Kaleidoscope: A Crowdsourcing Testing Tool for Web Quality of Experience

Pengfei Wang*†, Matteo Varvello ‡, Aleksandar Kuzmanovic†
*Software College, Northeastern University, China
†Department of Computer Science, Northwestern University
‡Brave Software
wangpf@stumail.neu.edu.cn, varvello@brave.com, akuzma@northwestern.edu

***Abstract*—Today's webpages development cycle consists of constant iterations with the goal to improve user retention, time spent on site, and overall quality of experience. Big companies like Google, Facebook, Amazon, etc. invest a lot of time and money to perform online testing. The prohibitive costs of these approaches are an entry barrier for smaller players. Further, the lack of a substantial user-base can be problematic to ensure statistical significance within a reasonable duration. In this paper we propose Kaleidoscope, an automated tool to evaluate Web features at a large scale, quickly, accurately, and at a reasonable price. Kaleidoscope can test two crucial user-perceived Web features – the style and page loading. As far as we know, it is the first testing tool to replay page loading by controlling visual changes on a webpage. Kaleidoscope allows to concurrently load a webpage in two versions (e.g., different fonts, with vs without ads) that are shown to a participant side-by-side. Further, Kaleidoscope also allows a participant to interact with each webpage version and provide feedback, e.g., respond to a questionnaire previously prepared by an "experimenter". Kaleidoscope supports both voluntary and paid testers from FigureEight, a popular crowdsourcing platform. Using hundreds of FigureEight testers, we validate that Kaleidoscope matches the accuracy of trusted in-lab tests while providing results about 12x faster (and arguably at a lower cost) than A/B testing. Finally, we showcase how to use Kaleidoscope's page loading feature to study the user-perceived page load time (uPLT) of a webpage.**

I. Introduction

The Web has become a de-facto way for billions of people to access the Internet. How to satisfy website visitors has consequently become a key concern for Web developers. To achieve this goal, developers (re)design websites over and over. For example, font sizes are changed to improve a website readability, new buttons are introduced or moved around, ads are placed in new strategic locations, etc. At the same time, there's been a lot of efforts by both industry [7], [9], [10] and academia [14], [15], [26], [30] to study how to improve Web Quality of Experience (QoE).

Web developers often leverage data-driven decision making methods to converge towards a successful website design. For example, if increasing the font size produces an overall increase in the time users spend on site, the change is considered successful. However, the whole process of data-driven decision making, including collecting data, extracting patterns and facts from the data, and utilizing those facts to make inferences that influence decision-making, becomes very difficult due to the complexity of websites. Big companies (e.g., Google and Facebook) spend a lot of time (and money) to perform *online testing* (e.g., A/B testing) and *slow releases* [45], [49]. These approaches lie on the assumption that, with a large enough user-base, a statistical analysis will provide some insights on which feature is better perceived by a website customer base. Nevertheless, Megahan et al. [35] show that only 1 out of 8 A/B tests produce statistically significant results.

Such online testing approaches are not quite suitable for less popular websites or researchers, due to the lack of user base and prohibitive cost. In-lab testing is a common alternative, where participants are invited to interact with different versions of a website and then a questionnaire is proposed. For researchers, bulletin boards in school campuses are the main way to recruit such participants either for the love of science or for potential rewards like Amazon gift cards [37]. This approach is fairly labor intensive and time consuming, and it suffers from an inherent scalability limitation.

However, developing such an automated testing tool is challenging. First, it should be able to test any Web parameters including both styles and page loading at the same time at scale. Previous tools [34], [51] can only test single parameter due to the design limitation. For example, Eyeorg [51] can test user-perceived page load time by showing videosof loading webpages. Other style parameters (e.g., font size, etc.) cannot be tested at the same time since the video may change these parameters. The font size could be changed when we change the video size. Second, the whole process of the tool should be fast, credible and cost little. An ideal tool should be as credible as the in-lab test, and as fast and low cost as the crowdsourcing test. Third, the testing tool should be easy to use for both developers and testers. A simple and powerful tool can help both developers and testers to save time in the test. Developers can concentrate more on Web developing and testers can do more tests within a limited time.

The goal of this work is to build an automated tool to test a website features (font, overall appearance, speed, etc.) quickly and confidently at a large scale with little cost. Our key idea is to build a controlled testing environment within the browser and leverage remote crowdsourced testers. This tool should take as input N versions of a website, target demographics, target Web page load, and a questionnaire. It would then automatically construct the measurement task and collect the testing results.

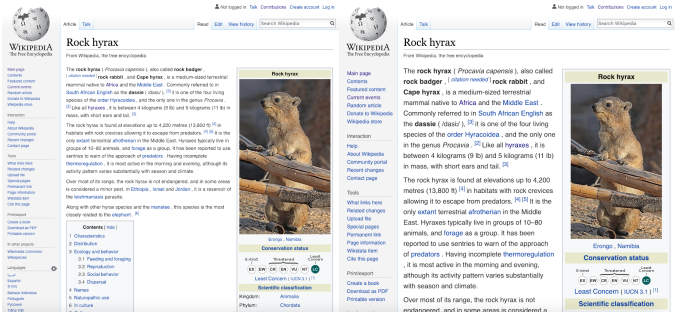


Fig. 1. An integrated webpage in Kaleidoscope, the left is a a version of the webpage and the right is another version

We realize the above idea with Kaleidoscope, to the best of our knowledge the first automatic testing tool which can evaluate nearly all Web features (e.g., font size, font color, image style, layout, page load, etc.) at a client side. Kaleidoscope realizes the above "controlled testing environment" within the browser via a browser extension[1] which fetches N versions of a website to be tested, and locally stores them on the tester machine. This ensures that, regardless of a tester's connectivity, website versions are consistently loaded. Further, it allows fine-grained control on the "speed" at which Web objects are loaded thus emulating different testing conditions (e.g., "network profiles"). Next, Kaleidoscope implements *side-by-side loading* [44], [53] to facilitate the tester's job when offered two different versions of a website under test. This is realized by having the browser extension loading up both versions of a webpage via two iframes (see Figure 1) while respecting the network profiles under test. Tester feedback is collected via customized questionnaires after each test. Kaleidoscope supports both *voluntary* testers, e.g., colleagues and friends, and *paid* testers recruited from popular crowdsourcing platforms like FigureEight [11] and Amazon Mechanical Turk [1].

We first evaluate Kaleidoscope by comparing its *accuracy* with in-lab testing realized via trusted participants. We study a classic Computer Human Interface (CHI) question [16], [19], [36], [38], [41]–[43]: *What is the best font size for online reading?"* Using the Wikipedia webpage, we change the main text to 5 different font sizes ranging from 10 up to 22pt. The experiment verifies that Kaleidoscope matches the accuracy of trusted in-lab tests, and both studies show the similar preference trends of font size for online reading. Next, we evaluate Kaleidoscope's accuracy when compared to classic A/B testing. In this case, we generate a new version of our group webpage and test it both via Kaleidoscope and A/B testing, separately. We find that Kaleidoscope is much faster (more than 12 times faster in this case) than A/B testing, and Kaleidoscope can reach to a more statistically significant result relative to A/B testing. At the end, we show the page load feature of Kaleidoscope with a Wikipedia webpage. We set different showing time for different parts of the webpage, and conduct the experiment to study which parts are more important to improve user-percieved page load time (uPLT). The result indicates the main text content is more important contrasting with the other auxiliary content (*i.e.,* the navigation bar) in improving uPLT.

In providing a comprehensive solution for testing Web features at a large scale, we make the following contributions.

- We present Kaleidoscope, the first crowdsourcing testing tool, implemented through browser extension, to evaluate Web QoE at a client side at a large scale.
- Kaleidoscope can test any front visual Web features including but not limited to layout, font, font size, line spacing, image, background, color, etc.
- Kaleidoscope is the first testing tool for developers to replay the page loading with considering visual loading features (e.g., speed index, above-the-fold time, user-perceived page load time) on a real webpage.
- We devise a set of quality control methods for Kaleidoscope to ensure the testing result quality, including hard rules, engagement, control questions, etc.
- Extensive experiments are conducted to show how Kaleidoscope can help small entities to test Web at scale quickly and accurately with low costs. Besides, Kaleidoscope does not impact websites' revenues and normal operations since we do not modify direclty on the running websites.

The remainder of this paper is structured as follows. Section II introduces some background and motivate our work. In Section III, we detail Kaleidoscope's design and implementation. Section IV extensively evaluates Kaleidoscope's performance. Section V summarizes the state of the art, and we finally concludes the paper in Section VI.

II. Background and Motivation

In a Web context, A/B testing or *split* testing consists of generating two competing versions of a webpage and analyze how they perform, e.g., if they get served faster, attract more customers, etc. Let's assume that a website decides to test a new template for its landing page. With A/B testing, half of the traffic is shown the original version of the page (the control) and half is shown the modified version of the page (the variation). Over time, statistical analysis of, for example, engagement data sugwgests whether the *variation* is successful or not.

Many variations of A/B testing are possible. For example, users can be regular visitors (as in the above example) or paid participants; explicit questions can be asked or statistical analysis might be needed to draw some conclusions from an A/B test. A/B is a fundamental component of the Web development cycle to ensure that every change to a website produces positive results. However, it also suffers from some well-know limitations that we discuss next.

Data collection and analysis— The positive/negative outcome of a variation is computed using different metrics which

[1]https://chrome.google.com/webstore/detail/kaleidoscope/kepcffndaflpplpdenjhmepadhjjcbpc

depends on the variation under test. For example a protocol change like switching from http/1.1 to http/2.0 is measured by the page load time, or how fast a webpage is loaded. A novel marketing strategy would instead be measured in terms of the number of clicks, time spent on site, etc. This implies a non-negligible effort in designing the data collection and analysis associated with each specific A/B test. Therefore, A/B testing is not a panacea, and not all answers can be found via it. This effort is simplified in presence of in-lab experiments where direct questions can be asked to some (paid) participants. The drawback here is, however, the high cost, logistic complexity, and limited scale.

Negative impression— The extent and duration of A/B testing can leave a permanent mark on a website. Users might be drawn away from unsuccessful A/B tests, and search rank might also be impacted. For the latter, Google has highlighted some precise guidelines to be respected when doing A/B testing, *e.g.*, avoid abusing cloaking [39], duration recommendation, user-base size, etc. Besides, A/B testing usually changes the current running websites. Hence, such modifications might affect the profitability of the website business.

Experimental settings— In addition to the potential complexity of having to test multiple variations, the experimental settings are also relevant to an A/B test. For example, PLT largely fluctuates based on the time of the day, network location, and device characteristics. Unless in-lab testing is conduced, these "settings" are indeed variables that are not under control of an experimenter. They thus need to be taken into account, further increasing the testing complexity.

Overall, we can summarize that A/B testing is a powerful and useful tool in the hand of Web developers. However, its complexity is quite high, which in turn limits its applicability mostly to big entities (Google, Facebook, etc.) who can dedicate large teams to continuously integrate A/B testing in their development cycles. Smaller entities on the Web just lack the resources and expertize to perform such tests. We design Kaleidoscope to take advantages of both A/B testing and in-lab testing, as we elaborate in the next section.

III. Kaleidoscope Design and Implementation

This section presents the design and implementation of Kaleidoscope, a crowdsourcing testing tool that evaluates the Web features with the power of crowdsourcing at a large scale. In the reminder of this section, we first provide an overview of Kaleidoscope, then dig into the details of its design and implementation.

A. Rationale and Overview

Our goal is to build a testing tool which allows to simplify the A/B testing pipeline making it largely accessible. One key insight from the previous section is that in-lab testing largely simplifies A/B testing (settings control, explicit feedback, ease of data collection) but it suffers from high cost and complexity—paid participants need to be physically located in the lab–which inherently limits its scalability. Our rationale

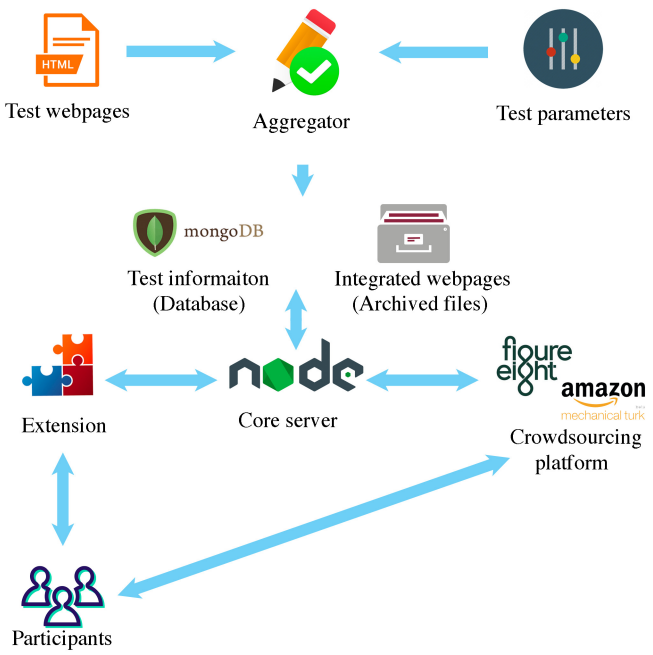


Fig. 2. The structure and workflow of Kaleidoscope

is thus to leverage the benefits of in-lab A/B testing without requiring paid participants to be physically in the lab.

To do so, our key idea is to use browsers as labs where to perform A/B testing. As we will detail later, today's browsers are powerful enough to enable complex A/B tests where different website versions and page loading features can be tested. Further, the large penetration of Web browsers ensures ease of recruiting, e.g., by integrating with existing crowdsourcing platforms such as FigureEight and Amazon Mechanical Turk.

Figure 2 shows the basic structure and workflow of Kaleidoscope. Kaleidoscope inputs N versions of a webpage and related test parameters, and utilizes crowdsourcing platforms to recruit participants. To overcome the known problem in A/B testing and in-lab testing, (i) Kaleidoscope asks questions to testers directly instead of analyzing the implicit data. We also use a side-by-side comparison approach (see Figure 1) to help testers understand the Web features more easily, especially for testing page load speeds [21], [24]. (ii) Kaleidoscope is an independent testing tool and has no impact on the current running website (business). (iii) Kaleidoscope has a more controllable test environment with implementing the browser extension client. For example, a participant will check all versions of a webpage in a test, and a visual page load method is also devised to eliminate the networking discrepancy among different participants. Beyond that, Kaleidoscope solves the high cost, logistic complexity, and limited scale of in-lab testing.

To test N different verisons of a webpage designed by Web developers, they are input to Kaleidoscope. It creates a set of integrated webpages based on it, and stores them

TABLE I
TEST PARAMETERS

Notation	Type	Explantation
test_id	string	The test identification
webpage_num	int	The number of test webpages
test_description	string	The description of a test
participant_num	int	The number of participants involved in the test
question	array	The asked questions during the test
webpages	array	The basic information of all test webpages
web_path	string	The relative folder path of a test webpage
web_page_load	int	The page load simulating value
web_main_file	string	The initial html file name of a test webpage
web_description	string	The description of a test webpage

in the database and storage system. Then, the crowdsourcing platform recruits a crowd of participants to perform the test according to the test information provided by Kaleidoscope. Each participant is asked to install a browser extension to perform the test. During the test, the integrated webpages are shown to the participant in sequence. Questions are asked to the participant after every integrated webpage is displayed. All responses from participants are collected, and Web QoE is analyzed.

Kaleidoscope consists of three main components: *aggregator* , *core server*, and *browser extension*. Kaleidoscope's aggregator is responsible to generate integrated webpages and store user feedback in the storage system. The aggregator uses a webpage composition (html, css, js, etc.) and the experiment input (parameters under test) to generate integrated webpages to be tested. The detailed design and implementation of aggregator are given in subsection III-B. The main functions of the core server are to send the test task information (e.g., test id, task instruction, etc.) to the crowdsourcing platform (e.g., FigureEight, Amazon Mechanical Turk, etc.), provide resources to browser extension, collect results from participants, and conclude the final Web QoE measurement results. The browser extension is mainly aimed for participants to conduct the test. It downloads the integrated webpages locally, monitors participants' behavior and uploads the test data to the core server. Next, the three main components of Kaleidoscope are explained separately.

B. Aggregator

We do the test data preparation in aggregator . In short, two kinds of test data should be prepared and stored in the system – *test information* and *integrated webpages*. Both test information and integrated webpages are generated from the inputs of test webpages and test parameters. The test information includes test id, comparison question(s), the number of participants, the description of the test, etc. The test id is mainly for Kaleidoscope, crowdsourcing platforms and participants to identify a specific test. The comparison questions are asked after each integrated webpage is shown during the test . We only store the comparison questions since we always compare two different versions of a webpage side by side, so the response from the participant must be one of the three– "Left", "Right" and "Same". The number of participants shows how many participants are required to be recruited in the test. We developed an initial HTML document which has two iframes side by side for integrated webpages (see Figure 1), and each iframe links to a version of the test webpage. In addition, we also take other factors into consideration such as simulating page load, simplifying the download files, etc. To use Kaleidoscope, test webpages and test parameters should be input to the system. Next, we elaborate how test webpages and test parameters look like. After that, we show how to modify test webpages, create integrated webpages, and store the test data.

How are test webpages organized? Consider a static webpage saved from a browser — that is how a test webpage is organized. A test webpage (a version) has an initial html document and its related resources including images, js, css and other necessary files. All these resources of a webpage are stored within one folder, and it also may have subfolders. Meanwhile, all test webpages are included in one parent folder.

How do test parameters look like? Test parameters contain the information of both the test itself and test webpages. We adopt JavaScript Object Notation (JSON) format [5] to store test parameters since it is easy for humans to read and write, meanwhile easy for machines to parse and generate. We also develop a tool (Web interface) to help users to generate such format test parameters. Users can input parameter one by one according to the hint. Due to the page limits, we won't provide details of it. We illustrate the details of test parameters in Table I. The value of the key "webpages" contains the test webpage information. Besides, all other keys are used for describing the test, and they will be explained below.

How to modify test webpages? Before generating integrated webpages, the original test webpages are modified. Static webpages usually have a large amount of resource files and folders. Moreover, test webpages may have the same resource files, which means they could have the same file names. Such complexity makes it much more difficult to merge two test webpages together when we create an integrated webpage. In addition, we cannot interact with the operating system directly in a browser extension[2]. To make test webpages easy and feasible for the browser extension to download, all resource files of a test webpage are compressed within one html document (file) borrowing the power of SingleFile [8]. To simulate page load process, we inject a JavaScript function, developed by us, to the compressed test webpage with the given page load simulating parameters (*i.e.,* "page_load" in "webpages").

How to simulate page loading? Page Load Time (PLT) is defined as the JavaScript "onload" event where all of the external content of a webpage has been fetched and evaluated. However, it does not work from a user's perspective. For example, a webpage can be ready although its below-the-fold content has not been loaded totally. The current alternative metrics are mainly based on visual perspectives, e.g., Time to

[2]Browser extensions cannot access local files (e.g., create folders, decompress files, etc.) for security reasons.

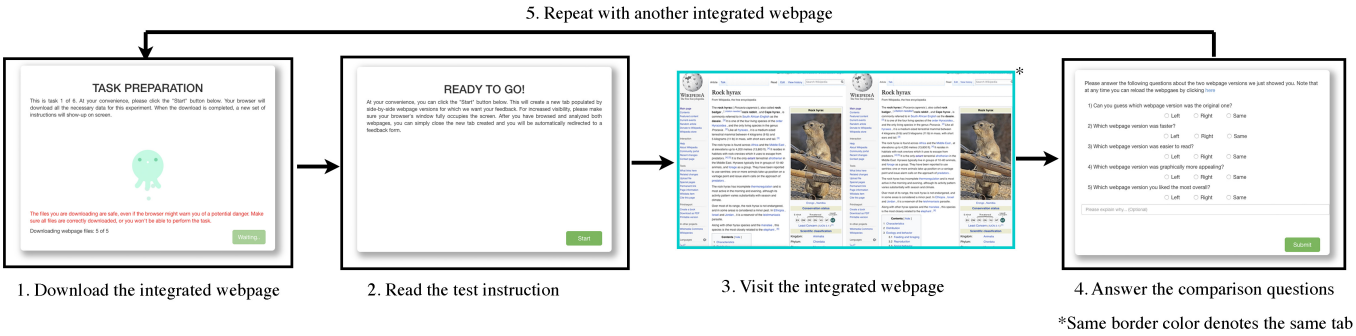


Fig. 3. Test flow of browser extension

First Meaningful Paint (TTFMP) [20], Above-the-fold Time (AFT) [13], [60], Speed Index [29] and user-perceived Page Load Time (uPLT) [34], [48], [51]. So the highlight here is that we devise a method to replay the visual changes of a webpage. In Kaleidoscope, we control the display time of contents (DOMs) by injecting the JavaScript function to the test webpage to simulate loading webpages. The function first hides all DOMs, then shows them based on the given simulating parameters. We utilize the value of the key "web_page_load" in test parameters to simulate page loading. For example, all DOMs will be displayed randomly within 2000 milliseconds when "web_page_load" is set to 2000. A more complicated way is to set specific display times of different DOMs in a test webpage. The value of "web_page_load" is a JSON array in this case. For every element in the array, the key is the locator of a DOM, and the value is when the DOM will be displayed. For instance, ["#main":1000,"#content p": 1500] means "#main" will be shown after 1000 milliseconds, and "#content p"' will be displayed after 1500 milliseconds. Moreover, we can replay a real world page load with this methodology. One can first record the video of loading a real world webpage within a browser. It can be done via many tools such as PhantomJS [40], Chrome Devtools [27], and FFmpeg [2]. Then, the values of "web_page_load" are set according to the display times of the real world page load – which parts are shown at what time.

How to generate integrated webpages? We generate an integrated webpage for every two different test webpages. As mentioned above, each integrated webpage includes two test webpages side by side. We develop an initial html document including two iframes side by side for the integrated webpage. Each iframe is directed to one test webpage. According to the combinatorial theory, we have C_N^2 different integrated webpages given N test webpages. Each integrated webpage includes a developed initial html document and two different test webpages. Similar to the test webpage, all resources of an integrated webpage are stored within one folder. Finally, the aggregator saves all integrated webpages in the storage system and updates the information in a database.

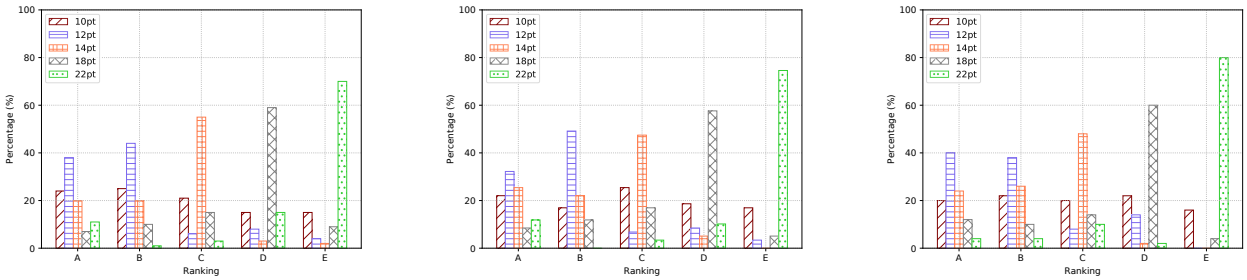
How to store test data? Using MongoDB [6], a scalable and flexible database system, we deploy a non-relational database to store the test data. We created three collections in MongoDB, which are similar to tables in SQL, yet have no structure. The three collections store information of integrated webpages, basic test information, and responoses from participants, separately. Meanwhile, all resource files of integrated webpages are saved in the storage system. We create a new folder which is named after the test id, and all related files of integrated webpages are stored in it. The core server can access these resources, and serve them directly to participants.

C. Core server

The core server is the key element connecting the test resources, browser extension, and crowdsourcing platform. It has four main functions: post the test task to the crowdsourcing platform, provide test resources to the browser extension, collect responses from participants, and analyze the final results. The core server is built as a Web server using NodeJS – an event-driven architecture capable of asynchronous I/O. Such architecture can optimize throughput and scalability in a Web server. After the aggregator finishes preparing all essential test data, the core server sends the test information to a crowdsourcing platform, and the platform recruits participants to perform the test. In this paper, we use the crowdsourcing platform FigureEight, previously called CrowdFlower, to recruit participants. It is easy to extend Kaleidoscope to other crowdsourcing platforms since the development processes are similar for different platforms. All test online resources are fetched from the core server, and all test responses from participants are uploaded to the core server via Ajax [54].

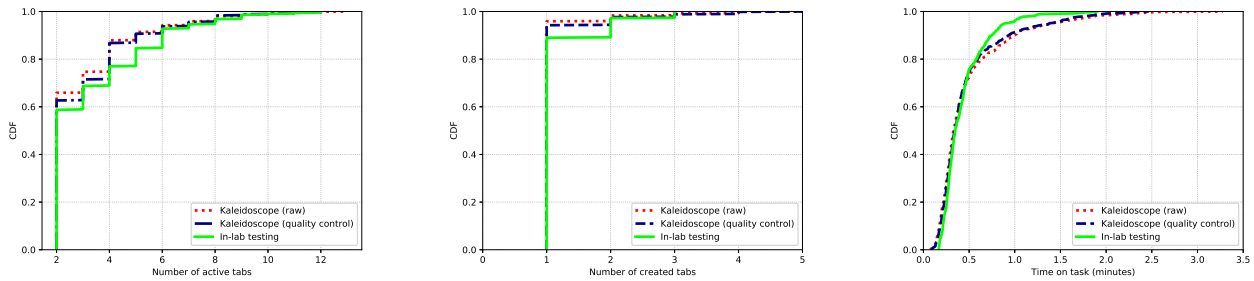
D. Browser extension

The browser extension is the main place for participants to execute the test task. Chrome has become the most popular browser in recent years [52], so we develop it as a Chrome extension. In addition, browser extensions are similar (e.g., Firefox, Edge, etc.), so we can develop browser extensions for other browsers easily. Firefox has written instructions on how to port a Google Chrome extension to Firefox [22], and Edge plans to accept all Chrome extensions in the near future [58]. Our browser extension is available in the Chrome Web Store, and participants also can download the extension by themselves.



(a) Kaleidoscope (raw) (b) Kaleidoscope (quality control) (c) In-lab testing

Fig. 4. Kaleidoscope vs in-lab testing—question feedback



(a) CDF of the number of active tabs (b) CDF of the number of created tabs (c) CDF of time spent per overall task (minutes)

Fig. 5. Kaleidoscope vs in-lab testing—testers behavior.

Before the test, the browser extension asks participants to provide the test information (*i.e.,* test id, contributor id) acquired from the crowdsourcing platform and standard demographic information (*i.e.,* gender, age, country, and self-assessed technical ability) of themselves. We collect the demographic information at a coarse enough granularity, so it is no danger of identifying individual people. Meanwhile, a detailed instruction of how to perform the test is also given to the participants.

Performing test After all of the above are done, the test will be started. A basic flow of performing a test in the browser extension is shown in Figure 3. The participant first downloads an integrated webpage, then visits it by opening a new tab. A set of comparison questions are required to be answered after each integrated webpage is visited. The integrated webpage can be revisited as many times as one wants in order to ensure the participant observes everything clearly. The above process is repeated until all integrated webpages are tested completely. In the end, all test results from the participant are uploaded to the core server. It is easy to know that C_N^2 integrated webpages should be visited given N test webpages. We also utilize sorting algorithms (e.g., bubble sort, insertion sort, etc.) to reduce the number of integrated webpages when only one comparison question is asked. We omit details for space constraints.

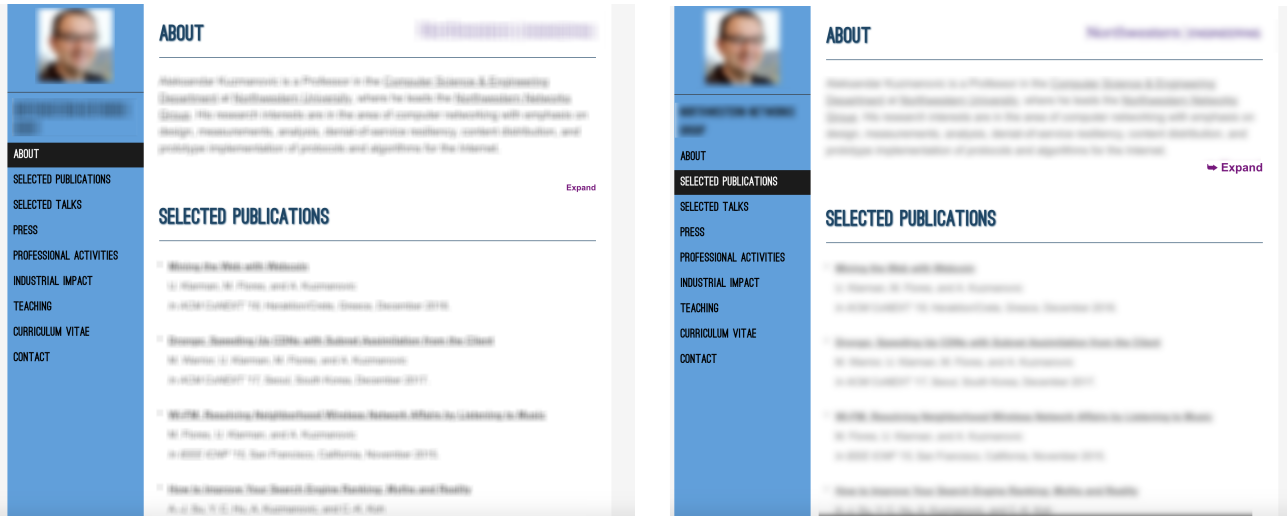
Quality Control We devise a number of approaches to ensure the quality of a crowd worker, and hence the quality of the test. We have *hard rules* for every Kaleidoscope test, and participants must follow them to complete the test. For example, participants must answer all comparison questions in order to move to the next integrated webpage. In addition, the *engagement time* a worker spends on a test is a rough indication of the quality of their work [46]. On the one hand, a short time indicates an unengaged worker; on the other hand, a long time might indicate that the work is distracted. We record how long participants spend on each test, how many times they open the test tabs and the active tabs, etc. Besides, we use *control questions* to help us to verify the quality of participants. It is a common technique in crowdsourcing experiments to randomly insert questions with known answers [31]. To be specific, we occasionally show two copies of the same version webpages , or two significant different webpages to check participants' behaviors. We also follow the *crowd wisdom* – the majority vote of all responses presents the pseudo-ground truth. Participants whose responses deviate from it significantly can be dropped [17], [57].

IV. Evaluation

In this section, we evaluate Kaleidoscope. First, we validate Kaleidoscope's accuracy with respect to both trusted in-lab testing (Section IV-A) and A/B testing (Section IV-B). Next, we use a simple case study to show Kaleidoscope's page load feature (Section IV-C).

A. Kaleidoscope vs In-lab Testing

The goal of this section is to validate the *accuracy* of the crowdsourced responses that Kaleidoscope collects in com-



(a) The original version (A)

(b) The variant version (B)

Fig. 6. Test webpages

parison with a "pseudo" ground truth collected from trusted participants under in-lab testing. We choose to study an *old* CHI question: *What is the best font size for online reading?* We choose this question since it is simple enough and well studied. Many Computer Human Interface (CHI) studies [16], [19], [36], [41] suggest it is good to have a font size with 12 points or 14.points for online reading, and a larger font size [38], [42], [43] could be more friendly to dyslexia people.

1) Methodology: We use a Wikipedia webpage as a test case since it is text-heavy and commonly used in previous CHI studies [42], [43]. Precisely, we chose the "rock hyrax" Wikipedia page[3] since it relates to a topic of general interest, neither technical nor purely academic. Next, we download the webpage from Wikipedia website and generate 5 versions with variable font size of the main text (10pt, 12pt, 14pt, 18pt and 22pt).

In Kaleidoscope, for each of the above webpage versions, we set the same page loading setting (3 seconds, as the original page load time when accessing the original page from our premises). The comparison test question is: *Which webpage's font size is more suitable (easier) for reading?"* Then, we generate the JSON-based test parameters which we input, along with the above 5 webpage versions, to the aggregator to generate the test information and integrated webpages for Kaleidoscope testing. The browser extension also generates extra integrated webpages to control quality. The two test webpages in these integrated webpages are either exact same or significantly different, so we can know the right answers beforehand. For example, the browser extension hold an integrated webpage where two test webpages are same; we also have an integrated webpage in which the main font size of one test webpage is 4 pt and the other test webpage's main font size is 12 pt. Each recruited participant will compare at most 11 integrated webpages, and one of them is for quality control.

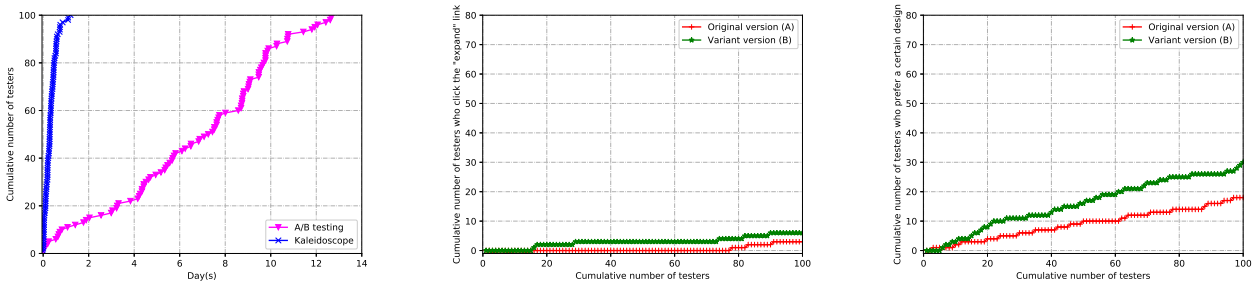
[3]https://en.wikipedia.org/wiki/Rock_hyrax

We still leverage Kaleidoscope (with the same configuration) for the in-lab settings, but we further spend time with the participants to carefully explain each step of the test.

For Kaleidoscope, we recruit 100 "historically trustworthy" testers from FigureEight, which ensures the high quality of responses. The total cost of this test is \$11 (\$0.11 for each participant) or \$0.01 for each side-by-side comparison. It takes about 12 hours to collect all 100 responses from participants. As trusted in-lab participants we recruit, over one week, 50 friends and colleagues who promise full commitment to the test.

2) Results: Figure 4(a), 4(b), Figure 4(c) show the statistical results of the raw Kaleidoscope testing, the Kaleidoscope testing with quality control and the in-lab testing separately. The x axis represents the ranking level for online reading, where "A" denotes the font size is best for reading, and "E" denotes the worst. After comparing all three figures, we conclude that the preference trends of font size for online reading are similar. It means Kaleidoscope can obtain a very good result contrasting with the pseudo-ground truth, and historically trustworthy feature of FigureEight does well in recruiting trusted participants. Besides, the result of Kaleidoscope with quality control is much closer to the in-lab testing than the raw Kaleidoscope testing. For instance, all three figures show that most participants vote 12 points as the "A" ranked font size. However, the second one is 10 points in Kaleidoscope (raw), this is different from results of both Kaleidoscope (quality control) and in-lab testing where the second one is 12 points. It indicates that Kaleidoscope with quality control can reach to a finer-grained accuracy than the raw Kaleidoscope. In addition, the results of Kaleidoscope with quality control and in-lab testing are closer to the CHI studies [16], [19], [36], [41] which suggest 12 points and 14 points are optimal font size for general people to read online.

During the experiments, we also ask participants the consent



(a) Evolution over time of the total number of testers.

(b) A/B testing result

(c) Kaleidoscope result of question C

Fig. 7. Kaleidoscope vs A/B testing

to monitor their browsing behavior. Specifically we record how many tabs they create, how frequently the active tab is changed, and the duration of each side-by-side comparison. The goal is to capture a participant "engagement" and compare trusted in-lab participants with (remote) paid participants. Figure 5(a), 5(b), and 5(c) summarize the results in form of Cumulative Distribution Functions (CDFs) computed per side-by-side comparison, e.g.,Figure 5(c) shows the CDF of the duration of each side-by-side comparison when distinguishing between Kaleidoscope (raw and filtered via quality control) and in-lab testing. Overall, the tester behaviors are similar when comparing trusted and paid participants, even more when filtering via quality control. This further validates the quality of the results that Kaleidoscope can provide as well as the effectiveness of the quality control in detecting and removing participants with abnormal behaviors. For example, the longest side by side comparison in the raw data lasted 3.3 minutes; after filtering, this value reduces to 2.5 minutes which is much closer to 1.9 minutes measured via in-lab testing.

B. Kaleidoscope vs A/B Testing

The goal of this section is to validate the performance of Kaleidoscope in comparison with the more classic A/B testing. We do so by leveraging the landing page of our research group since it is the only website we own with some daily traffic, and A/B testing has to be run on a real website.

1) Methodology: Figure 6(a) shows both the "A" (original) and "B" versions of our research group webpage. We blur out the parts of the webpage that contains information not related to the test. Our official group webpage includes 9 sections including "about", "selected publications", "selected talks", "press", etc. Some sections are not shown to save spaces unless a visitor click the "Expand" button at the right end side of the currently visible section. Due to the little visibility of the "Expand" button, we opted to generate a "B" version of the page where: 1) the text's button is 1.5 times larger, 2) it is enriched with a captivating symbol, 3) it is positioned closer to the main text.

With A/B testing, we need a method to infer which page version is more appealing to our visitors. We do so by monitoring how likely a visitor is to click on the "Expand" button based on the version served. With Kaleidoscope, we instead ask the following explicit questions: *which webpage is graphically more appealing?* (question A), *which version of the 'Expand' button looks better?* (question B), and *which version of the 'Expand' button is more visible?* (question C). "A" (original) and "B" versions of our research group webpages are aggregator 's input (as Figure 7), along with the the same page load settings (3 seconds).

For each test, Kaleidoscope and A/B, we consider 100 participants. For Kaleidoscope, we recruit 100 historically trustworthy participants via FigureEight. We pay each participant $0.1 (total cost: $10). We serve the "A" (original) and "B" versions of our research group webpage until we have 100 visitors. At each visit, "A" and "B' versions are served with equal probability randomly. For privacy concerns, we only record if the visitor clicks the "Expand" button and which version of webpage (s)he is browsing.

2) Results: We first compare the time required to complete our experiment via Kaleidoscope and A/B testing, respectively. Figure 7(a) shows that about one day was enough to recruit 100 participants via Kaleidoscope, while 12 days where needed via A/B testing. Note that this is just a demonstrative and simplified analysis as several extra factors are currently ignored. For example: 1) the cost/duration of developing A/B testing, 2) A/B speedup in presence of more popular websites—although we should stress that Kaleidoscope is designed to help less popular websites, 3) Kaleidoscope speedup via higher rewards and/or via additional crowdsourcing websites and parallel campaigns.

Next, we compare results collected via A/B testing and Kaleidoscope with respect to the potential benefit of a new "Expand" button. The results of A/B testing and Kaleidoscope are shown in Figure 7(b) and Figure 7(c) separately. In A/B testing, 51 participants visit the "A" (original) version, and only 3 visitors click the "Expand" button. By contrast, 49 participants visit the "B" version generating 6 clicks. While the number of clicks for the "B" version is increasing, the P value [4] of this A/B testing is only 0.133, which indicates that this A/B test is not significant enough. In other word,

[4]https://vwo.com/ab-split-test-significance-calculator/

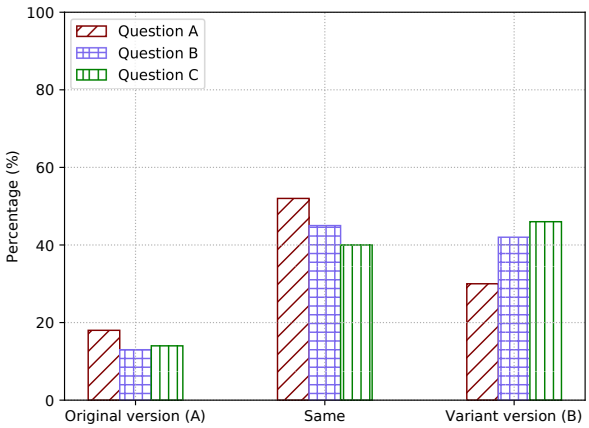


Fig. 8. Responses of all questions in Kaleidoscope

we cannot say (yet) that the new "Expand" button is more visible and more visits (and time) are needed. In contrast, when answering question C in Kaleidoscope we find that 46 participants think the new design is more visible, and only 14 participants believe the original one is better (see Figure 8, question C). The calculated P value of the Kaleidoscope testing is 6.8×10^{-8}, and we can deduce that the new "Expand" button is more visible than the original one at 99% confidence.

Finally, we analyze the responses collected for all three questions via Kaleidoscope (see Figure 8 summarizes). For question A (*which webpage is graphically more appealing?*) the majority of responses (50%) indicate that the two webpages are equally appealing. This is intuitive since the very small variation introduced does not alter the overall look and feel of the page (see Figure 7). For question B (*which version of the 'Expand' button looks better?*), we start observing an increase in positive responses for version "B", *i.e.,* the improved 'Expand' button (question B), although the answer "Same" (45% participants) narrowly edges out the answer "variant version" (42% participants).

C. Page Load Feature

Instead of controlling the out-of-dated metric – page load time, we turn to control the visibility of a webpage in Kaleidoscope. As far as we know, Kaleidoscope is the first tool to replay the webpage with controlling the visual changes on a webpage. Currently, many webpage loading metrics, such as, Time to First Paint (TTFP) [28], Above-The-Fold time (ATF) [18], [48], user-perceived page load time (uPLT) [34], [51], are proposed to focus on measuring the visual changes when loading a webpage.

The goal of this section is to test the page load feature of Kaleidoscope. In this experiment, we make an investigation on which parts of a webpage should be loaded more quickly to improve Web QoE with Kaleidoscope.

1) Methodology: We still use the Wikipedia webpage adopted in the first experiment as the test case. Based on our observation, the webpage can be mainly divided into two parts – navigation bar and main text content. To evaluate which part is more important for user-perceived page load time, we generate two test webpages. They have same style but different page load settings. One version of the webpage shows all elements of the navigation bar in 2 seconds and all elements of the main text content in 4 seconds (version A). On the contrary, the other version of the webpage shows all elements of the navigation bar in 4 seconds and all elements of the main text content in 2 seconds (version B). Both two test versions are load completely (no visual change) in 4 seconds, which means the above-the-fold (ATF) time are same for both two versions. During the test, we ask participants that " Which version of the webpage seems ready to use first?" In this experiment, participants can also tell their feelings of the testing with inputing texts optionally. We recruit 100 participants for this task via FigureEight, and choose participants who are historically trustworthy. Each participant is paid $0.1, and the total cost is $ 10. It takes about 12 hours to collect all 100 responses.

2) Results: The result is shown as Figure 9, and it indicates that participants think the webpage, which main text content is loaded faster, is ready to use first. The result is more significant after filtering out some participants with the quality control method. Specifically, 46% participants think version B is faster in the raw Kaleidoscope testing. By contrast, 54% participants believe version B is faster after using quality control method.

Comments we collected from participants help shed light on the definition of "ready to use". The following three typical comments confirm that the main text content is crucial to the user-perceived page load time (uPLT) than the navigation bar:

> *"The main text of the article was available to read first. "*

> *"Right came fast and came full context instantly comparing to left."*

> *"I could see the text content 2-3 sec faster."*

It is not hard to understand – people usually look for related articles when they visit a Wikipedia webpage, so they focus on the main text content more. Besides, we also find some comments that have different thinkings on user-perceived page load time, and one example is as following:

> *"By browsing and moving are done with the same degree"*

The above comment indicates that that participant only cares about the visual changes of the webpage. Based on these comments and comparison result, we can summarize that most participants care more on the main text content than other auxiliary content on the Wikipedia webpage. In addition, the experiment indicates that user-perceived page load times may be quite different even though other visual page loading metrics, such as above-the-fold time, are same. We only scratch the surface of what can be done with page load feature of Kaleidoscope in this experiment. Kaleidoscope

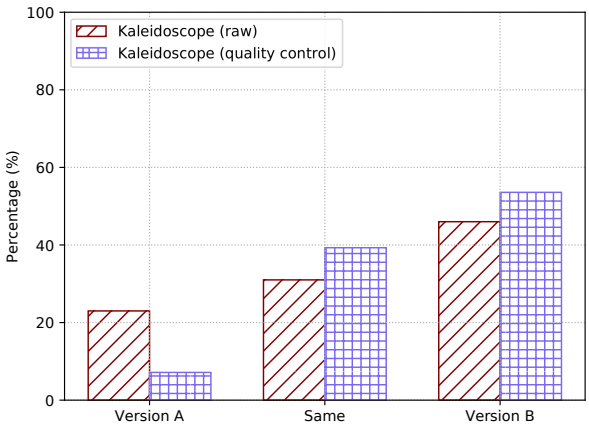


Fig. 9. Result of page load feature

can do more with replaying page loading, e.g., comparing http/1.1 and http/2.0, understanding both style and page load together in a webpage, etc.

V. Related Work

Testing Web has become an essential approach for developers to understand and improve their websites. Based on the recent report [33], the web testing market size is expected to grow from USD 3.50 Billion in 2017 to USD 5.45 Billion by 2022, at a Compound Annual Growth Rate (CAGR) of 9.2% during the forecast period. In addition, testing Web is also becoming more and more difficult due to Web's high complexity and customers' different goals. Therefore, experts from both industry [7], [9], [10] and academia [14], [15], [26], [30] have been keeping working on it to make the testing easier and more efficient.

Tech giants have invested a lot in building their own Web testing platforms. For example, LinkedIn [59] built a powerful and flexible A/B experimentation platform that enables their developing teams across LinkedIn to make informed decisions faster at a large scale. Google also has Google Optimize [3] which offers A/B testing, website testing, and personalization tools to help improve Web experience. Amazon Web Services [50] enable customers to create developer environments and expand testing machine fleet easily with their products. Besides, some specialized testing companies were started to help build such Web testing, such as Optimizely [7], VWO [10], GTmetrix [4], etc. However, the website has to pay a lot to in using third-party tools or developing its own testing tools. Besides, such A/B testing is also not suitable for less popular websites or researchers since the test requires a number of website users.

The Web design and load speed are two main concerns in developing websites. In terms of Web design, a lot of research has been done on how to design a friendly and "good-looking" website from different aspects, such as font [36], font size [42], [43], layout [23], color [12], [25], [47], etc. Recently, Miniukovich et al. [36] proposed concrete design guidelines for Web readability by summarizing 61 readability guidelines in a series of papers in related workshops and conferences. Ferris et al. [25] summarize color theory, demonstrate the effect of color on business, and construct a framework to serve as a reference for Web designers.

Originally, the page load user experience was measured using a simple metric – onLoad, *i.e.,* a browser event indicating that "all of the objects in the document are in the DOM, and all the images, scripts, links and sub-frames have finished loading." Many researches [55], [56] have been done to understand how to improve it from different aspects. Modern webpages are however a complex collection of hundreds of different objects, and researchers found that such metrics struggle in representing the actual user experience. This has motivated a surge of interest in better understanding user experience on Web. Hence, new measuring metrics have been introduced recently. Time to First Paint (TTFP) [28], Above-the-fold Time [18], [48] and user-perceived page load time (uPLT) [34], [51] are three new metrics to measure page load features. All these new metrics are focusing on visual changes of loading a webpage, so understanding how people think when visiting a webpage is more important.

Leveraging crowdsourcing [32] to test the website is an important idea since we can get real responses from real people with little cost. Varvello et al. [51] propose Eyeorg – a crowdsourcing platform for Web QoE measurement. It only measures user-perceived page load time with showing videos of page loading to participants, and collect their responses such as which page loaded faster, when the page is loaded completely, etc. Another uPLT measurement platform WebGaze [34] also adopted the video idea. The advantage of utilizing videos is that it gives a consistent experience to all participants, regardless of their network connections and device configurations. However, it leads to limited visibility, and we cannot interact with it as a common webpage.

VI. Conclusions

In this paper, we presented Kaleidoscope, the first crowdsourcing testing tool implemented through a browser extension to evaluate websites at a client side at scale. Kaleidoscope can test both style and page loading speed at a controllable environment. Unlike A/B testing, Kaleidoscope does not impact the websites' revenues and normal operations since it does not change the currently running website. Instead, Kaleidoscope only stores N versions of the test webpage and test parameters. Then, it automatically constructs the crowdsourcing testing task and collects the testing results from the recruited participants. Finally, we conduct three experiments comparing Kaleidoscope with in-lab testing, A/B testing, and showing a case study of understanding uPLT. Going forward, we plan to make Kaleidoscope a tool that any small entities and researchers can use to evaluate their new Web features, without worrying about the challenges of designing a Web test from scratch.

REFERENCES

[1] Amazon mechanical turk, 2018. https://www.mturk.com/.

[2] Ffmpeg, 2018. https://www.ffmpeg.org/.

[3] Google optimize, 2018. https://optimize.google.com/optimize/home/.

[4] Gtmetrixt, 2018. https://gtmetrix.com/.

[5] Json, 2018. https://www.json.org/.

[6] Mongodb, 2018. https://www.mongodb.com/.

[7] Optimizely, 2018. https://www.optimizely.com/.

[8] Singlefile, 2018. https://github.com/gildas-lormeau/SingleFile.

[9] Speedtest, 2018. http://www.speedtest.net/.

[10] Vwo, 2018. https://vwo.com/.

[11] Why most a/b tests give you bullshit results, 2018. https://www.figure-eight.com/.

[12] Wouter A Alberts and Thea M van der Geest. Color matters: Color as trustworthiness cue in web sites. *Technical communication*, 58(2):149–160, 2011.

[13] Asrese Alemnew, Vassilis Christophides, Renata Teixeira, Dario Rossi, et al. Narrowing the gap between qos metrics and web qoe using above-the-fold metrics. In *PAM 2018-International Conference on Passive and Active Measurement*, pages 1–12, 2018.

[14] Esben Andreasen, Liang Gong, Anders Møller, Michael Pradel, Marija Selakovic, Koushik Sen, and Cristian-Alexandru Staicu. A survey of dynamic analysis and test generation for javascript. *ACM Computing Surveys (CSUR)*, 50(5):66, 2017.

[15] Dennis Appelt, Cu D Nguyen, Annibale Panichella, and Lionel C Briand. A machine-learning-driven evolutionary approach for testing web application firewalls. *IEEE Transactions on Reliability*, 67(3):733–757, 2018.

[16] British Dyslexia Association et al. Dyslexia style guide. *British Dyslexia Association*, 2012.

[17] Jeremy Blackburn and Haewoon Kwak. Stfu noob!: predicting crowd-sourced decisions on toxic behavior in online games. In *Proceedings of the 23rd international conference on World wide web*, pages 877–888. ACM, 2014.

[18] Diego Da Hora, Dario Rossi, Vassilis Christophides, and Renata Teixeira. A practical method for measuring web above-the-fold time. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 105–107. ACM, 2018.

[19] Vagner Figueredo de Santana, Rosimeire de Oliveira, Leonelo Dell Anhol Almeida, and Maria Cecília Calani Baranauskas. Web accessibility and people with dyslexia: a survey on techniques and guidelines. In *Proceedings of the international cross-disciplinary conference on web accessibility*, page 35. ACM, 2012.

[20] Google developers. First meaningful paint, 2018. https://developers.google.com/web/tools/lighthouse/audits/first-meaningful-paint.

[21] Prasenjit Dey, Parvez Ahammad, et al. Perceived performance of top retail webpages in the wild. *ACM SIGCOMM Computer Communication Review*, 47(5):42–47, 2017.

[22] MDN Web Docs. Porting a google chrome extension, 2018. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Porting_a_Google_Chrome_extension.

[23] Rodney Graeme Duffett. Facebook advertisings influence on intention-to-purchase and purchase amongst millennials. *Internet Research*, 25(4):498–526, 2015.

[24] Edge. Comparing websites loading speed visually side-by-side with duoload, 2018. https://edgetalk.net/comparing-websites-loading-speed-visually-side-side-duoload/.

[25] Kevin Ferris and Sonya Zhang. A framework for selecting and optimizing color scheme in web design. In *System Sciences (HICSS), 2016 49th Hawaii International Conference on*, pages 532–541. IEEE, 2016.

[26] Boni Garcia, Francisco Gortazar, Luis Lopez-Fernandez, Micael Gallego, and Miguel Paris. Webrtc testing: challenges and practical solutions. *IEEE Communications Standards Magazine*, 1(2):36–42, 2017.

[27] Google. Chrome devtools, 2018. https://developers.google.com/web/tools/chrome-devtools/.

[28] Google. First meaningful paint, 2018. https://developers.google.com/web/tools/lighthouse/audits/first-meaningful-paint.

[29] Google. Speed index, 2018. https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index.

[30] Huan Gui, Ya Xu, Anmol Bhasin, and Jiawei Han. Network a/b testing: From sampling to estimation. In *Proceedings of the 24th International Conference on World Wide Web*, pages 399–409. International World Wide Web Conferences Steering Committee, 2015.

[31] Tobias Hossfeld, Christian Keimel, Matthias Hirth, Bruno Gardlo, Julian Habigt, Klaus Diepold, and Phuoc Tran-Gia. Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing. *IEEE Transactions on Multimedia*, 16(2):541–558, 2014.

[32] Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[33] MarketsandMarket INC. Web performance market worth 5.45 billion usd by 2022, 2018. https://www.marketsandmarkets.com/PressReleases/web-performance.asp.

[34] Conor Kelton, Jihoon Ryoo, Aruna Balasubramanian, and Samir R Das. Improving user perceived page load times using gaze. In *NSDI*, pages 545–559, 2017.

[35] Justin Megahan. Why most a/b tests give you bullshit results, 2018. https://venturebeat.com/2016/02/08/why-most-ab-tests-give-you-bullshit-results/.

[36] Aliaksei Miniukovich, Antonella De Angeli, Simone Sulpizio, and Paola Venuti. Design guidelines for web readability. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pages 285–296. ACM, 2017.

[37] Novell. Take user tests and earn amazon gift cards, 2018. https://www.novell.com/communities/coolsolutions/sign-take-user-tests-earn-amazon-gift-cards/.

[38] Beth A O'Brien, J Stephen Mansfield, and Gordon E Legge. The effect of print size on reading speed in dyslexia. *Journal of Research in Reading*, 28(3):332–349, 2005.

[39] Optimizely. Ab testing and search engine optimization, 2018. https://help.optimizely.com/Set_Up_Optimizely/AB_Testing_and_Search_Engine_Optimization.

[40] Phantomjs. Phantomjs, 2018. https://github.com/ariya/phantomjs.

[41] Luz Rello, Gaurang Kanvinde, and Ricardo Baeza-Yates. Layout guidelines for web text and a web service to improve accessibility for dyslexics. In *Proceedings of the international cross-disciplinary conference on web accessibility*, page 36. ACM, 2012.

[42] Luz Rello, Martin Pielot, and Mari-Carmen Marcos. Make it big!: The effect of font size and line spacing on online readability. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3637–3648. ACM, 2016.

[43] Luz Rello, Martin Pielot, Mari-Carmen Marcos, and Roberto Carlini. Size matters (spacing not): 18 points for a dyslexic-friendly wikipedia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 17. ACM, 2013.

[44] Jake Rocheleau. Compare websites loading speed (side-by-side) with this tool, 2018. https://www.hongkiat.com/blog/website-speed-testing-duoload/.

[45] First Round. How a/b testing at linkedin, wealthfront and ebay made me a better manager, 2018. https://bit.ly/2AJablU.

[46] Jeffrey M Rzeszotarski and Aniket Kittur. Instrumenting the crowd: using implicit behavioral measures to predict task performance. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 13–22. ACM, 2011.

[47] Daisuke Saito, K Saito, K Notomi, and M Saito. The effect of age on web-safe color visibility for a white background. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 5145–5148. IEEE, 2006.

[48] Antoine Saverimoutou, Bertrand Mathieu, and Sandrine Vaton. Web browsing measurements: An above-the-fold browser-based technique. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1630–1635. IEEE, 2018.

[49] Vlad Savov. Gmails biggest redesign is now liver, 2018. https://www.theverge.com/2018/4/25/17277360/gmail-redesign-live-features-google-update.

[50] Amazon Web Services. Development and test, 2018. https://aws.amazon.com/dev-test/.

[51] Matteo Varvello, Jeremy Blackburn, David Naylor, and Konstantina Papagiannaki. Eyeorg: A platform for crowdsourcing web quality of experience measurements. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 399–412. ACM, 2016.

[52] Steven J. Vaughan-Nichols. Chrome is the most popular web browser of all, 2017. https://www.zdnet.com/article/chrome-is-the-most-popular-web-browser-of-all/.

[53] Lea Verou. Duoload: Simplest website load comparison tool, ever, 2018. http://lea.verou.me/2017/02/duoload-simplest-website-load-comparison-tool-ever/.

[54] w3schools. Ajax introduction, 2018. https://www.w3schools.com/xml/ajax_intro.asp.

[55] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall. Demystifying page load performance with wprof. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 473–485, 2013.

[56] Xiao Sophia Wang, Arvind Krishnamurthy, and David Wetherall. Speeding up web page loads with shandian. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 109–122, 2016.

[57] Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

[58] Mark Wycislik-Wilson. Microsoft confirms you will be able to use chrome extensions in edge, 2018. https://betanews.com/2018/12/10/microsoft-edge-chrome-extensions/.

[59] Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. From infrastructure to culture: A/b testing challenges in large scale social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2227–2236. ACM, 2015.

[60] Goran andrli. What is above the fold time and what to do with it, 2018. https://www.globaldots.com/fold-time/.