Swimming in a Sea of JavaScript or: How I learned to Stop Worrying and Love High-Fidelity Replay

John A. Berlin  
*Old Dominion University*

Michael L. Nelson  
*Old Dominion University*

Michele C. Weigle  
*Old Dominion University*

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_fac_pubs

Part of the Archival Science Commons, and the Databases and Information Systems Commons

**Original Publication Citation**

This Conference Paper is brought to you for free and open access by the Computer Science at ODU Digital Commons. It has been accepted for inclusion in Computer Science Faculty Publications by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.
Swimming In A Sea Of JavaScript Or: How I Learned To Stop Worrying And Love High-Fidelity Replay

John A. Berlin, Michael L. Nelson, and Michele C. Weigle
Old Dominion University
Department of Computer Science
Norfolk, Virginia, USA
{jberlin,mln,mweigle}@cs.odu.edu

1 INTRODUCTION
Preserving and replaying modern web pages in high-fidelity has become an increasingly difficult task due to the increased usage of JavaScript. Reliance on server-side rewriting alone results in live-leakage and or the inability to replay a page due to the preserved JavaScript performing an action not permissible from the archive. The current state-of-the-art high-fidelity archival preservation and replay solutions rely on handcrafted client-side URL rewriting libraries specifically tailored for the archive, namely Webrecorder’s and Pywb’s wombat.js [12]. Web archives not utilizing client-side rewriting rely on server-side rewriting that misses URLs used in a manner not accounted for by the archive or involve client-side execution of JavaScript by the browser.

We have developed a general framework for the automatic generation of client-side rewriting libraries using the Web Interface Design Language (Web IDL) [10] that is archive and replay system independent. We provide a high-level overview of the auto-generation framework and evaluation performed that tested the auto-generated client-side rewriter’s ability to augment the existing server-side rewriting system of the Internet Archive’s Wayback Machine [3]. We show that client-side rewriting would both increase the replay fidelity of mementos and enable mementos that were previously unplayable from the Internet Archive’s Wayback Machine to be repayable again.

2 BACKGROUND AND RELATED WORK
Brunelle and Kelly [6] conducted a study of 1,861 URls which had mementos in the Internet Archive between 2005 to 2012 in order to identify the impact of JavaScript on the archivability of web pages. They found that JavaScript was responsible for 52.7% of all missing resources and that by 2012 JavaScript was responsible for 33.2% more missing resources than in 2005. Brunelle and Kelly [4, 5] also conducted a study that looked at the proportion of missing resources for mementos in the Internet Archive between June 1 and June 30. We then pre-crawled the URI-Ms using the Google Chrome browser controlled via the DevTools Protocol removing URI-Ms from the frontier that redirected more than 10 times or took longer than 20 seconds for the browser to navigate to the page, resulting in 577 resolved URI-Ms. We then crawled each composite memento using the controlled browser four times, twice without client-side rewriting and twice with client-side rewriting, recording the number of requests made by the composite memento and the number of requests blocked by the Wayback Machine’s content-security policy (CSP).

The crawler visited each composite memento for a maximum of 90 seconds or until network idle was determined. The determination for network-idle was calculated by keeping track of the request and response pairs for a page, and when there was only one in-flight request (no response) for 3 seconds the crawler moved to the next URI-M. Once all crawls had completed, we selected the data generated from one of the two crawls, with or without client-side rewriting, that recorded the most number of requests. We found

3 AUTO-GENERATION
Web IDL was created by the W3C to “describe interfaces intended to be implemented in web browser”, “allow the behavior of common script objects in the web platform to be specified more readily”, and “provide how interfaces described with Web IDL correspond to constructs within ECMA Script execution environments” [10]. Our framework uses the Web IDL definitions for the JavaScript APIs of the browser included in or link to by the HTML and CSS specification in combination with the description of how Web IDL maps to the JavaScript environment, provided by the Web IDL specification, in order to auto-generate a client-side rewriting library. This allows the generated rewriter to perform the same URL rewriting done server-side in addition to applying targeted overrides to the JavaScript APIs of the browser in order to intercept and rewrite un-rewritten URLs client-side.

We have released the generated client-side rewriter as Firefox browser extensions so that others may use it to improve the replay of mementos from the Internet Archive. Note that although the generated client-side rewriter is similar to the de-facto implementation for client-side rewriting libraries, wombat.js, it is replay system agnostic.

4 EVALUATION
We retrieved the TimeMaps for the web pages listed in the June 2017 Alexa top 1,000,000 most visited websites and selected the first 700 pages, excluding Google and Facebook pages, that had a memento in the Internet Archive between June 1 and June 30. We then pre-crawled the URI-Ms using the Google Chrome browser controlled via the DevTools Protocol removing URI-Ms from the frontier that redirected more than 10 times or took longer than 20 seconds for the browser to navigate to the page, resulting in 577 resolved URI-Ms. We then crawled each composite memento using the controlled browser four times, twice without client-side rewriting and twice with client-side rewriting, recording the number of requests made by the composite memento and the number of requests blocked by the Wayback Machine’s content-security policy (CSP).

The crawler visited each composite memento for a maximum of 90 seconds or until network idle was determined. The determination for network-idle was calculated by keeping track of the request and response pairs for a page, and when there was only one in-flight request (no response) for 3 seconds the crawler moved to the next URI-M. Once all crawls had completed, we selected the data generated from one of the two crawls, with or without client-side rewriting, that recorded the most number of requests. We found

---

2https://chrome.google.com/webstore/detail/wayback%20/kcpoejobnjkdilndjkgcmmnakccjkhka
3https://chromedevtools.github.io/devtools-protocol/
Cumulative Number Of Blocked Requests
Cumulative Number Of Requests Made

<table>
<thead>
<tr>
<th>Requests Made</th>
<th>Requests Blocked</th>
</tr>
</thead>
<tbody>
<tr>
<td>120,000</td>
<td>32,212</td>
</tr>
<tr>
<td>140,000</td>
<td>45,051</td>
</tr>
<tr>
<td>160,000</td>
<td>5,972</td>
</tr>
<tr>
<td>182,122</td>
<td>87.5% Decrease</td>
</tr>
<tr>
<td>1,500</td>
<td>+45,051 Requests</td>
</tr>
<tr>
<td>2,500</td>
<td>3,500</td>
</tr>
<tr>
<td>3,500</td>
<td>4,500</td>
</tr>
<tr>
<td>5,500</td>
<td>6,500</td>
</tr>
<tr>
<td>6,500</td>
<td>847</td>
</tr>
</tbody>
</table>

Wayback Machine with and without client-side rewriting which used three different ways of lazy loading its images (Figure 3).

but ensuring both high fidelity replay and the secure replay of archived JavaScript necessarily requires an archive to employ client-side rewriting. Client-side rewriting is a general solution to the archived JavaScript necessarily requires an archive to employ client-side rewriting.

5 CONCLUSIONS

One might believe that the usage of client-side rewriting is only limited to the most dynamic of web pages or web applications, but ensuring both high fidelity replay and the secure replay of archived JavaScript necessarily requires an archive to employ client-side rewriting. Client-side rewriting is a general solution to the increasingly difficult problems of mitigating the impact of JavaScript on archivability, increasing users’ perception of archival quality and ensuring the secure replay of JavaScript.

6 ACKNOWLEDGMENTS

This work sponsored in part by NEH HK-50181 and NSF IIS 1526700.

REFERENCES