Client-side Reconstruction of Composite Mementos Using ServiceWorker

Sawood Alam, Mat Kelly, Michele C. Weigle, and Michael L. Nelson
Department of Computer Science, Old Dominion University
Norfolk, Virginia, USA - 23529
{salam,mkelly,mweigle,mln}@cs.odu.edu

ABSTRACT

We use the ServiceWorker (SW) API to intercept HTTP requests for embedded resources and reconstruct Composite Mementos without the need for conventional URL rewriting typically performed by web archives. URL rewriting is a problem for archival replay systems, especially for URLs constructed by JavaScript, that frequently results in incorrect URI references. By intercepting requests on the client using SW, we are able to strategically reroute instead of rewrite. Our implementation moves rewriting to clients, saving servers' computing resources and allowing servers to return responses more quickly. In our experiments, retrieving the original instead of rewritten pages from the archive resulted in a one-third reduction in time overhead and a one-fifth reduction in data overhead. Our system, `reconstructive.js`, prevents the live web from leaking into Composite Mementos while being easy to distribute and maintain.

CCS CONCEPTS

•Information systems →Digital libraries and archives; World Wide Web;

KEYWORDS

ServiceWorker, Memento, Composite Memento, Web Archive, Archival Replay

ACM Reference format:
Sawood Alam, Mat Kelly, Michele C. Weigle, and Michael L. Nelson. 2016. Client-side Reconstruction of Composite Mementos Using ServiceWorker. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

ServiceWorker (SW) is a new client-side web API [11] that can be used to intercept all the network requests, originating from web pages in its scope, for embedded resources. A Composite Memento [2] is an archived HTML page along with all the embedded resources (page requisites) that are necessary to render the page correctly. Web archival replay systems rewrite embedded resource references to point to their archival versions e.g., a reference to `external.example.net/logo.png` is changed to `archive.example.org/<datetime>/external.example.net/logo.png`.

We use SW API to reconstruct Composite Mementos from the originally captured data without any such URL rewriting. By intercepting requests on the client-side, we are essentially rerouting instead of rewriting. Rerouting is an effective mechanism to block live web leakage, or "*zombies*". URIs constructed by JavaScript (JS) are often difficult to discover by static analysis for rewriting. For example, in Figure 1 the page was archived on September 3, 2008, but when observed on September 28, 2012, it pulled in a banner ad from the live web, which seems to provide a prophetic look at the 2012 presidential candidates [5]. Client-side rerouting also saves bandwidth by allowing necessary rewriting of the content (such as archival banner inclusion) on the client side. Hence, there is no need to send extra data with each HTML response. Client-side solutions, such as Memento for Chrome[1], involve installing a browser add-on, which limits the adoption by users. Additionally, each add-on/extension adds the maintenance burden to developers while being available for users of only specific platforms. Our exploratory technique works well when SW is supported. However, a server-side fallback is necessary for production usage to avoid the risk of zombies and broken references when SW is not supported.

In our experiments, retrieving the original instead of rewritten pages from the the Internet Archive (IA) resulted in a one-third reduction in time overhead and a one-fifth reduction in data overhead. Our system prevents zombies from Composite Mementos while being easy to distribute and maintain. It is a lightweight and portable system that can be used with any Memento server such as a web archive or a Memento aggregator.

2 BACKGROUND

A memento is a timestamped representation of a web resource identified by a URI-M [13]. A web page is often comprised of a base HTML page and various embedded resources such as images, stylesheets, JS, fonts, and other media (each with an independent URI) that are necessary to render the page correctly. A Composite Memento is a memento of a base HTML page and mementos of all corresponding embedded resources around the same time as the base page to render the page the way it looked in the past [2].

References to these embedded resources can be absolute URIs, absolute paths, or relative paths in attributes like `href` or `src`. A relative path is relative to the base URL, either explicitly specified using `<base>` element or implicitly derived from the URL of the current page. In order to dereference a resource over HTTP the client must resolve its reference to an absolute URL. To do so, a client may use various pieces of information to accomplish this such as domain's root URL, origin's base URL, and the path of the resource. When the domain name or the root path of the site changes, some of these references may resolve incorrectly.

[1] http://bit.ly/memento-for-chrome

Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

Figure 1: Live Ad Zombie Leaks into an Archived Page

When an archived page is replayed, both the domain name and the root path are changed. To correctly route all the resource references, web archival replay systems (such as OpenWayback[2], PyWB[3], Memento Reconstruct[4]) perform static analysis of HTML pages, CSS, and JS files to rewrite `href` and `src` attributes in a way that points to their archival versions. Alternatively, a proxy can be configured or a browser add-on can be installed to reroute requests appropriately.

ServiceWorker is a new client-side web API that acts as a proxy between a web application and the web server. It can intercept all the network requests originating from web pages in its scope. A SW is installed in the form of a JS file loaded from a path on a host like any other resource. Any resource under that path of the host is in the scope of the SW. Subsequent requests originating from any web page in scope are also intercepted by the SW, even if the resource resides on an external domain. This API is often used to make web applications accessible offline using client-side caching, background data synchronization, and push notifications. We use this API to reconstruct Composite Mementos from the originally captured data without any URL-rewriting. Our technique can be utilized by other URI-based services such as web annotations. SW is still in the working draft phase, but already implemented by many major browsers.

3 RELATED WORK

Ainsworth introduced a framework for assessing temporal coherence of a Composite Memento [1]. A temporal violation may occur due to poor archiving or poor playback. The latter is the focus of our work.

Kelly created the state-of-the-art Acid test suite for archival systems [10], both capture and replay. While Kelly's Archival Acid test was focused on evaluating the capture quality and pixel-perfect rendering, it does not cover all cases of how a network request can be initiated and where the responses come from. Our focus is mainly on the network activity to make sure that each response is coming from the appropriate archive and there are no zombies. We evaluate rerouting of all the requests originated explicitly, implicitly, or after any interaction with the page.

Measuring the quality of the crawling or capture is beyond the scope of this work. Brunelle has done exhaustive research about the impact of missing resources [6] and capturing the deferred representation [7] (rendered state of a page after some interactivity or JS execution). Our focus is to load those resources properly if present in the archive.

Jones proposed using the `Prefer` HTTP header with the existing Memento protocol [13] to request the unaltered (raw) archived web content [8]. Current practice is to use a URL based technique (appending `id_` to the `datetime` digits), which has been a little-known feature of the Wayback Machine.

Sanderson discussed the challenges and solutions discovered for implementing the Memento protocol in a variety of environments, including MementoFox (a Firefox add-on), a plugin for Internet Explorer, and an Android-based browser [12].

4 METHODOLOGY

Figure 2 illustrates the workflow of our reconstruction method. Suppose a user visits `archive.example.org`, which installs a SW `reconstructive.js` in the user's web browser under the root of the domain. This SW is detached from the page and persists in the browser independently to watch all network activities originated under its scope. The user then loads a copy of `www.example.com` from `archive.example.org` that was archived on January 26, 2017. This memento has an embedded image that points to an external domain `external.example.net`. The browser would have sent the request to the external domain, but due to the presence of the SW, it will be intercepted. The SW `reconstructive.js` gets access to the request object that contains a `referer` header (the URL of the originating page that is shown in the address bar of the browser). Based on the available information (e.g., the `datetime` of the originating page) the SW can create a new request, load the corresponding resource from the archive, make any modification in the response (if needed, such as adding banners in HTML pages), and return the response to the page for rendering. To maintain the same-origin boundary for external resources that might load more resources, such as iframe source or CSS, we first issue a `302` redirect to the corresponding URI-M locally from the SW. All the logic of exclusions and rerouting is present in the `reconstructive.js` file, which can be updated on the server when needed. When the user visits the home page of the archive, the corresponding SW will be updated automatically. Every request originating from our SW has a custom header, `X-ServiceWorker: reconstructive.js:v1`, so the server can decide whether it needs to return a server-side

[2] https://github.com/iipc/openwayback
[3] https://github.com/ikreymer/pywb
[4] http://timetravel.mementoweb.org/

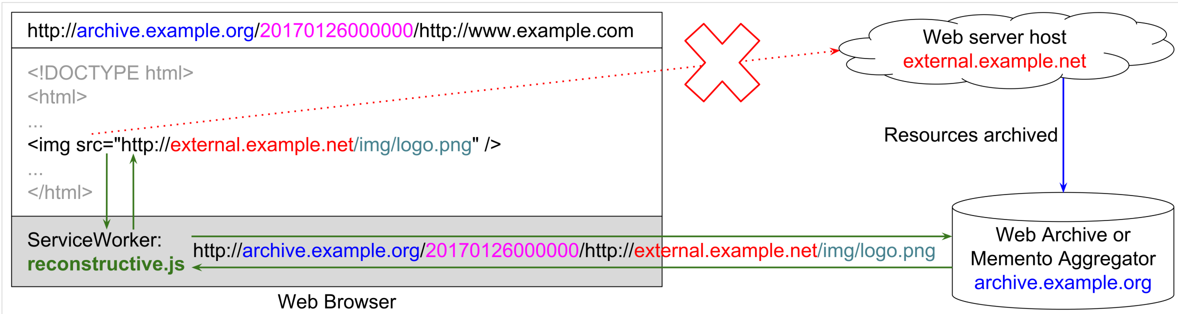

Figure 2: ServiceWorker reconstructive.js Intercepts a Zombie Resource and Reroutes to its Archived Copy

rewritten response as a fallback for unsupported clients or attempt to install/update the SW.

Our goal in this exploratory work was to effectively reconstruct a Composite Memento with zero rewriting. However, using client-side rewriting we can add other useful features such as an archival banner with metadata and toolbar, pointing hyperlinks to the archived version, or JS handlers adding custom behavior when hyperlinks are followed.

5 IMPLEMENTATION

The reference implementation of the `reconstructive.js` is being used by our InterPlanetary Wayback archival replay system [3, 9]. Additionally, we use `reconstructive.js` in our Memento aggregator, MemGator [4], to facilitate cross-archive Composite Memento reconstruction. To deal with the Cross-Origin Resource Sharing (CORS) [14] restrictions we added Memento Proxy feature in MemGator. We open-sourced `reconstructive.js`[5], IPWB[6], and MemGator[7].

6 EVALUATION

To quantify the benefits of client-side rerouting we collected 500 domains[8] and sequentially fetched their mementos closest to January 26, 2017 (using `20170126000000` as the 14-digit `datetime` string) in both original and rewritten forms from the IA. We repeated this three times and found the time variance less than 30 seconds. The rewritten responses took an average of 26.97 minutes while the original versions took only 19.88 minutes. Hence, there is an overhead of 35.66% in the mean response time with the server-side rewriting and banner inclusion.

Among the 500 requests there were 33 `403 Forbidden`, five `404 Not Found`, and one `504 Gateway Timeout`. There were 19 cases where the original archived memento was a `302 Found` with a `Location` header containing an absolute path. If not rewritten by the server, the path would resolve to an irrelevant or non-existing resource. There were 79 `301 Moved Permanently` responses for which IA shows a splash page for a few seconds then redirects. We eliminated all of these 137 problematic responses and evaluated the data sizes for the two cases. Original responses were 44.98 MB total while the rewritten responses were 50.74 MB accumulated, hence, a 12.80% overhead. If we also include the 3.09 MB of 79 `301 Moved Permanently` splash pages then the data overhead becomes 19.68%. We also found that IA can lead to a different chain of redirects for original and rewritten requests when negotiating with the same starting `datetime` value, resulting in different terminal mementos for the two cases.

While the time and data overhead evaluations are not directly tied to the SW, the benefit is worth knowing as the server-side URL rewriting and banner inclusion is not necessary when SW-based rerouting is utilized. Reported data overhead is calculated over sampled home pages that are all HTML pages, which might differ significantly for a different sample set. Rewriting is only performed on text files (i.e., HTML, CSS, or JS) while binary files such as images are served unchanged. Also, reported time values may differ significantly depending on the network latency when the experiment is carried out. However, we always expect some savings in both time and data overheads when requesting the original, instead of the rewritten content from the archive.

To evaluate the archival replay reconstruction quality we created the Archival Capture Replay Test Suite (ACRTS)[9] with different scenarios of how a web page might initiate a network request. We archived ACRTS and saved the resulting Web ARChive (WARC) file. We then changed the live ACRTS site in a way that all the resource references remained the same, but their content was changed. Using various replay systems we loaded the archived ACRTS from the stored WARC file. Depending on how effective the replay system is, it might load resources from the archive (☑), leak from the live site (☒), or not load at all (☐). The latter might happen either because the requested resource was not present in the archive or the replay system resolved the location incorrectly. Correct routing from the archive is desired in an effective archival replay system.

Table 1 shows how well each of the listed archival replay systems reconstructs a composite memento when resource requests are originated from different conditions. OpenWayback relies only on server-side rewriting, hence, fails when URLs are constructed using string concatenation and variables in JS. PyWB uses both server-and client-side rewriting to mitigate live leakage, which results in good reconstruction. Memento Reconstruct uses PyWB as the replay engine, but reconstructs the page from aggregated resources in an iframe. The scrolling issue is caused by the iframe configuration, while others are due to the CORS restrictions when archives return rewritten responses. Memento for Chrome is a pure client-side system that redirects the base page to corresponding URI-M without intercepting embedded resource requests, hence the quality depends on the target archive. Our SW based system, `reconstructive.js`, requests for the original content from archives and makes all the rerouting decisions on the client. This saves bandwidth and prevents zombies.

[5]https://github.com/oduwsdl/reconstructive
[6]https://github.com/oduwsdl/ipwb
[7]https://github.com/oduwsdl/memgator
[8]Moz's list of the top 500 domains on the web ranked by the number of linking root domains on January 26, 2017. https://moz.com/top500.

[9]https://ibnesayeed.github.io/acrts/

Table 1: URL Rewriting/Rerouting Results in Different Archival Replay Systems (A: OpenWayback, B: PyWB, C: Memento Reconstruct, D: Memento for Chrome, and E: Reconstructive)

Resource Loading Scenarios	A	B	C	D	E
Relative path	☑	☑	☑	☑	☑
Absolute rooted path	☑	☑	☑	☑	☑
Absolute local URL	☑	☑	☑	☑	☑
Absolute external URL	☑	☑	☑	☑	☑
External resource from an external iframe	☑	☑	☑	☑	☑
Loaded by an inline CSS	☑	☑	☑	☑	☑
Loaded by a CSS file	☑	☑	☑	☑	☑
Loaded by CSS @font-face	☑	☑	☐	☑	☑
Loaded by image srcset	☑	☑	☑	☑	☑
Added by an inline JS on page load	☒	☑	☑	☒	☑
Added by an inline JS on page scroll	☒	☑	☐	☒	☑
Added by an inline JS on click	☒	☑	☑	☒	☑
Added by a JS file	☒	☑	☑	☒	☑
Added by an Ajax request	☒	☑	☐	☒	☑

Apart from the evaluated quantitative advantages there are some other advantages as well. Our method enables the ability to verify the fixity of the archived content on the client side as the server returns the original archived content without any modification. It has many of the same features that are provided by browser add-ons such as MementoFox (deprecated). However, maintaining separate add-ons for each popular browser and keeping them up-to-date is a difficult task. Encouraging users to install add-ons is another barrier to adoption. In contrast, a SW is easier to maintain, update, and distribute as it is a JS file hosted on the web server and updates in clients' browsers automatically. Additionally, the same code works in many different browsers.

The support for SW was introduced in Chrome 40 and Opera 27 in January 2015. Firefox 44 added support in January 2016. Firefox does not support SW in private browsing mode. For security reasons SW only runs over HTTPS. There is 61.55% support globally as of January 31, 2017[10].

7 FUTURE WORK

We would use the `Prefer` header for content negotiation [8] when it is supported by web archives. We would like to add a customizable archival banner as part of the client-side rewriting using HTML5 Web Components[11] to avoid any style conflicts with the page. Ability to verify the fixity of the archived content would be another valuable addition.

SW is an emerging technology, which has many opportunities to create useful services and tools. We would like to investigate the possibility of a variation to our SW that can be used by webmasters in personal sites, wikis, blogs, or other content management systems to use web archives as a fallback cache when embedded resources are gone missing from the live web. Missing resources are a big issue in social platforms like forums where users post content, including external media, that make the thread less useful when gone. Additionally, we can add the ability to request web archives to capture resources that are loaded by any user of a page from a domain that hosts our SW. These use cases can encourage utilization of web archives by webmasters to cause fewer `404`s for users.

[10] http://caniuse.com/#feat=serviceworkers
[11] https://www.webcomponents.org/specs

8 CONCLUSIONS

We utilized the ServiceWorker web API to explore intercepting requests and reconstruct Composite Mementos without the need for conventional URL rewriting of web archives. We developed a prototype implementation and used it as the replay system for an InterPlanetary File System (IPFS) based standalone web archive and for a Memento aggregator for cross-archive reconstruction. We created a test suite to evaluate different archival replay systems to measure the rerouting quality where our implementation passes all the cases. In our experiments, retrieving the original instead of rewritten pages from the IA resulted in a one-third reduction in time overhead and a one-fifth reduction in data overhead. Our system prevents Composite Mementos from zombies while being easy to distribute and maintain. It is a lightweight and portable system suitable for any Memento server.

9 ACKNOWLEDGEMENTS

This work is supported in part by NSF grant III 1526700. We thank Scott Ainsworth for his feedback.

REFERENCES

[1] Scott G. Ainsworth, Michael L. Nelson, and Herbert Van de Sompel. 2014. A Framework for Evaluation of Composite Memento Temporal Coherence. (2014). http://arxiv.org/abs/1402.0928

[2] Scott G. Ainsworth, Michael L. Nelson, and Herbert Van de Sompel. 2015. Only One Out of Five Archived Web Pages Existed As Presented. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media (HT '15)*. 257–266. DOI:http://dx.doi.org/10.1145/2700171.2791044

[3] Sawood Alam, Mat Kelly, and Michael L. Nelson. 2016. InterPlanetary Wayback: The Permanent Web Archive. In *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '16)*. 273–274. DOI:http://dx.doi.org/10.1145/2910896.2925467

[4] Sawood Alam and Michael L. Nelson. 2016. MemGator - A Portable Concurrent Memento Aggregator. In *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '16)*. 243–244. DOI:http://dx.doi.org/10.1145/2910896.2925452

[5] Justin F. Brunelle. 2012. Zombies in the Archives. http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html

[6] Justin F. Brunelle, Mat Kelly, Hany SalahEldeen, Michele C. Weigle, and Michael L. Nelson. 2015. Not All Mementos Are Created Equal: Measuring The Impact Of Missing Mementos. *Int. J. on Digital Libraries* 16, 3-4 (2015), 283–301. http://dx.doi.org/10.1007/s00799-015-0150-6

[7] Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. 2015. Archiving Deferred Representations Using a Two-Tiered Crawling Approach. (2015). http://arxiv.org/abs/1508.02315

[8] Shawn M. Jones, Lyudmila Balakireva, Harihar Shankar, Michael L. Nelson, and Herbert Van de Sompel. 2016. Mementos In the Raw, Take Two. http://ws-dl.blogspot.com/2016/08/2016-08-15-mementos-in-raw-take-two.html

[9] Mat Kelly, Sawood Alam, Michael L. Nelson, and Michele C. Weigle. 2016. InterPlanetary Wayback: Peer-To-Peer Permanence of Web Archives. In *Proceedings of the 20th International Conference on Theory and Practice of Digital Libraries (TPDL '16)*. 411–416. DOI:http://dx.doi.org/10.1007/978-3-319-43997-6_35

[10] Mat Kelly, Michael L. Nelson, and Michele C. Weigle. 2014. The Archival Acid Test: Evaluating Archive Performance on Advanced HTML and JavaScript. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '14)*. 25–28. DOI:http://dx.doi.org/10.1109/JCDL.2014.6970146

[11] Alex Russell, Jungkee Song, and Jake Archibald. 2015. Service Workers. https://www.w3.org/TR/service-workers/

[12] Robert Sanderson, Harihar Shankar, Scott Ainsworth, Frank McCown, and Sam Adams. 2011. Implementing Time Travel for the Web. *Code{4}Lib Journal* 13 (2011). http://journal.code4lib.org/articles/4979

[13] Herbert Van de Sompel, Michael L. Nelson, and Robert Sanderson. 2013. HTTP Framework for Time-Based Access to Resource States – Memento. RFC 7089.

[14] Anne van Kesteren. 2014. Cross-Origin Resource Sharing. https://www.w3.org/TR/cors/