up to 30% in extreme cases. Our results about the overhead of ads is in line with the findings of the study by Varvello and Livshits.

Heitmann et al. performed another empirical study targeting ad-blocking features of the Brave browser, while loading 23 web apps mined from the Alex list [8]. Similar to our study and that of Varvello and Livshits [7], the study by Heitmann et al. confirms that ads can have a significant impact on the battery life of an Android device, with different savings depending on the characteristics of the loaded web app and, in this case, also depending on the configuration of the Brave browser. Despite the methodological similarities, our study complements the ones by Varvello and Livshits and Heitmann et al. since: (i) our study focuses on the presence of ads and analytics in web apps, whereas the other two studies focus on the blocking of ads, which are still downloaded, parsed, and (generally) executed by the browser, (ii) our study includes also analytics in addition to ads, and (iii) our study targets primarily web developers and how they can improve their own web apps, whereas the other studies target primarily either end users (Varvello and Livshits) or browser vendors (Heitmann et al.).

D’Ambrosio et al. performed a study on the privacy and energy consumption of mobile web apps running on the Firefox browser [9]: the study is composed of two main parts: (i) a user study aiming at eliciting the users’ perceptions and behaviour with respect to the privacy of mobile web apps (and browsers in general) and the battery consumption while browsing the Internet; and (ii) an empirical assessment of the impact of enabling privacy-enhancing mechanisms on the energy consumption of the browser. In this context, the considered privacy-enhancing mechanisms included: ads, analytics, “unwanted content”, Web bugs, and social widgets. Interestingly, the empirical assessment provided initial evidence that enabling privacy-enhancing mechanisms in the Firefox browser in Android led to significant energy savings (i.e., between 3% and 35%). This result is in line with the results of our study, where we obtained, as an example, an average energy saving of 41.36% and 37.97% in Chrome when removing ads and analytics, respectively. We obtained higher savings since our study focuses on the presence of ads and analytics in mobile web apps, whereas [9] focuses on the blocking of ads only. Another difference between the two studies is that the study by D’Ambrosio et al. focuses on privacy concerns from the users’ perspective, whereas our study focuses purely on energy concerns from the perspective of developers (who have control on the source code of the mobile web apps), browser vendors, researchers, and end users. The study by D’Ambrosio et al. has been carried out in 2013, whereas our study is carried out in 2022, and as such the two experiments used very different versions of Android and browsers with respect to our study.

Huber et al. evaluated the energy-efficiency of web-based UI components in cross-platform mobile apps [10] via a controlled experiment [11]. The intuition behind this experiment is to provide a mechanism in which cross-platform mobile apps can be developed in such a way that energy-intensive UI components (e.g., animations) are handled as native Android objects, and then the contents of the web app are managed via Web technologies. The experiment in [11] indicated a substantial increase in energy-efficiency for the Dialog (38.3%) and Sheet (54.7%) UI components, but no improvement for the Drawer and Scrolling components. Our study differs from the one by Huber et al. since we focus on ads and analytics of mobile Web apps running in Android browser, instead of focussing on the UI of cross-platform mobile apps.

Recently, Dornauer and Felderer conducted a secondary study on energy-saving strategies and techniques for mobile web apps in the 2012–2022 period [12]. The study has been designed and executed by following the systematic literature review methodology and it provides a thorough overview of (i) the hardware components that are commonly associated with energy consumption of mobile web apps, (ii) the various approaches available in the state of the art for measuring the energy consumption of mobile web apps, (iii) the most recurrent experimental settings considered by researchers when measuring mobile web apps, and (iv) existing approaches for reducing the energy consumption of mobile web apps. When dealing with the latter point (i.e., existing approaches for energy reduction), the authors of [12] extracted a category of approaches called "Reduction of web-traffic", which includes also the usage of ad-blocking techniques (although there is no mention of blocking/removing analytics); according to the authors, blocking ads in the browser saves energy primarily because (i) the network requests originally coming from the ads are not made and (ii) the CPU is less stressed since the animations, visualizations, and rendering operations of the ads are not executed. Being designed as a controlled experiment, our study is methodologically different from [12]. Nonetheless, the measurement-based results emerging in our study confirm the literature-based results emerging from [12].

In 2021, Papadogiannakis et al. conducted an experiment exploring how websites operate in terms of tracking and sharing of user data and behaviour, and how this aligns with a user’s consents of a website’s cookies policies [13]. While the authors investigated the extent of websites that leak user data using analytics, the discrepancy of this practice in varying countries, and GDPR violations, our experiment has a different goal as we look at the energy consumption of analytics-related code in web apps. The authors have provided detailed descriptions for detecting how websites track users through methods such as browser fingerprinting, first-party ID leaking and third-party ID synchronization. This reveals the complexity of user-tracking in web apps, and could prove useful in measuring the performance costs of analytics-related code in web apps for the experiment reported in this paper.

In practice, measuring the costs of running ads on mobile devices can be a convoluted endeavor. While performance and network cost can be easily determined by tools or software such as tcpdump, accurate energy consumption metrics may require specialized equipment, which would be unreasonable to believe any developer may have at their disposal. In 2016, Gui et al. proposed and evaluated two lightweight statistical approaches to determine the ad energy cost supported by the end users [14]. The Static Model was designed for the early stages in an app’s development lifecycle, when the app under development is yet to be fully implemented. This model provides insight into ad energy costs by looking at the ad refresh rate, the ad size, and the ad type (image or text). The second method introduced proposes a comparison between a fully implemented app with ads and the same app without ads. The authors look at three core components that are essential for the execution of ad functionalities: display energy, network energy, system energy. While the authors determine energy consumption of ads on native mobile applications using the aforementioned formula, the current experiment employs a software tool called Android Runner [2]. More specifically, the BatteryStats plugin for this tool is used to determine the energy consumption of ads and analytics on mobile web applications, while the PerfumeJS plugin for Android runner is used to measure the performance of these applications.

Papadopoulos et al. conducted a qualitative and quantitative survey in 2018 about the cost of digital advertisements and compared the user
cost along with the cost of the advertiser. They highlight the cost of data usage (in bytes) by the user during advertisements; on average, a user pays three times more for using the advertisement bytes than the advertiser pays for the advertisement. Using more bytes during advertisements implies direct increase in usage of the network related hardware on the device. In total, 67.4% of users paid more in bytes than what the advertisers paid for the same ads to be delivered. This means that most mobile users pay more in data plan cost to download each impression (or even in total throughout the year) than the corresponding cost that advertisers pay to send the given ads displayed [15].

In 2010, Simmons et al. carried out experiments to measure the energy consumption and performance costs of advertisements and analytics on personal computers (PCs), with different hardware and software configurations [16]. Such configurations ranged from different types of CPUs, GPUs and operating systems, to different browsers or different means of filtering ads and analytics on web applications. The results of the experiments have shown that running advertisements has caused a 3.4% increase in energy consumption which was comparable to a 5% decrease in battery life on laptops with desktop-life CPUs, a conclusion drawn by AnandTech in their search for the most energy-efficient browser. It would be interesting to determine whether the technological advancements that have been accomplished along the years will lead to a positive impact on the cost of running ads and analytics on mobile web apps. Mobile devices have come a long way, in both hardware and software advancements, albeit battery life is still troubling many of us. It would be of great interest to determine whether mobile devices are prone to a significant decrease in battery life due to constant exposure to ads and analytics when browsing online.

In summary, our study provides novel knowledge and evidence regarding the energy consumption and performance impact that analytics have on mobile web apps, as opposed to focusing on the different ways in which analytics can be implemented. Furthermore, even though the impact of ads on energy consumption and performance in the context of mobile software has been assessed in previous scientific studies, the centre of attention of our study is on the removal of ads (and analytics) from mobile web apps, instead of either applying ad-blocking techniques in the browser or improvements in native Android applications.

3. Experiment definition

In this section, the QQM framework [17] is utilized with the goal of defining the purpose of the described experiment, the research questions we intend to answer with this paper, and the metrics used to accomplish this.

The objective of the experiment can be defined formally by following the Goal-Question-Metric [17] approach in the following manner:

“Analyze the costs of ads and analytics for the purpose of evaluation with respect to their impact on performance and energy consumption as seen from the point of view of web developers, browser vendors and researchers in the context of mobile web applications on Android”. A visual representation of this definition is shown in Fig. 1.

Mobile devices tend to have relatively small batteries and limited resources, such as CPU and memory, especially compared to larger devices, such as laptops and desktop computers. Therefore, it is important to understand how advertisement and analytics software injected into web applications affect battery usage and application performance when running on mobile devices. More specifically, we are interested in exploring this topic for devices running the Android operating system (OS), considering that this OS is installed on 71% of mobile devices worldwide. Since advertisements and web analytics constitute different software operations, possibly with different effects on performance and energy consumption, we derive two categories of research questions, one for Energy Consumption and one for Performance. The resulting four research questions of this study are reported below.

RQ1: How does the use of ads affect the energy consumption of mobile web applications on Android?
RQ2: What is the impact of ads on the performance of mobile web applications on Android?
RQ3: How does the use of analytics affect the energy consumption of mobile web applications on Android?
RQ4: What is the impact of analytics on the performance of mobile web applications on Android?

4. Experiment planning

The objective of our experiment is to investigate the potential overhead the addition of advertisements and analytics may have on the performance and energy consumption of web applications running on mobile devices. To achieve that, it is important to define the context and planning of the research in a way that allows for clear and objective results to be derived. For that, we follow the guidelines proposed by Wohlin et al. [18].

4.1. Subjects selection

The subject group of this experiment is comprised of production-level web applications that are accessible on the internet, while the scope is related to the Android ecosystem. Additionally, to run these applications, it is necessary to use a web browser capable of reading and properly interpreting the content of web applications. Due to the vast number of choices for these aspects of the experiment, certain decisions were made to ensure its feasibility, as described below.

In order to perform the experiment, the use of an Android mobile device is necessary, as our focus is on the use of web applications on such devices only. Many options are available today, and from those we prompt to use a Nexus 9 device for all the runs performed during our experiment. Only this device is used, as the employment of a second or third device would require extra effort that was not feasible to be completed within the time frame of this experiment.

Furthermore, we note that the Android version installed on the Nexus device is Android 7.1.1. This version is one of the 11 major Android OS releases that have been published by Google since 2013. However, since the runs of our experiment are performed on web browsers, we expect that the version of the Android OS would have minimal influence on the subjects and results of our experiment.

Furthermore, we expect different browsers to have divergent results, as the underlying mechanism of rendering and running a web page’s contents might differ between different options. In an attempt to make the research results as general as possible, while maintaining the experiment’s execution feasibility, we opted to perform our tests on the two most popular browsers on Android. In particular, we consider Google Chrome and Opera, with a market share of 59.3% and 1.32% respectively. While making this selection, we exclude the Samsung Internet browser, which, with a market share of 4.32%, is generally the 2nd most popular browser on Android. The reason for the exclusion of this type of browser is the fact that it is only available on Samsung devices and could result in measures that are not representative of the wider Android ecosystem. Thus, we opt to use the next most popular browser on Android, namely Opera.

Lastly, to draw meaningful and reliable conclusions about the effects of ads and analytics on the performance and energy consumption of web applications, it is important to consider web applications that are

---

popular and widely used. This makes the research results applicable for a great number of mobile users, thus increasing their significance. In an attempt to do that, we narrowed the list of considered web applications down to the ones included in the Tranco list [19] generated on the 24th of September 2022. This list contains the one million most popular web applications in the world in descending order.

Once this list is acquired, using a Python script, we select the 1000 most popular and unique domain names contained in it. This means that only the first entry is logged between URLs with the same domain name but different extensions. As an example, if both www.amazon.nl and www.amazon.it are contained in the Tranco list, and the former is situated closer to the top compared to the latter, then only the former instance of the Amazon website is included in the final list. Finally, conducting non-probabilistic quota sampling [20] on the exported list, we manually select 9 web pages that contain both ads and analytics across 3 different categories, namely education, news, and science, with 3 entries for each category. Table 1 lists the selected subjects that are used throughout our experiment execution, while Table 2 presents the inclusion and exclusion criteria of our sampling process.

Table 1
<table>
<thead>
<tr>
<th>Domain name</th>
<th>Category</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>theguardian.com</td>
<td>News</td>
<td>133</td>
</tr>
<tr>
<td>dailymail.co.uk</td>
<td>News</td>
<td>227</td>
</tr>
<tr>
<td>w3schools.com</td>
<td>Education</td>
<td>345</td>
</tr>
<tr>
<td>independent.co.uk</td>
<td>News</td>
<td>393</td>
</tr>
<tr>
<td>britannica.com</td>
<td>Education</td>
<td>497</td>
</tr>
<tr>
<td>biomedcentral.com</td>
<td>Science</td>
<td>776</td>
</tr>
<tr>
<td>geeksforgeeks.org</td>
<td>Education</td>
<td>787</td>
</tr>
<tr>
<td>scientificamerican.com</td>
<td>Science</td>
<td>951</td>
</tr>
<tr>
<td>science.org</td>
<td>Science</td>
<td>970</td>
</tr>
</tbody>
</table>

Table 2
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inclusion</td>
<td>Top 1000 unique URLs of Tranco list</td>
</tr>
<tr>
<td>Exclusion#1</td>
<td>Presence of ads &amp; analytics</td>
</tr>
<tr>
<td>Exclusion#2</td>
<td>Downloadable using appropriate tools</td>
</tr>
<tr>
<td>Exclusion#3</td>
<td>Ability to properly run locally</td>
</tr>
</tbody>
</table>

The presence of ads and analytics are identified as the independent variables of our experiment with the purpose of addressing the research questions outlined in Section 3. More specifically, the factor considered to answer RQ1 and RQ2 is the presence of ads. The presence of analytics is examined while answering RQ3 and RQ4. For each run carried out in the experiment, one of the three following treatments occurs:

- Ads and analytics-related code is present (original code)
- Ads-related code is absent
- Analytics-related code is absent

As the objective of the conducted research is to find the effect of the aforementioned independent variables on the energy consumption and performance of web applications, these qualities constitute the dependent variables of our experiment. Particularly, the amount of energy consumed during a run is measured in Joules (J) and to measure its value, the BatteryStats plugin in AndroidRunner is employed. Regarding the performance of web applications, it is measured in milliseconds (ms) in terms of first contentful paint and full page load time. Table 3 lists these dependent variables and related information.

Additionally, it is considered that different browsers may introduce performance variations to the results, as the underlying browser mechanism is not always the same. Therefore, we have identified the type of browser as a blocking factor in our experiment. For this reason, two different browsers, Google Chrome and Opera, are used separately to conduct the experiment.

4.3. Experimental hypotheses

When web applications come with ads and analytics functionality, the browser needs to retrieve, process and load the code related to these tasks, on top of the code that is required to run the main functionality of the web applications. Furthermore, such tasks require additional network requests to be performed by the browser, to allow for the proper operation of analytics or display of ads on its window.

As a result, the assumption that the presence of ads and analytics on web applications leads to greater energy consumption, but also worse performance, compared to versions of the same applications without them, is considered reasonable. Moreover, as discussed in Section 4.1, the type of browser could affect the measures of our experiment, thus, we consider this variable as a blocking factor. For this reason, the experiment is performed separately for each type of browser we have selected, namely Google Chrome and Opera. Therefore, to answer the
Table 3
Dependent variables of experiment.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>RQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Consumption (EC)</td>
<td>Energy consumed in Joules (J) by completely loading a web-page on the mobile browser</td>
<td>RQ1, RQ3</td>
</tr>
<tr>
<td>First Contentful Paint (FCP)</td>
<td>Time in milliseconds (ms) until the first text or image is displayed</td>
<td>RQ2, RQ4</td>
</tr>
<tr>
<td>Full Page Load Time (FPLT)</td>
<td>Time in milliseconds (ms) until the website is fully interactive</td>
<td>RQ2, RQ4</td>
</tr>
</tbody>
</table>

4.4. Experiment design

In order to measure EC, FCP, and FPLT for each website, we adhere to the three treatments listed in Section 4.2. Considering these treatments, a paired comparison design is conducted – a cross-over design [21] – in which each treatment is applied on each subject. Furthermore, to ensure that no bias exists in the order of execution, each web application is served with a random treatment, while the subject selected at each run is up to chance as well. The randomization is also extended to the profiler that is used each time to take measurements on the mobile device. This way the effect of uncontrolled external factors is mitigated for a certain subject. Therefore, each website is served three times in each of the selected browsers.

Furthermore, we use two profilers, BatteryStats and PerfumeJS, to conduct measurements on the selected subjects. Each profiler runs once on every subject for each experiment trial. Consequently, the result is 108 runs for each trial. It is possible that the measured values of the dependent variables (EC, FCP, FPLT) fluctuate during execution; for this reason, we intend to run each trial 20 times and store each result. To replicate server-like behaviour for locally hosted web-pages, we intend to use a proxy server with bandwidth throttling, since locally hosted web-pages load quickly and measured values for FCP and FPLT may not be realistic in this case.

4.5. Data analysis

To answer the research questions proposed in Section 3, the collected experiment measures are analysed in three steps: data exploration, check for normality and data normalization, hypothesis testing, and effect size estimation.

Using a combination of descriptive statistics, histograms, and violin plots, we receive a preliminary understanding of metrics related to energy consumption and performance during the data exploration phase. Next, for each treatment and browser used, we explore whether the measures of the dependent variables are normally distributed. This is achievable by two means:

- Visually inspecting a density plot or Q–Q plot;
- By conducting a Wilk–Shapiro normality test which examines whether a variable is normally distributed in some population.

With respect to the hypothesis testing, provided the data assumes normality, a simple t-test could be used to test whether the research questions have any statistical significance behind them. Otherwise, the Wilcoxon signed-rank non-parametric test will be used. The significance level will be $\alpha = 0.05$.

Finally, the Cliff’s Delta measure will be applied to statistically assess the size of variations in energy consumption and performance of treatments. In addition to the hypothesis testing, Cliff’s Delta is a non-parametric effect size measure that will be employed to ascertain the presence of any practical relevance in the differences between the treatment combinations.

5. Experiment execution

This section is divided into several subsections with the purpose of demonstrating the organization of the experiment from the hardware perspective to the execution of the experiment. The first section discusses how the subjects of the experiment are prepared. After that, the process that is followed to collect the data of our independent variables is presented. Finally, the process of measuring the dependent variables is addressed.
5.1. Preparation

The first step in the experiment regards the selection of subjects to be examined during its execution. As described in Section 4.1, we have 9 web applications. To serve the selected applications locally, we use the python-wget\textsuperscript{16} tool to download the files necessary to run them. Then each application is manually checked for any errors in its code that would prevent it from running on a local configuration. Some commonly encountered issues are, incorrect references to the website’s respective JS or CSS and Cross-Origin Requests being blocked. These issues were solved by updating the path of the resources and redirecting each website URL to the device’s local IP. This step is essential as, in most cases, the ads & analytics would be prevented from working if the domain name is incorrect.

Afterward, for each saved website, the ads & analytics are removed manually from the code, resulting in three versions of the same website. One version contains both ads & analytics, which is the original version of the application, while another version is stripped of code related to ads and the third is stripped of analytics-related code. This process is done in pairs (two team members) to mitigate the potential bias of having a single person do it. Subsequently, we parse the code and test whether the examined website is able to run without any errors. Additionally, to ensure that all of the ads & analytics have been removed from the website, we use WireShark\textsuperscript{17} to verify that no ad/analytics related network requests are made. The end result consists of three files for each webpage (<websitename>_adfree.html, <websitename>_analyticsfree.html).

Additionally, once all 9 subjects have been prepared, we need to embed additional JavaScript code into the web pages that will send an HTTP POST request with the FCP and FPLT values. Once the Android Runner receives the request, the experiment will prematurely stop since the PerfumeJS profiler has already gathered all of the necessary data. However, this behaviour is not favourable when gathering energy consumption data with BatteryStats (see Section 5.2).

5.2. Setup

A visual representation of the setup of the experiment, the hardware used, and the connections between them is available in Fig. 2. The frameworks and tools we use were chosen because they are lightweight, have already been widely used, and are well documented.

Before executing the experiment, it is necessary to configure the setup of the hardware to ensure that all required tests are possible and can be performed correctly. Two devices are used — a mobile device running the Android operating system, which is used to perform the experiment on, and a Raspberry Pi device that performs the experiment, hosts the different versions of each web application, and collects the required metrics from the mobile device. The mobile device is the Nexus 9 model and the Raspberry Pi model is B+. The system specifications of these devices are shown in Tables 4 and 5.

The two devices are connected via a USB-to-USB-C cable, allowing access to the phone via the Raspberry Pi. Once this connection has been established, the AndroidRunner tool, installed on the Pi, is used to perform the experiment on the selected subjects. AndroidRunner is a Python framework designed to run experiments to collect measures in Android apps, both native and web. It is integrated with Android Debug Bridge and MonkeyRunner, which are tools that are utilized during the experiment too. Android Debug Bridge is a tool that enables the Raspberry Pi to communicate with and run commands on the mobile device through a command-line interface. This communication mainly involves commands for opening the web pages within a specific browser. Additionally, MonkeyRunner is a tool that allows us to interact with visible elements on a device and is used mainly to mimic human interactions.

In order to retrieve measures from the mobile device, we employ the PerfumeJS and BatteryStats plugins. Both of these tools are plugins that are installed within AndroidRunner. PerfumeJS allows for the collection of performance measures such as FCP and Full Page Load, while BatteryStats is used to provide an estimate of battery consumption on a mobile device.

Along with AndroidRunner, it is also necessary that we host the web applications, downloaded with the process explained in Section 5.1 on a web server, so that it is possible to access them through the mobile device and collect the necessary measures. To accomplish this, we use Apache HTTP Server,\textsuperscript{18} an open-source cross-platform web server software to host the websites on the Raspberry Pi. Therefore, the Android device is required to have both Google Chrome and Opera pre-installed before conducting experiment trials. The Android device used for this experiment had the Chrome version 85.0 and, Opera version 71.3 pre-installed. These browsers will be used to access the websites which are hosted on the Raspberry Pi.

To help minimize the effect that an unstable network would have on the experiment, we connect the Raspberry Pi to a local network using an Ethernet cable. Additionally, we use Wondershaper to restrict the bandwidth of the network adapter.\textsuperscript{19} We throttle the network speed of the Raspberry Pi to the likes of 4G LTE: 20 megabits per second, minimizing the possibility of fluctuating internet speeds.

5.3. Measurement

As described in the previous section, the trials are executed with AndroidRunner. The PerfumeJS and BatteryStats plugins are used to gather measures from the mobile device.

For each trial, we start by running a script that randomly chooses one of the 27 web applications that are hosted on the Apache web server. The script ensures each web page is only chosen once. To ensure that each run of the experiment is not affected by operations carried out earlier, we used two solutions. First, after each trial, the browser cache & storage is deleted to avoid implications that previously stored data can have on the measurements of the next run. Secondly, AndroidRunner is configured to not charge during the experiment to ensure consistent measurements. Additionally, the mobile device idles for 30 s after each run to ensure all background processes from the previous run have been killed.

\textsuperscript{16} https://www.scrapingbee.com/blog/python-wget/.

\textsuperscript{17} https://www.wireshark.org/.

\textsuperscript{18} https://httpd.apache.org/.

\textsuperscript{19} https://github.com/magnific0/wondershaper.
5.4. Analysis

All analysis is conducted using R\textsuperscript{20} in RStudio,\textsuperscript{21} more specifically versions 4.2.1 and 2022.08.2-579, respectively. The data science library tidyverse is used for data exploration, analysis, and plotting. To this end, we use the tidyverse\textsuperscript{22} packages dplyr\textsuperscript{23} and ggplot2.\textsuperscript{24}

6. Results

Figs. 3 and 5 depict the energy consumption and performance metrics (FCP, FPLT) of web applications with both ads and analytics present (\(\alpha\)), only analytics (\(\text{no_ads}\)) and only ads (\(\text{no_analytics}\)) as observed in the browsers Chrome and Opera. The specific values of all the measures collected are documented in Tables 9, 10, and 11.

In particular, Table 9 shows an overview of the measures collected for treatments with ads and analytics present. This Table documents the energy consumption, the first contentful paint, and the full page load time in Opera and Chrome. Tables 10 and 11 document the same measures for each browser, for treatments \(\text{no_ads}\) and \(\text{no_analytics}\).

In the following subsections, we investigate the results according to their impact on energy consumption and performance. For each category, we delve into data exploration where the focus is on presenting certain descriptive statistics for the measures collected, such as the minimum, maximum, and median values. The median is presented as it gives a more comprehensive picture of the distribution of the data collected, compared to mean values, considering outlier values. Afterward, we present how normality checks were performed, we conduct hypothesis testing for the set of hypotheses we specify in Section 4.1 and measure the size of effect where applicable.

6.1. Impact on energy consumption (RQ1, RQ3)

6.1.1. Data exploration

**ADS & ANALYTICS** — The runs on Chrome with both ads and analytics present had a maximum energy consumption of 1043.08 J and a minimum consumption of 322.73 J. The median consumption during the runs was 686.57 J. Regarding Opera, we observed that for most of the runs that were performed with versions of web applications with ads and analytics present, the median energy consumed was 562.63 J. Furthermore, for this browser the maximum energy consumption was 1091.06 J and the minimum consumption had a value of 127.95 J. The documented values suggest that the different browsers have different capabilities to host a web application with ads and analytics-related code present. The minimum energy consumption in Opera is somewhat of an outlier value, suggesting that this browser has a strong capacity to process ads and analytics.

**NO ADS** — In the case where only analytics were included in a web application’s code, the maximum consumption in Chrome was 789.78 J, the minimum value was 162.99 J and the median was 402.55 J. In Opera, when only analytics were present, the median energy consumption was measured to be 376.36 J. The maximum consumption was 1077.95 J and the minimum was 127.95 J. These results give the impression that analytics have a larger energy-consumption effect in Opera than in Chrome.

**NO ANALYTICS** — For the versions of web applications that included ads, but no analytics, the minimum energy consumption in Chrome was measured at 245.85 J, while the maximum consumption was 1281.09 J. The median consumption was 425.96 J. In the cases where only ads are present in Opera, the median energy consumption was measured to be 376.36 J. The maximum consumption was 1077.95 J and the minimum was 127.95 J. These results give the impression that analytics have a larger energy-consumption effect in Opera than in Chrome.

---

\textsuperscript{20} https://cran.r-project.org/.
\textsuperscript{21} https://www.rstudio.com/.
\textsuperscript{22} https://www.tidyverse.org/.
\textsuperscript{23} https://dplyr.tidyverse.org/.
\textsuperscript{24} https://ggplot2.tidyverse.org/.
6.1.2. Normality checks

In order to check for normality, the Shapiro–Wilk test was used in each of the different measures for every combination of browser and treatment. The W- and p-values collected from this test are presented in Table 6. The original distribution of the data is depicted in Fig. 3 for energy consumption in both browsers. It is observable from Fig. 3 that the data were not originally normally distributed. The same is confirmed by looking at the p-values in Table 6 for most of the data collected, with the exception of First Contentful Paint on Opera.

We applied a normalization algorithm provided by the bestNormalize R package\(^{25}\) to the obtained data in an attempt to achieve a normal distribution. However, the transformed data were not normally distributed as well. For this reason, we opted to use a statistical test that does not assume normality, namely the Wilcoxon Rank Sum test. This is explained in more detail in the next subsection, Hypothesis Testing.

6.1.3. Hypothesis testing

Without assuming normality in the data, a Wilcoxon Rank Sum test was executed to explore the significance of the difference that ads and analytics may have on the energy consumption of web apps. The results from the Wilcoxon test can be found in Table 7. Taking these results into account, we can draw certain conclusions about the impact of ads and analytics on the energy consumption of web applications. The treatments of no_ads and no_analytics resulted in statistically significant differences.

However, there is an exception regarding the treatment related to the removal of analytics-related code (no_analytics) from web applications when run on Opera; for this treatment, the null hypothesis could not be rejected as the p-value is 0.08844, higher than the set significance level of 0.05.

6.1.4. Effect size estimation

The strength of the differences between treatment combinations is evaluated by estimating the effect size using Cliff’s Delta and analysing a density plot. Looking at the density plots in Fig. 4, it is expected that the effect size for Ads and Analytics (o) is large against those Without Ads (no_ads) and small against those Without Analytics (no_analytics), considering the height and spread of both curves. The test shows an indication of the effect sizes on the energy consumption, the values of which are found in Table 8. The energy consumption for the treatment no_ads has a large size effect for both Opera and Chrome. For the treatment no_analytics, the size of the effect is small in Opera, and medium in Chrome.

6.2. Impact on performance (RQ2, RQ4)

6.2.1. Data exploration

ADS & ANALYTICS — In the context of the web application treatment involving both advertisements and analytics, the First Contentful Paint...
Table 9: Descriptive statistics for web applications without ads and analytics.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Browser</th>
<th>Variable</th>
<th>EC (J)</th>
<th>FCP (s)</th>
<th>FPLT (s)</th>
<th>EC (J)</th>
<th>FCP (s)</th>
<th>FPLT (s)</th>
</tr>
</thead>
</table>
| NO ADS — In the context of the treatment that involve no ads but only analytics, the following First Contentful Paint (FCP) metrics were measured for Opera: the maximum value was 3.65 s, the minimum value was 0.93 s, and the median value was 1.36 s. For Chrome, the corresponding FCP metrics were as follows: the maximum value was 4.70 s, the minimum value was 1.23 s, and the median value was 2.41 s. When considering the Full-Page Load Time (FPLT) metric, the maximum value measured for Opera was 15.10 s, the minimum value was 1.43 s, and the median value was 4.93 s. For Chrome, the corresponding FPLT metrics were as follows: the maximum value was 12.64 s, while the minimum value was 1.50 s. The median value was 5.28 s.

We can notice a significant difference in Full-Page Load Time when advertisements are not loaded. It can be assumed that the loading of ads along with an application’s main content leads to longer loading times.

Table 10: Descriptive statistics for web application without ads.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Browser</th>
<th>Variable</th>
<th>EC (J)</th>
<th>FCP (s)</th>
<th>FPLT (s)</th>
<th>EC (J)</th>
<th>FCP (s)</th>
<th>FPLT (s)</th>
</tr>
</thead>
</table>
| NO ANALYTICS — In the treatment where analytics were absent but ads were present (i.e. no_analytics), we observed that the maximum FCP on Chrome was 4.45 s, while the minimum FCP was 1.31 s. The median value for FCP on Chrome was 2.38 s. In contrast, the same treatment on Opera yielded a maximum FCP of 2.90 s, a minimum FCP of 0.92 s, and a median FCP of 1.35 s.

Next, we analysed the FPLT metrics for this treatment. The FPLT average measure for Opera was 6.29 s, whereas the median value on Google Chrome was 5.64 s.

One important observation is that the measures for this treatment indicate a very small difference in performance when compared with web applications running the full (ads & analytics included) version. Consequently, this leads us to believe that analytics do not affect the loading time or the FCP for webpages significantly.

In both browsers, the difference in Full-Page Load Time for the web application version with and without analytics is much smaller than the one noticed for the versions of the same subjects with and without ads. A potential explanation for this could stem from the fact that advertisements usually come with embedded images, videos, or audio that increase the size of the files that need to be downloaded and processed by the browser. Consequently, it takes longer for a website to fully load. This is not the case for analytics, which are usually implemented only within the code of the web application. In most iterations, we observe that not having advertisements results in faster page load times (FPLT) in both browsers; between the 3 treatments we explore, the performance is best when advertisements are excluded, followed by the web application versions without Analytics, while it is lowest when both Ads and Analytics are present. This is in accordance with our assumptions stated in Section 4.3 — this topic is discussed further in Section 7. Probably because Analytics do not involve images/videos and their collected data is sent to the servers asynchronously (so without impacting FPLT).

(FCP) and Full-Page Load Time (FPLT) were evaluated on Chrome and Opera. These values are later used as a reference for comparison for the subsequent two treatments, no_ads and no_analytics.

The Chrome runs resulted in a maximum FCP of 5.22 s and a minimum FCP of 1.33 s. The median FCP of the runs was 2.53 s. Conversely, Opera had a maximum FCP of 3.40 s, a minimum FCP of 0.87 s and a median FCP during the runs was 1.42 s.

Furthermore, when considering the same treatment, Chrome exhibited a maximum FPLT of 14.27 s and a minimum FPLT of 2.34. The median FPLT of the runs was 6.54 s. In contrast, Opera’s maximum and minimum FPLT were 19.50 and 2.03, respectively, and the median FPLT was recorded to be 7.59 s.
6.2.2. Normality checks

By analysing the distributions present in Fig. 5, we can ascertain that there are no visible normal distributions of the performance-related metrics. Additionally, the Wilk–Shapiro values present in Table 6 (e.g., for ads and analytics checks we had the following results: $1.761 \times 10^{-08}$ for FCP on opera, $0.000902$ for FPLT on Opera, $0.001183$ for FCP on Chrome, $1.267 \times 10^{-05}$ for FPLT on Chrome, etc.) indicate that we must reject the null hypothesis of normally distributed data. In order to perform any parametric hypothesis testing, we continue by transforming the data using the bestNormalize\(^{26}\) R package.

6.2.3. Hypothesis testing

Regarding the First Contentful Paint metric, removing ads or Analytics did not have any statistically significant effect, since the $p$-value was greater than 0.5 in both cases. Therefore, none of the treatments had a significant effect on the time it took for the web app to send a response to the client. When considering the results of our experiment related to the full page load time metric, the treatments no_ads and no_analytics were significant; the $p$-values of the Wilcoxon Rank Sum test were measured to be less than the significance level of 0.05. As a result, the null hypothesis can be rejected. This means that the time it took to fully load a web application’s page was significantly less without ads and without Analytics, than the time it took to load with ads and Analytics present. Table 12 depicts the statistical significance levels estimated by the Wilcoxon Rank Sum test.

6.2.4. Effect size estimation

The strength of the differences between treatment combinations will be evaluated by estimating the effect size using Cliff’s Delta and analysing a density plot. Looking at the density plots in Figs. 6 and 7, it is expected that the effect size for Ads and Analytics is against both Without Ads and Without Analytics, individually, considering the height and spread of both curves. The Cliff’s Delta test shows an indication of the effect sizes on the performance of the web apps, the values of which are found in Table 13.

The First Contentful Paint for the treatment no_ads had a small size of effect in Opera, and a negligible size of effect in Chrome. The size of effect for the measure Full-Page Load Time for the same treatment were small in both Opera and Chrome.

For the treatment no_analytics, the size of effect of First Contentful Paint was negligible in both Opera and Chrome. Regarding the Full-Page Load Time metric, the sizes of effect were small on both browsers.

7. Discussion

In order to guide the discussion of the obtained results, Table 14 presents a complete summary of the statistical analysis and effect size estimation procedure we performed in this experiment. The remainder of this section will elaborate on the main implications of the results about both ads (Section 7.1) and analytics (Section 7.2).
Table 12
Wilcoxon Rank Sum test results.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>With ads and analytics (full) vs. Without ads (no_ads)</th>
<th>Browser</th>
<th>Opera</th>
<th>Chrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td></td>
<td>FCP</td>
<td>FPLT</td>
<td>FCP</td>
</tr>
<tr>
<td>W-value</td>
<td></td>
<td>4668</td>
<td>4952</td>
<td>4342.5</td>
</tr>
<tr>
<td>P-value</td>
<td></td>
<td>0.07729</td>
<td>0.009905</td>
<td>0.4035</td>
</tr>
</tbody>
</table>

Table 13
Cliff’s Delta estimation results.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>With ads and analytics (full) vs. Without ads (no_ads)</th>
<th>Browser</th>
<th>Opera</th>
<th>Chrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td></td>
<td>FCP</td>
<td>FPLT</td>
<td>FCP</td>
</tr>
<tr>
<td>Cliff’s Delta</td>
<td></td>
<td>0.1525926</td>
<td>0.222716</td>
<td>0.0722222</td>
</tr>
<tr>
<td>Delta size</td>
<td></td>
<td>Small</td>
<td>Small</td>
<td>Negligible</td>
</tr>
</tbody>
</table>

7.1. Impact of ads on the energy consumption and performance of mobile web apps (RQ1 and RQ2)

As shown in Table 14, we can safely conclude that ads have a statistically significant impact on the energy consumption of mobile Web apps. Therefore, regardless of the browser used, the presence of ads significantly impacts the energy consumption when loading web apps. For this reason:

**Implication for Web developers**

We recommended Web developers consider the added value of having (multiple) ads in their Web applications. We also advise to prioritize the quality of ads over their quantity when deciding what types of ads to include in Web apps (e.g., by choosing ads platforms with a proven low overhead).

At the same time, we are aware that ads are a crucial source of revenue for most web applications, so it is very unlikely that ads will be totally removed from web apps to preserve the energy consumption of mobile devices. For this reason, we believe it is important to provide support to Web developers for precisely assessing the energy/computational demands of their ads. Such support must come primarily from browser vendors, like the Mozilla Foundation or Google. Specifically:

**Implication for browser vendors**

Browser vendors should provide support to Web developers for precisely assessing the energy/computational demands of ads (and analytics).

For example, Google Lighthouse\(^\text{27}\) is one of the most used platforms for debugging (mobile) Web apps in Chrome and it provides already a large set of fine-grained performance metrics, such as largest contentful paint, first input delay, etc. However, at the time of writing, there is no support for either (i) estimating the energy efficiency of web apps or (ii) metrics dedicated to the computation overhead due to the presence of non-content-related components, such as ads and analytics. For example, Google’s Core Web Vitals\(^\text{28}\) might be expanded with a metric representing the ratio between actual contents and ads/analytics in terms of computation, battery, and networking overhead. We invite browser vendors to invest more in these lines of action in order to better support Web developers in creating greener and more efficient (mobile) web apps in the future.

Moreover, at the time of writing, Google is using their Core Web Vitals metrics also as a factor for computing the ranking of a web page in the Google search engine.\(^\text{29}\) Ranking high in a search engine is a fundamental factor for the visibility and success of many online services. So, including energy-related metrics in the set of Core Web

\(^{27}\) https://developer.chrome.com/docs/lighthouse/overview/.

\(^{28}\) https://web.dev/articles/vitals.

\(^{29}\) https://developers.google.com/search/docs/appearance/core-web-vitals.
Chrome: tends to require more energy to run, a device's battery will drain more displayed on mobile devices. If the foundation on which an ad is based implemented can largely impact the energy consumed when they are results includes the vendors of ads platforms. The way such ads are development process, and search engine ranking systems.

The integration of energy-efficiency metrics in Web development tools, Web toward developing more efficient and generally greener web apps:

<table>
<thead>
<tr>
<th>Ads</th>
<th>Statistical significance</th>
<th>Effect size</th>
<th>Analytics</th>
<th>Statistical significance</th>
<th>Effect size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption</td>
<td>Chrome: ✔️</td>
<td>Chrome: large</td>
<td>Opera: ✔️</td>
<td>Opera: ✔️</td>
<td>Chrome: ✔️</td>
</tr>
<tr>
<td>Full Page Load Time (FPLT)</td>
<td>Opera: ✔️</td>
<td>Opera: small</td>
<td>Opera: ✔️</td>
<td>Opera: ✔️</td>
<td>Opera: small</td>
</tr>
</tbody>
</table>

Vitals metrics would also incentivize the community of Web developers towards developing greener Web apps:

**Implication for search engine vendors**

We recommend search engine vendors to include energy-related metrics as factors for computing the ranking of web apps. This would influence the developers towards developing greener Web apps.

Given the high stakes about having a sustainable Web, the above-mentioned metrics should not be defined by individual search engine vendors, but rather they should come from an independent body who has already experience in the area. Energy-efficient Web apps have been the subject of investigation for many software engineering developers for several years now [12]. Researchers in the field have years-long experience in designing and conducting experiments and solutions for making the Web greener and more energy-efficient. So:

**Implication for researchers and search engine vendors**

We call for a close collaboration between software engineering researchers and search engine vendors for the definition of accessible, explainable, and replicable metrics for representing the energy efficiency of Web apps.

Recently, a first step towards making the Web more energy-efficient is the definition of the Web Sustainability Guidelines by the Sustainable Web Design Community Group of the W3C standardization body. Those guidelines might be a good starting point for the definition of energy-efficiency metrics for Web apps. The integration of those metrics into a unified Green scoring system might be a first step towards the integration of energy-efficiency metrics in Web development tools, Web development process, and search engine ranking systems.

Another group that needs to consider the implications of our results includes the vendors of ads platforms. The way such ads are implemented can largely impact the energy consumed when they are displayed on mobile devices. If the foundation on which an ad is based tends to require more energy to run, a device’s battery will drain more quickly. Therefore:

**Implication for Web developers and ads platform vendors**

The overhead imposed by ads is either non-statistically-significant or has a small effect on the overall performance of Web apps.

About energy-aware ads, they are made possible thanks to the Battery Status API, which is currently available in some (but not all) major mobile browsers. It is important to note that in this context we are suggesting to use the Battery Status API exclusively for tuning the served ads with respect to their energy/computation demands; web developers and ads platform vendors must not use such API for other purposes, e.g., users fingerprinting or unethical behaviours [22,23].

As shown in Table 14, the results of the Wilcoxon Rank Sum test conducted on the FCP metric do not allow us to make conclusive claims about the impact of ads on such metric. In contrast, the statistical analysis performed on the FPLT metric revealed a statistically significant difference in the time required for fully loading web pages between the original version of the web app and its version without ads. Consequently, we can infer that:

**Implication for ads platform vendors**

We recommend ads platform vendors to be mindful and adopt processes that are as energy-efficient as possible while maintaining the desired level of quality for the served ads. Possible solutions for achieving the suggested recommendation include:

- provide energy-aware ads (and analytics, but less prominent), where energy-demanding ads (e.g., those based on a video or on an interactive mini-game) are served only when the battery of the smartphone of the user is above a certain threshold;
- perform testing campaigns dedicated to the energy consumption of the served ads on different devices and browsers (also for analytics, but to a lesser extent); this will potentially lead to the detection and fixing of the energy inefficiencies;
- to include a clear sustainability policy for the ads platform; this might also drive responsible Web developers towards the usage of more sustainable ads platforms, injecting a virtuous cycle towards a greener Web;

Table 14
Summary of the obtained results.

<table>
<thead>
<tr>
<th>Ads</th>
<th>Statistical significance</th>
<th>Effect size</th>
<th>Analytics</th>
<th>Statistical significance</th>
<th>Effect size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption</td>
<td>Chrome: ✔️</td>
<td>Chrome: large</td>
<td>Opera: ✔️</td>
<td>Opera: ✔️</td>
<td>Chrome: ✔️</td>
</tr>
<tr>
<td>Full Page Load Time (FPLT)</td>
<td>Opera: ✔️</td>
<td>Opera: small</td>
<td>Opera: ✔️</td>
<td>Opera: ✔️</td>
<td>Opera: small</td>
</tr>
</tbody>
</table>

31 https://w3c.github.io/battery.
lead to significant negative consequences in terms of users’ retention, conversion rates, and ultimately on the success of the provided services [24]. So, it is expected that ads platform vendors are heavily optimizing their served ads against performance overhead. So, the time that is required for the first components of a web app to be downloaded, parsed, and executed by the browser is lightly impacted by the presence of ads. Since a user is able to partially operate on a web app, even if not all the application resources have been retrieved, we assume that the observed difference on FPLT between full and no-ads versions of a web application does not make a large difference on how the app is used in practice. However, this is only an assumption that is based on results retrieved using automation software. Performing an experiment that explores the practical impact of ads on the usage of web applications by real user via a large-scale user study is part of future research work.

7.2. Impact of analytics on the energy consumption and performance of mobile web apps (RQ3 and RQ4)

The results of the Wilcoxon Rank Sum test indicate that the impact of analytics on the energy consumption of web apps is statistically significant for Chrome, but not for Opera. This result confirms our initial assumption that the use of analytics might lead to higher energy consumption, but only for Chrome.

### Implication for Web developers

The overhead imposed by analytics is statistically-significant (with a medium effect size) in Google Chrome, but it is not statistically-significant for Opera.

Since this result is valid only for one of the two browsers we used for our experiment, it is possible that the impact of analytics on the energy consumption of web apps is greatly affected by the underlying browser mechanisms that are used to properly operate such logic (this result is also evident when we look at the results related to the individual web apps — see our replication package). According to this reasoning:

### Implications for browser vendors and analytics platform vendors

We recommend browser vendors to optimize the energy efficiency of the APIs used by analytics platform vendors (e.g., the access to the fields of the Navigator DOM object*) in order to reduce the impact that analytics have on the consumption of a mobile device’s battery, and, consequently, improving the experience of users.

Similarly, we recommend analytics platform vendors to test and debug the energy efficiency of the JavaScript/CSS libraries they provide to Web developers for implementing their user tracking and analytics features.

---

a [https://html.spec.whatwg.org/multipage/system-state.html#the-navigator-object.](https://html.spec.whatwg.org/multipage/system-state.html#the-navigator-object.)

In terms of performance, the findings related to analytics are consistent with those we obtained for ads. More specifically, analytics had no statistically significant effect on the first contentful paint (FCP) metric, but significantly impacted the full page load time (FPLT) metric. Consequently:

### 8. Threats to validity

In this section, we discuss possible threats to the validity of the experiment with the intention of evaluating the generalizability and soundness of the experiment results. For this purpose, we follow the classification of threats detailed by Cook and Campbell (1972). [25].

#### 8.1. Internal validity

Threats in this category stem from limitations in the design and execution of the experiment.

**History** — As suggested in [12], during the experiment execution, multiple individual runs were performed for each combination of subject/main-factor/browser. Specifically, we opted to have 10 individual runs for each combination, leading to a total of 540 individual runs for a total of more than 5 h of (manually-supervised) shear execution time. Having repeated runs also helps in achieving higher statistical power, which we recognize could have been even higher if more repetitions were performed. In any case, the number of repeated runs in our experiment is in line with the state of the art, which can range from 3–6 repetitions up to 20–30 [12]. Due to available resources and time, we opted to have 10 repeated runs for each subject/main-factor/browser combination and to have a higher number of repetitions in future replications of the study. Moreover, the whole experiment execution occurred in two parts, with five repetitions conducted in each of them. This was necessary due to limitations of the tablet device that was used for this experiment; managing the charging periods of the device was not possible using Android Runner in this case, thus, in order to avoid complicating the results of the profilers we opted to avoid charging the device at all. We achieved this by having the device unplugged from any source of power and performing the experiment using “ADB over WiFi”.[33] Separating the experiment in two parts was necessary to avoid the device getting fully discharged during the experiment execution; after the completion of the first part, the device was charged to 100%. While performing the experiment we realized that this planning was not enough, as the device’s battery reached low levels many times. For this reason, when the battery was at 30%, we stopped the execution of the experiment, fully charged the device, and resumed the experiment right after using the — progress flag, provided by Android Runner. This approach might have created a less than optimal environment in terms of reliability, as the conditions under which the two parts of the experiment run might have not been the same. We tried to mitigate this threat by charging the device immediately after the first part was complete, and then starting the second part right after the device had fully charged. At the same time, it is possible that outside resources related to ads and analytics, requested by the local web applications, changed during the charging period, albeit the chances are low.

**Maturity** — When a certain web application is loaded by a web browser for the first time, cache data is stored that enables a better performance in subsequent visits of that application. In order to avoid complications in the measurements related to caching, we clear the cache of each browser before every experiment run. Furthermore, the

---

time between runs is set to 30 s, allowing the Android operating system to “kill” any background activity that is not necessary before the next run starts.

**Selection** — The subjects examined in this paper were selected manually from the Tranco list, as explained in Section 4.1. A requirement for their inclusion is that they contain both ads and analytics functionality and that they can be run locally. Initially, the 12 most popular websites in the categories of social media, entertainment, and news were selected. However, we quickly realized that applications involving log-in screens or a type of media streaming required a complicated solution to run locally. Furthermore, some applications were found to have at least one dysfunctional part that was not noticed during the first round of subject selection and testing. For this reason, we performed a second round of subject selection, increasing the pool of considered websites from 100 to 1000. This time, a more thorough testing process occurred, resulting in a set of 9 web applications. In this way, we properly mitigate any threats that could arise from application malfunctions during the execution of the experiment. Even though a thorough selection of subjects has been performed, ensuring that they run locally as expected, we realize that certain functionalities of the original applications may not transfer to a local implementation.

**Reliability of Measures** — There are various OS processes and device settings that could interfere with and affect the execution of each run. The threat of background processes and notifications polluting the results has been mitigated by disabling all system notifications and multiple applications that are not relevant to the experiment. Moreover, the device’s screen brightness is set to the minimum, while location services and Bluetooth are disabled before the experiment begins. The tablet running the experiment was located near the Raspberry Pi, which is connected to a router through an Ethernet cable and acts as the internet source for the mobile device. This setup ensures the effect on internet source for the mobile device. This setup ensures the effect on the experiment results. The solutions employed make use of conventional software. These tools usually have different degrees of precision or ways of measurement were defined very early in the study setting the foundation for the next steps. Moreover, in the experiment planning Section 4 the research subjects, the hypotheses, the dependent variables, and the treatments for our factors were specified.

**Interaction of setting and treatment** — The results of our study apply to mobile Web apps loaded on Android devices only. While designing the study we decided to use an Android device since (i) the Android platform is the most representative one in the mobile ecosystem as it covers more than two-thirds (70.76%) of the global market share of mobile operating systems,34 followed by iOS with only 28.53% and (ii) as confirmed in two recent secondary studies on measurement-based experiments on the mobile Web [12,26], the research community is using almost exclusively Android devices in their experiments, thus allowing us to compare our results to those of other researchers. Hosting web applications locally on a Raspberry Pi and accessing them from a mobile device on the same network leads to unrealistic performance when loading resources of those applications on the tablet. The close proximity to the server that hosts the applications means that the pages are loaded very fast. At the same time, when requesting ads or analytics-related resources from third-party services, the quality of the Internet connection might fluctuate, thus resulting in different performance between runs. To mitigate such limitations, we used a network throttling service, called WonderSharper,35 to set the maximum upload and download rates for all Internet traffic, minimizing potential irregularities caused by the Internet connection. However, we did not throttle the connection between Raspberry Pi and the tablet during our experiment, thus leading to a potential artificially-fast connection between the first network hop and the user.

### 8.3. Construct validity

Limitations related to the way an experiment is constructed, mapping the theory to observation, constitute threats to the construct validity of a research.

**Inadequate pre-operational explication of constructs** — Using the GQM approach, we established the constructs of our research before its execution was considered and conducted. The objective, research questions and ways of measurement were defined very early in the study setting the foundation for the next steps. Moreover, in the experiment planning Section 4 the research subjects, the hypotheses, the dependent and independent variables, and the treatments for our factors were specified.

**Mono-method bias** — There are many available solutions to measure the energy consumption of a mobile device, both software and hardware. These tools usually have different degrees of precision or ways in which the consumption is calculated. In our experiment, we used only one such tool, the BatteryStats plugin of Android Runner. This means that all our results are, possibly, biased to the methodology that this solution utilizes. To partially mitigate such threats, we introduced some data redundancy to ensure that the measurements are consistent throughout the experiment. Furthermore, we performed an assessment of the reliability of the measures. Regarding the web application performance results, the solutions employed make use of conventional metrics, thus avoiding the introduction of threats.

---

34 https://gs.statcounter.com/os-market-share/mobile/worldwide/
35 https://github.com/magnific0/wondershaper
8.4. Conclusion validity

Threats to conclusion validity surround matters which may lead to a doubtful or incorrect conclusion of the experiment.

Low statistical power — In order to mitigate threats to low statistical power, we used a consistent number of subjects across the treatments, specifically 9 web applications. For each app, 3 treatments were applied across 2 browsers. Each treatment was applied 5 times per web app. For each run, each of the measures (EC, FCP, and FPLT) were collected. In total, there were 540 runs conducted. This results to a large data set of measures for the different metrics we explore, increasing the power of the statistical results we conclude.

Violated assumption of statistical tests — To mitigate the threat of breaching the assumptions of the statistical tests used, we assessed the properties of the data sets and decided whether they fulfil all the requirements for the desired tests. For this reason, before proceeding with the statistical analysis using a test that has a normal distribution of the data as a requirement, the Shapiro–Wilk test was performed on all the original data sets in order to evaluate the normality of their distributions. If this was not the case, then we proceeded with a different statistical test, that does not assume the normality of the distribution of the data. The test employed in these cases was the Wilcoxon Rank Sum test.

9. Conclusions

This study highlights the impact of ads and analytics on Android mobile web apps in terms of energy consumption and performance. We sampled nine web apps from publicly-available list of the top websites worldwide and applied three treatments to each of them: with Ads and analytics, without Ads, and without analytics. Then, each version of each web app is loaded multiple times on an Android device via two different browsers, namely Google Chrome and Opera. During the execution of the experiment, we collected measures about the Energy Consumption (EC), First Contentful Paint (FCP), and Full Page Load Time (FPLT) metrics.

According to the statistical analysis of the collected measures, we conclude that:

• Ads significantly impact the energy consumption of mobile web apps loaded in both Chrome and Opera, both with a large effect size.
• Ads significantly impact the full page load time (FPLT) metric of mobile web apps loaded in both Chrome and Opera, but with only a small effect size.
• Ads tend to do not impact the first contentful paint metric of mobile web apps loaded neither in Chrome nor in Opera.
• Analytics have a significant impact on the energy consumption of mobile web apps loaded in Chrome (with a medium effect size), but not in Opera.
• Analytics significantly impact the first contentful paint metric of mobile web apps loaded in both Chrome and Opera, with only a small effect size.
• Analytics tend to do not impact the first contentful paint metric of mobile web apps loaded neither in Chrome nor in Opera.

Based on the results listed above, in Section 7 we provide recommendations to web developers, browser vendors, ads/analytics platform vendors, search engine vendors, and researchers. Possible future work includes replicating the experiment with web applications for which both ads and analytics have been removed from the content. This will help demonstrate whether the power consumption and extra load time scale with the number of ads and analytics. Additionally, we suggest replicating this experiment using iOOS devices, as iOS still covers 28.53% of the global market share of mobile operating systems. Also, even though the FCP metric is related to how the contents of the web app are rendered in the browser, an interesting future direction for this study would be to include the interaction of the user and how it might impact the rendering and displaying of contents in terms of energy- and performance-related metrics. Finally, the results we obtained in this study are focussing primarily on the performance and energy overhead of using ads and analytics from a technical perspective; an interesting future research direction would be to empirically assess how the overall user experience would be impacted when users access mobile web apps with different levels of ads and analytics (e.g., in terms of user engagement, user satisfaction, fatigue, and behaviour in general).

CRediT authorship contribution statement

Christos Petalotis: Investigation, Data curation, Visualization, Writing – original draft. Luka Krumpak: Investigation, Data curation, Visualization, Writing – original draft. Maximilian Stefan Floriou: Investigation, Data curation, Visualization, Writing – original draft. Laréb Fatima Ahmad: Investigation, Data curation, Visualization, Writing – original draft. Shashank Athreya: Investigation, Data curation, Visualization, Writing – original draft. Ivano Malavolta: Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References


