LogUI: Contemporary Logging Infrastructure for Web-Based Experiments

Maxwell, D.M.; Hauff, C.

DOI
10.1007/978-3-030-72240-1_59

Publication date
2021

Document Version
Accepted author manuscript

Published in
Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Proceedings

LogUI: Contemporary Logging Infrastructure for Web-Based Experiments

David Maxwell and Claudia Hauff

Delft University of Technology
Delft, The Netherlands
{d.m.maxwell,c.hauff}@tudelft.nl

Abstract. Logging user interactions is fundamental to capturing and subsequently analysing user behaviours in the context of web-based *Interactive Information Retrieval (IIR)*. However, logging is often implemented within experimental apparatus in a piecemeal fashion, leading to incomplete or noisy data. To address these issues, we present the **LogUI** logging framework. We use (now ubiquitous) contemporary web technologies to provide an easy-to-use yet powerful framework that can capture virtually any user interaction on a webpage. **LogUI** removes many of the complexities that must be considered for effective interaction logging.

Keywords: Logging · Framework · Experimental Infrastructure

1 Introduction

Contemporary *web applications* are complex and ubiquitous [17]. At their heart, a series of manipulations are undertaken on the *Document Object Model (DOM)*[1], where *HTML elements* are created and modified during the lifespan of a webpage. *Web-based* experimental apparatus is commonplace within the IIR community to examine an interface's usability and the behaviours exhibited by those who use it. Vital to these studies is the concept of *logging user interactions.* Interaction logs are generated by capturing and recording a user's interactions (or *events*) with webpage(s) during a search and/or browsing session.

Researchers often work on their own web-based apparatus, including their own logging infrastructure. Anecdotal observations highlight that logging is often achieved in a piecemeal fashion, often considered to be an afterthought leading on from the implementation of the main system. However, this is undesirable. Infrastructure can be complex to implement [1], with researchers forgetting to log key events, or misunderstanding implementation nuances. This can lead to low quality logs, with missing and/or noisy data—with the potential for *post-hoc* frustrations when interpreting the data. While attempts have been made to develop logging infrastructure over the years (refer to §2), we have failed to find an easy-to-use (and affordable) solution that considers the necessary complexities to generate clean logs—and is able to exploit contemporary web technologies.

[1] The *DOM* is the tree-like structure of *HTML elements* that constitute a webpage.

As such, we present in this paper the **LogUI** framework. The framework can capture and record high-quality, fine-grained user interaction data over the course of a search and/or browsing session. It can be easily integrated within any existing web application that is run on a contemporary web browser/framework, meaning support on both desktop and mobile platforms is possible.

2 Existing Approaches

While a large number of IIR studies report measures such as click depths (on *Search Engine Result Pages (SERPs)*), mouse trails and movements, dwell times, keypresses, and so forth, descriptions about how the underlying data are captured are seldom provided. Indeed, logging apparatus is often implemented *in situ* within the wider experimental system. A number of logging tools (and associated literature) exist. Phillips and Dumas [16] presented a number of criteria that effective logging infrastructure must comply with.

The mid-2000s to the mid-2010s saw a shift in focus from *platform-specific* [14, 16,21] to *web-based* experimental apparatus, including interaction logging infrastructure that focused on examining the DOM. Examples of solutions from this period included *MLogger* [10], *PooDLE* [5], *Search-Logger* [18], *Wrapper* [13], *UsaProxy* [3, 4], the framework by Hall and Toms [11], *WHOSE* [12], and *YASFIIRE* [20]. Some of these solutions required additional software to be installed (such as browser toolbars), while others made use of an intermediatary *proxy server* to inject logging code, as used in subsequent studies [2, 6, 7, 15].

Despite advancements, these approaches were less than ideal [8]. A second *browser war* led to a rise in prominence for client-side scripting (i.e., *ECMAScript*, or *JavaScript*), and in turn increased the capabilities of browsers. Existing solutions became redundant, with new, *JavaScript-only* solutions such as *ALF* [9]. In the commercial space, tools such as *Google Analytics*, *Hotjar*, *Matomo*, and *eTracker Analytics* are available. While useful, these tools offer either coarse-grained logging (for *SEO*); are prohibitively expensive; are not designed to be loosely coupled (leading to integration difficulties); or use outdated technologies. A more recent solution is *UXJs* [19], but it may still have issues when integrating with modern web applications using frameworks such as *React*. These modern frameworks use JavaScript to *'draw'* elements on a webpage; the fine-grained logging solutions mentioned above do not cater for elements drawn *after* the page initially loads. We believe that there is therefore a pertinent need for logging apparatus supported by researchers within the IIR community.

3 The LogUI Framework

With a high-level overview of **LogUI** shown in Figure 1, we now present a brief discussion of the framework's architecture, highlighting the main components. This includes: the **client**; what can be logged; the **server**; and ease of integration. Note that all circled numbers (e.g., ❶) pertain to the component highlighted with the same number as in Figure 1.

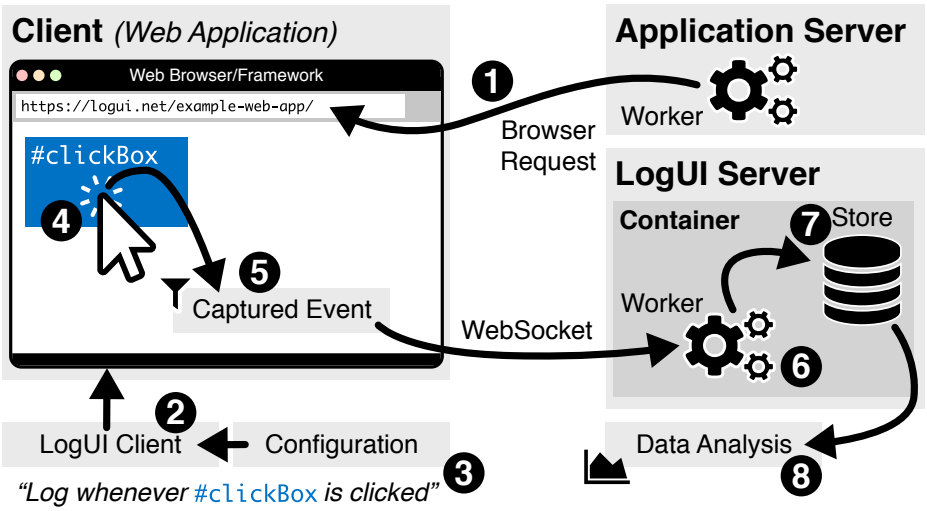

Fig. 1. Architecture diagram of the **LogUI** framework. Refer to §3 for a detailed explanation, along with descriptions of the eight highlighted components.

LogUI Client Library: The **LogUI** client is a JavaScript library that provides advanced functionality for the tracking of events associated with a specific element on a webpage, or associated with the webpage as a whole. To clarify, an *event* pertains to a specific action—such as the click of a user's mouse—on a specific *element*—such as a snippet as presented on a SERP. Events pertaining to the page as a whole could be, for example, the resizing of the web browser's viewport, tracking mouse movements, the scrolling of the page, or the web browser no longer being the user's active window *(losing focus)*. As previously mentioned, **LogUI** can be used in a contemporary web browser[2]; web-based application frameworks such as *Electron* are also supported if required.

Given an existing web application (such as experimental apparatus) where fine-grained event logging is required ❶, one can integrate **LogUI** by including the compiled **LogUI** client library ❷. A configuration object must be supplied to **LogUI** ❸. This tells the client library what elements on the page should be logged—and for what events (see below). When an event occurs ❹, the **LogUI** client then packages up the data for the event (along with any specified *metadata*, see below) ❺, and sends the packaged data down the established *Websocket* connection to the listening **LogUI** server worker process ❻.[3]

Loggable Elements and Events: Standardised *CSS Selectors* are used in the configuration object ❸ to allow for the selection of *any* element within the DOM. Any standardised DOM events can also be used (such as `mouseover` or `keyup`). As mentioned, page-wide events can also be logged. Changes to the DOM are

[2] **LogUI** has been tested with *Chrome*, *Edge*, *Firefox*, *Opera*, and *Safari*.

[3] Note that the *Application Server* and ***LogUI** Server* are two entirely different processes, and can be run on separate computers (with *CORS* support enabled).

also incorporated, with **LogUI** watching for new elements matching a given CSS selector, and applying the necessary events.[4] We also include so-called *grouping,* where one or more events can be chained together to act as a single, managable event (e.g., `mouseover` and `mouseout` events would constitute a grouped `hover` event). This means that additional logic can be added to avoid logging *noisy* events, such as when scrolling. This is a novel and non-trivial feature, and while it affords additional complexity, it results in cleaner logs.

Metadata: One or more pieces of *metadata* may be required to be packaged with a logged event.[5] **LogUI** provides numerous *metadata sourcers*, allowing for the extraction of data from different locations (e.g., the attribute of the element, or `localstorage`). We also include sourcers for frameworks like React, allowing one to extract a `prop` or `state` value from the associated *component*.

LogUI Server: The server authenticates a **LogUI** client, and receives the packaged event data ❻. It is then placed in backing storage ❼ (with session IDs, allowing for filtering/merging). Captured data for search/browsing sessions can be then downloaded and used for data analysis ❽. The server is implemented within a containerised environment to aid portability.

Integration with Web Applications: LogUI can be seamlessly integrated within existing web applications. As it examines the DOM only, it is *framework agnostic.* The client is self-contained, meaning it does not interfere with other libraries. Logging is as easy as *1-2-3*: *(1)* include the client library within the web application; *(2)* specify what elements and events to log; and *(3)* start a server instance to receive the logged events. The framework provides support for web applications on a single webpage, or over multiple webpages. Interactions over multiple pages can therefore be counted as a single session. A simple API is also provided to start and stop the library, or reset the session as required.

Availability: Code is open source and available from GitHub. The client is accessible at `https://github.com/logui-framework/client/`, with server code at `https://github.com/logui-framework/server/`. Documentation for both components are also available in the respective repository.

4 Summary

We have described our new logging framework, **LogUI**. The complexity that the framework handles (along with the relative simplicity of using it) will provide a powerful new tool for researchers to deploy when logging user interactions as part of IIR experiments. We aim to continue developing the framework to support more advanced features[6], and will promote its use in a wide variety of experimental apparatus, leading to increased productivity for researchers.

Acknowledgements This research has been supported by *NWO* projects *SearchX* (`639.022.722`) and *Aspasia* (`015.013.027`).

[4] **LogUI** therefore supports contemporary client-side web application frameworks.

[5] Metadata examples could include the `docid` for a document presented on a SERP, or, more generally, the condition a participant is assigned to in an *A/B test*.

[6] Features could include a *UXJs*-style [19] analysis interface, or screen capturing.

References

1. Alexander, J., Cockburn, A., Lobb, R.: AppMonitor: A tool for recording user actions in unmodified Windows applications. Behavior Research Methods **40**(2), 413–421 (2008)
2. Apaolaza, A., Harper, S., Jay, C.: Longitudinal Analysis of Low-Level Web Interaction through Micro Behaviours. In: Proc. 26th ACM HT. p. 337–340 (2015)
3. Atterer, R., Wnuk, M., Schmidt, A.: Knowing the User's Every Move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In: Proc. 15th WWW. p. 203–212 (2006)
4. Atterer, R.: Logging usage of AJAX applications with the "UsaProxy" HTTP proxy. In: Workshop on Logging Traces of Web Activity, Proc. 15th WWW (2006)
5. Bierig, R., Gwizdka, J., Cole, M.J.: A user-centered experiment and logging framework for interactive information retrieval. In: Workshop on Understanding the User, Proc. 32nd ACM SIGIR. pp. 8–11 (2009)
6. Bigham, J., Cavender, A.: Evaluating existing audio captchas and an interface optimized for non-visual use. In: Proc. 27th ACM CHI. p. 1829–1838 (2009)
7. Bilal, D., Gwizdka, J.: Children's eye-fixations on google search results. Proc. ASIS&T **53**(1), 1–6 (2016)
8. Dekel, U.: A Framework for Studying the Use of Wikis in Knowledge Work Using Client-Side Access Data. In: Proc. 3rd WikiSym. p. 25–30 (2007)
9. Doolan, M., Azzopardi, L., Glassey, R.: ALF: A Client Side Logger and Server for Capturing User Interactions in Web Applications. In: Proc. 35th ACM SIGIR. p. 1003 (2012)
10. Edmonds, A., White, R.W., Morris, D., Drucker, S.M.: Instrumenting the Dynamic Web. J. Web Eng. **6**(3), 244–260 (2007)
11. Hall, M., Toms, E.: Building a common framework for IIR evaluation. In: Proc. 4th CLEF. pp. 17–28 (2013)
12. Hienert, D., van Hoek, W., Weber, A., Kern, D.: Whose – a tool for whole-session analysis in iir. In: Proc. 37th ECIR. pp. 172–183 (2015)
13. Jansen, B.J., Ramadoss, R., Zhang, M., Zang, N.: Wrapper: An Application for Evaluating Exploratory Searching Outside of the Lab. In: Workshop on Evaluating Exploratory Search Systems, In Proc. 29th ACM SIGIR (2006)
14. Kukreja, U., Stevenson, W.E., Ritter, F.E.: RUI: Recording user input from interfaces under Windows and Mac OS X. Behavior Research Methods **38**(4), 656–659 (2006)
15. Lassila, M., Pääkkönen, T., Arvola, P., Kekäläinen, J., Junkkari, M.: Unobtrusive Mobile Browsing Behaviour Tracking Tool. In: Proc. 4th IIiX. p. 278–281 (2012)
16. Philips, B.H., Dumas, D.J.S.: Usability Testing: Identifying Functional Requirements for Data Logging Software. Proc. Human Factors Society Annual Meeting **34**(4), 295–299 (1990)
17. Rossi, G., Urbieta, M., Distante, D., Rivero, J.M., Firmenich, S.: 25 Years of Model-Driven Web Engineering. What we achieved, What is missing. CLEI Elec. J. **19**(3), 5–57 (2016)
18. Singer, G., Norbisrath, U., Vainikko, E., Kikkas, H., Lewandowski, D.: SearchLogger: Analyzing Exploratory Search Tasks. In: Proc. 26th ACM SAC. p. 751–756 (2011)
19. Solís-Martínez, J., Espada, J.P., González Crespo, R., Pelayo G-Bustelo, B.C., Cueva Lovelle, J.M.: UXJs: Tracking and Analyzing Web Usage Information With a Javascript Oriented Approach. IEEE Access **8**, 43725–43735 (2020)

20. Wei, X., Zhang, Y., Gwizdka, J.: Yasfiire: Yet another system for iir evaluation. In: Proc. 5th IIiX. p. 316–319 (2014)
21. Westerman, S.J., Hambly, S., Alder, C., Wyatt-Millington, C.W., Shryane, N.M., Crawshaw, C.M., Hockey, G.R.J.: Investigating the human-computer interface using the Datalogger. Behavior Research Methods, Instruments, & Computers **28**(4), 603–606 (1996)