Int J Digit Libr (2016) 17:95–117
DOI 10.1007/s00799-015-0140-8

The impact of JavaScript on archivability

Justin F. Brunelle · Mat Kelly · Michele C. Weigle · Michael L. Nelson

Received: 7 November 2013 / Revised: 12 January 2015 / Accepted: 14 January 2015 / Published online: 25 January 2015

Abstract As web technologies evolve, web archivists work to adapt so that digital history is preserved. Recent advances in web technologies have introduced client-side executed scripts (Ajax) that, for example, load data without a change in top level Universal Resource Identifier (URI) or require user interaction (e.g., content loading via Ajax when the page has scrolled). These advances have made automating methods for capturing web pages more difficult. In an effort to understand why mementos (archived versions of live resources) in today's archives vary in completeness and sometimes pull content from the live web, we present a study of web resources and archival tools. We used a collection of URIs shared over Twitter and a collection of URIs curated by Archive-It in our investigation. We created local archived versions of the URIs from the Twitter and Archive-It sets using WebCite, wget, and the Heritrix crawler. We found that only 4.2 % of the Twitter collection is perfectly archived by all of these tools, while 34.2 % of the Archive-It collection is perfectly archived. After studying the quality of these mementos, we identified the practice of loading resources via JavaScript (Ajax) as the source of archival difficulty. Further, we show that resources are increasing their use of JavaScript to load embedded resources. By 2012, over half (54.5 %) of pages use JavaScript to load embedded resources. The number of embedded resources loaded via JavaScript has increased by 12.0 % from 2005 to 2012. We also show that JavaScript is responsible for 33.2 % more missing resources in 2012 than in 2005. This shows that JavaScript is responsible for an increasing proportion of the embedded resources unsuccessfully loaded by mementos. JavaScript is also responsible for 52.7 % of all missing embedded resources in our study.

Keywords Web architecture · Web archiving · Digital preservation

J. F. Brunelle (✉) · M. Kelly · M. C. Weigle · M. L. Nelson
Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
e-mail: jbrunelle@cs.odu.edu

M. Kelly
e-mail: mkelly@cs.odu.edu

M. C. Weigle
e-mail: mweigle@cs.odu.edu

M. L. Nelson
e-mail: mln@cs.odu.edu

1 Introduction

How well can we archive the web? This is a question that is becoming increasingly important and more difficult to answer. Additionally, this question has significant impact on web users [40,43] and commercial and government compliance [52,53,66].

The web has gone through a gradient of changes fueled by increasing user demand for interactivity. Early websites were relatively static, while continued adoption of web technologies has made the pages personalized and more interactive. JavaScript, which executes on the client, provides additional features for the web user, enabling or increasing interactivity, client-side state changes, and personalized representations. These additional features offer an enhanced browsing experience for the user.

JavaScript has enabled a wide-scale migration from web *pages* to web *applications*. This migration continued with the introduction of Ajax (first introduced in 2005 [28]), which combined multiple technologies to give web pages the ability to perform asynchronous client–server interactions after the HTML is loaded. The first wide-scale implementation of Ajax was in Google Maps in 2005, but Ajax was officially added as a standard in 2006 [70]. While archival tools per-

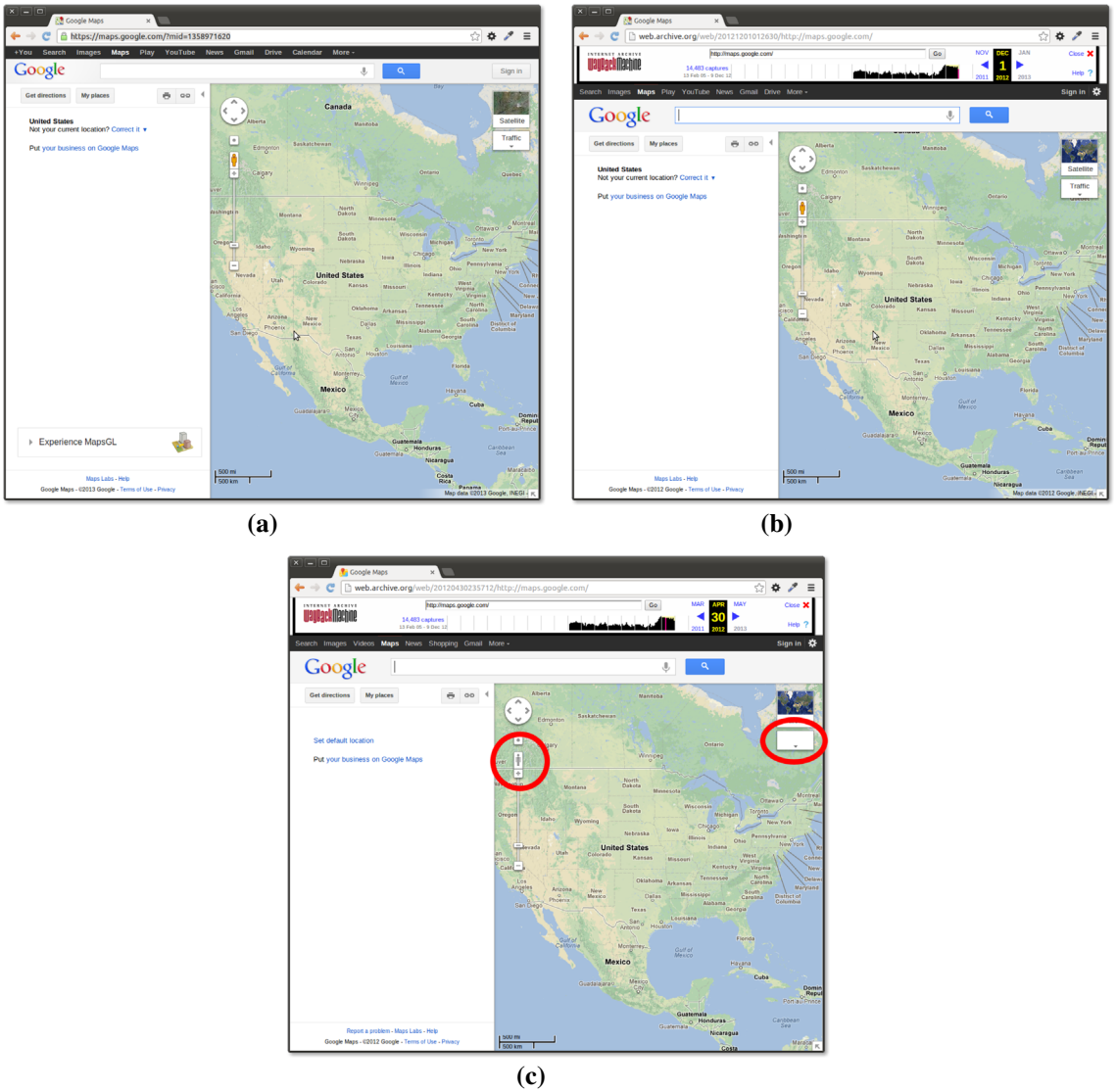


Fig. 1 Google Maps as it exists live and as a memento **a** live version, **b** archived Dec. 1, 2012, **c** archived Apr. 30, 2012

formed satisfactorily when the web consisted almost entirely of static resources, archival tools (e.g., Heritrix [51,65], WebCite [22], and the wget application [29]) cannot capture the more complex, feature-rich pages making up today's web.

The ease of archiving a web page (henceforth the *archivability*) is impacted by the migration from web pages to web applications. Over the years the web has shifted from a web of documents to a web of applications, and these applications require execution by the client to render what the user experiences. Client-side rendering has not historically been part of the archival process. Even if it were, the applications depend on client-side inputs that increase the variability between what is archived and what the user experienced [60].

Because of the advancement and increasing adoption of web technologies, the web is composed of an increasing proportion of personalized resources that are difficult to archive. We discuss how these technologies impact archivability in Sect. 9. Many culturally significant artifacts (such as Google Maps, shown in Fig. 1) are unable to be unarchived. Even if resources are archived, they sometimes become temporally inconsistent because JavaScript may load content from the live web (referred to as the live web "leaking" into the archive and discussed in more detail in Sect. 6.4) instead of only loading temporally appropriate content.

Throughout this paper we use Memento Framework terminology. Memento [68] is a framework that standardizes Web archive access and terminology. Original (or live web) resources are identified by URI-R, and archived versions of URI-Rs are called *mementos* and are identified by URI-M. Memento TimeMaps are machine-readable lists of mementos (at the level of single-archives or aggregation-of-archives) sorted by archival date.

This paper makes two contributions. The first contribution is a study of live resources and mementos to provide insight into what makes a resource archivable. We studied two collections of resources and measured them for archivability. Heritrix, WebCite, and wget are all used to capture a set of resources shared via Twitter and a set of Archive-It curated resources with the intent to determine how well each tool archives the resources. These tools, collections, and resulting mementos are compared, and the results are discussed in depth to show that Twitter resources are less archivable than Archive-It resources, and Heritrix is the best archival tool of the three observed. In culmination, the first contribution provides an analysis of how well we can archive the web.

The second contribution is a study of how archivability has changed over time. We study the mementos of the URI-Rs in our collections for archival quality and cause of missing embedded resources. We show that resources are becoming less archivable over time due to the increasing prevalence of JavaScript. In short, the second contribution provides insight into how well archival tools have performed in the past.

2 Deferred representations

According to the web architecture [10,23,32], servers deliver web resources to requesting clients such as a web browser (controlled by a user, such as Firefox [24]) or a web crawler (e.g., Heritrix or other robot). The content delivered to the client by the server is the *representation* of the web resource (e.g., HTML, PDF). The representation may also include a client-side programming language such as JavaScript which will execute when the representation is rendered by a web browser and the client-side code is run by a local compiler or—in the case of JavaScript [25]—an engine.

Executing a technology like JavaScript on the client can potentially cause the representation to change with or without subsequent requests to a server for additional resources. Ajax is often used to request additional resources to be embedded or used in a representation after the initial page load. This sequence is outlined in Fig. 2.

We define *deferred representations* as representations of resources that are difficult to archive because of their use of JavaScript and other client-side technologies. *Deferred* refers to the final representation that is not fully realized and constructed until *after* the client-side representation is rendered; the representation's final state is *deferred* until after client-side code and events have finished executing.

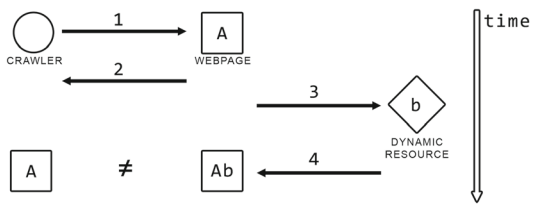


Fig. 2 Ajax interactions modify the DOM after the original page load

As we have mentioned, web browsers use an engine to run client-side JavaScript. Web crawlers often do not possess such an engine and, therefore, do not have the ability to execute JavaScript during the archival process. In other words, archival crawlers are not equipped with the features necessary to execute JavaScript. If Ajax requests additional resources to be embedded in the representation, crawlers will be unaware of the extra resources required to properly render the representation and, ultimately, will not archive these embedded resources.

Because most web crawlers do not have the ability to execute embedded JavaScript or other client-side technologies, the resulting mementos may only be partially operational or incomplete. We discuss crawlers' current capabilities further in Sect. 4.

Google Maps is an example of a resource with a deferred representation and where the lack of completeness of mementos is easy to observe. Figure 1a shows the live page, containing multiple interactive UI elements. The map in the middle is draggable, allowing the user to pan. An archived version (from April 30, 2012) of this page in Fig. 1c is missing UI elements and functionality (circled), and the interaction (e.g., panning and zooming) does not function. This is due to resources that would be loaded when the user clicks, but that are not preserved by the crawler. Figure 1b shows an archived (December 1, 2012) version that gives the façade of functionality when, in fact, resources on the live web are being loaded.

Figure 2 outlines the role Ajax plays in loading a resource like Google Maps. When a crawler fetches a web page (1), it waits for the page and all of the embedded resources to load to consider it fully rendered. At this point, the crawler preserves (2) all of the content on the web page (A). After the page has loaded, the page can request further resources (3), even without user interaction. The resource is then returned to the web page to be rendered (4) producing the final intended result (Ab). Because the crawler preserved the page prior to the page being fully loaded, the secondary content is not preserved and the archived version of the web page is not complete.

3 Prior work

Research relating to JavaScript and archivability has discussed the influences of web technologies and multiple client-side states. Likarish [38] developed a means of detecting JavaScript with certain facets (namely malicious code via deobfuscation) using Machine Learning techniques and Her-

itrix. Archive.today [4] is a page-at-a-time crawler that executes client-side code before saving the HTML (and is now Memento compliant [55]). Archive.today renders the representation, including content loaded from JavaScript, using headless browsing, a method that renders a representation and executes client-side events before capturing the final page code and representation.

Several efforts have performed analysis of personalized content and how to capture such content, although not with archival intent. These efforts, however, contribute greatly to the understanding of how personalized resources can change on the client. Livshits et al. at Microsoft Research have performed extensive research in the monitoring and capture of client-side JavaScript [36,37,39,47,49]. Most of this work has targeted debugging and security monitoring. Other works have included the interpretation of client-side state for crawling purposes [7,20,21,45,48]. Bergman, in a seminal work, described the quantity of resources we are unable to index (the "deep web") [9], a subset of which is due to personalized content. Ast extended Bergman's work by proposing approaches using a conventional web crawler (i.e., one not used for preservation) to capture content not linked to by individual URIs [5]. Other efforts have focused on monitoring Web 2.0 resources on the client [8,16,50,56] for user activity recording.

Archive-It has recently added Umbra to Heritrix in their archival process to help archive a pre-selected set of domains that are known to use JavaScript [59]. Heritrix also peeks into JavaScript of crawled resources and attempts to extract embedded URIs to be added to the frontier [31]. Google's crawler has also made attempts to index representations reliant on JavaScript [12].

McCown performed several comprehensive studies involving reconstructing web pages from the archives' offerings [42,43]. He created the Warrick program [41] as a result of his work.

Several survey papers have identified the roles of JavaScript and other Web 2.0 technologies in a social web environment that is becoming increasingly unarchivable [18,26,46,58,71]. The Smithsonian's blog offers tips on designing archivable resources and how to use ArchiveFacebook to archive a subset of one's social web presence [19]. ArchiveFacebook [27] is a Mozilla Firefox add-on that captures the user's Facebook profile and information in its native format with all embedded resources and content but is suitable for personal archiving only; it cannot be used in larger shared archives.

There have been prior attempts to quantify preservation. The PRISM project [35] assessed the practices threatening the preservation of web resources as they relate to accessibility (such as not using modern tools or having out of date software). Virtual Remote Control [44] created a framework for identifying archival targets and managing at-risk resources. Our previous work investigated the impact of missing embedded resources on users' perceived utility of web resources, quantifying the impact of missing embedded resources in mementos [13,14].

A more recent contribution is Archiveready [6], which provides an interpretation and numerical measure of archivability. The Archiveready service analyzes compliance to standards, number of externally hosted resources, and format of the resource HTML and CSS to establish an archivability measure. Our work extends the theories behind the Archiveready measures to include client-side technologies and supports these claims with a comparison of live and archived resources.

Ainsworth et al. investigated the question "How much of the Web is archived?" [2]. They sampled from larger collections and found that archival coverage varies dependent upon the sample source. For example, URIs shortened by Bitly users were most often newer and less frequently archived than those indexed by other services like DMOZ. In a follow-on experiment, SalahEldeen studied the decay of shared data on Twitter [63] and found that up to 11 % of content shared over Twitter disappears after 1 year, and 25 % disappears after 2 years. SalahEldeen also showed that mementos disappear from the archives, and content that was observed as missing can reappear in the future [64]. These studies highlight the ephemeral nature of shared content and demonstrate that shared links are often not archived, while the work described in this paper shows *why* these resources are difficult to archive. Our work builds on each of these two studies by determining what makes things harder or more difficult to archive and how collections can differ in archivability. We also extend our previous work [33] measuring the change in archivability over time by calculating content missing from mementos due to the use of JavaScript.

4 Current state of the art

As mentioned in the "Prior work" (Sect. 3), several prominent archival services exist and attempt to handle JavaScript-dependent representations. In this section, we briefly discuss the approaches that each of these archival services or tools take to archive web resources. For a more complete list of archival services, along with their archival performance, please refer to our prior work that developed an Acid Test to evaluate mementos created by archival tools and services [34].

4.1 Heritrix

The Heritrix Web Crawler [51] is the Internet Archive's primary archival tool [54]. Heritrix begins with a set of seed URI-Rs as its frontier (set of URI-Rs to archive), derefer-

ences each URI-R in the frontier, and extracts embedded URIs from the resource to add to the frontier. Heritrix can be configured to spider into the broader web or focus on a bounded seed list. The crawler captures all of the requisite resources and stores them in Web ARChive (WARC) files which are then ingested and viewed through the Wayback Machine [67].

Heritrix also peeks into embedded JavaScript files and code to extract any URIs that it can recognize in the code.

In the interest of performance, Heritrix does not use a headless browsing technology and only archives the content returned when the URI-R is dereferenced. As we mentioned in Sect. 3, Archive-It uses Heritrix and has added Umbra to its archival process. Umbra specializes in archiving resources with representations dependent upon JavaScript. A human-specified set of URI-Rs or resources within a specified domain is archived by Umbra when the URI appears in the frontier [11].

In short, Heritrix does not handle deferred representations, but instead, attempts to mitigate the impacts of JavaScript by extracting URIs from the embedded JavaScript code and using Umbra to archive a subset of the frontier.

4.2 WebCite

WebCite is a page-at-a-time archival service. Users submit a URI, and WebCite archives the resource and all embedded resources. WebCite does not use headless browsing and makes no attempt to mitigate the impacts of JavaScript on the memento representation.

4.3 wget

The wget command is a command-line tool that dereferences a URI and provides the option to store the resulting content. Users can also set flags to instruct wget to download all embedded resources, create WARC files, and other useful archival activities. However, wget operates at the HTTP-level only, and does not use headless browsing or make other attempts to execute client-side scripts.

4.4 Archive.today

Similar to WebCite, Archive.today is a page-at-a-time archival service in which users submit URIs for archiving. Archive.today uses headless browsing to record what embedded resources need to be captured to provide a high-quality memento, as well as takes a PNG snapshot of the representation to provide a static and non-interactive visualization of the representation. Because of the tools Archive.today uses, it offers the best handling of JavaScript and creates high-quality mementos.

4.5 PhantomJS

Even though it is not an archival tool, we used PhantomJS[1] throughout this study to load content in a headless browser and measure the HTTP headers involved with loading the resource. PhantomJS is a JavaScript API that runs from the command line. It implements headless browsing, which loads the entire page using WebKit.[2] It also offers screen capture, network monitoring, and page automation (the page automation is not used in this study). Because PhantomJS uses headless browsing, it executes client-side code like JavaScript and provides itself as a technology suitable for supplementing archival efforts.

In this article, we discuss the difficulties these archival tools and services have when encountering resources that rely on JavaScript. A utility like PhantomJS can mitigate the impact of JavaScript on memento quality. However, the challenge that JavaScript creates for archival institutions and tools is an open problem for which a solution does not currently exist. Our initial investigations indicate that there is a performance trade-off between Heritrix (which crawls very quickly but may miss embedded resources dependent upon JavaScript) and PhantomJS (which crawls very slowly but can discover embedded resources dependent upon JavaScript).

There will likely be no single solution to the open challenge of archiving JavaScript. Our future work will focus on understanding the performance trade-offs of using a headless browsing tool and using Heritrix. Our preliminary investigation has shown that Heritrix runs 12.15 times faster than PhantomJS.

5 Motivating examples

Resources that rely heavily on JavaScript to embed representations from other URIs—such as images, text in HTML iframes, etc.—are inherently difficult to archive because they are not known until after the resource is rendered on the client. Figures 4, 5, 6, 7 demonstrate this phenomenon. We archived versions of the resources in Figs. 3, 4, and 6 with wget, WebCite, and Heritrix and compared the resulting archived representation to the live representation of the resource . For the wget capture, we show the replay of the memento stored on `localhost` both with and without access to the Internet. Viewing the resource without Internet connectivity ensures that only content that is locally archived is loaded into the site (i.e., no leakage of the live web into the memento). This simulates the absence of the live web version of the resource.

[1] http://phantomjs.org/.

[2] https://www.webkit.org/.

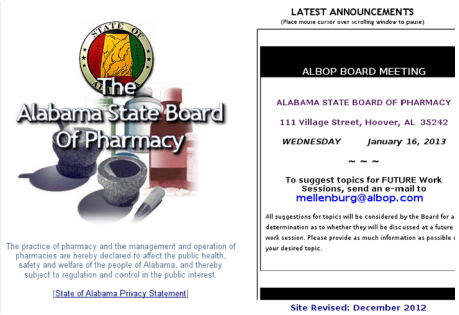


Fig. 3 The live version of http://www.albop.com/main.html (from the Archive-It collection) is perfectly archived using Heritrix, wget, and WebCite

Fig. 4 Live version of http://www.desbarresmanor.com/daytouring/history.html from the Twitter collection

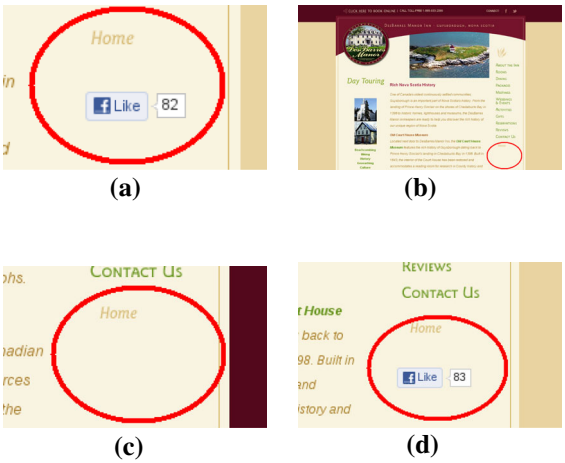


Fig. 5 As shown by these mementos, the site http://www.desbarresmanor.com/daytouring/history.html is mostly archived with minor content leakage **a** `Localhost` version captured with wget, with access to the Internet **b** `Localhost` version captured with wget, without access to the Internet **c** WebCite version **d** Wayback version

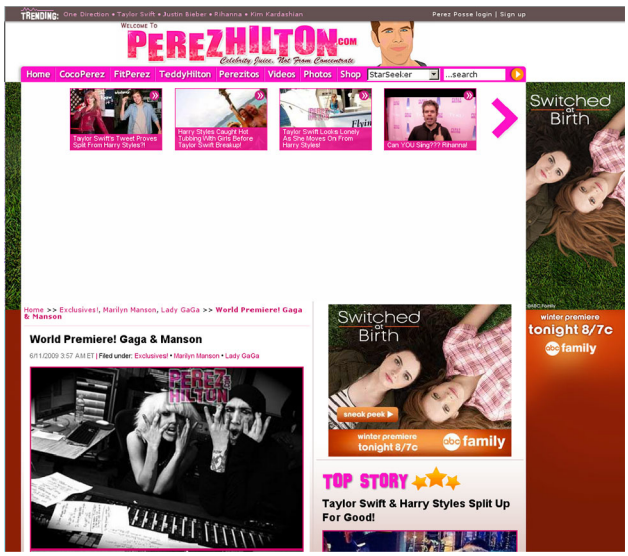


Fig. 6 Live version of http://perezhilton.com/2009-06-11-world-premiere-gaga-manson from the Twitter collection

Figure 3 shows a resource perfectly archived by the archival tools we examined. There is no leakage from the live web, and all requisite resources are captured by each tool. There is no JavaScript in this page, and the resources loaded into the page all originate from the same host. This can be considered the best-case archival scenario.

Figures 4 and 5 demonstrate a memento that is not perfectly archived but is "good enough". All of the "important" content—the content contributing to human understanding of the page—was captured; however, there were slight representation and stylistic issues. As an example, we have added red circles in the representations to highlight changes in content availability. The live representation (Fig. 4) has links and icons for sharing the resource over social media services, as well as a Facebook "Like" icon. The social media links disappear in the mementos (Fig. 5), and the Facebook "Like" icon disappears in the `localhost` without access to the Internet (Fig. 5b) and WebCite (Fig. 5c) versions. This issue with the representation is caused by leakage from the live web into our archives. Leakage is evidenced by the memento with Internet connectivity having the "Like" icon while the memento without Internet connectivity does not, because the JavaScript that loads the icon requests it from the live web. This means yesterday's HTML page has today's "Like" icon. In this case there is likely no problem, but this temporal inconsistency could be unacceptable in other cases (e.g., legal scenarios). Additionally, the WebCite memento (Fig. 5c) is missing a stylesheet, causing the background to be a dark red instead of the tan color of the other mementos and live version. This resource is an example of leakage from the live web and of the problems introduced by Ajax in a web page.

The example in Figs. 6 and 7 is poorly archived by most of the tools. The reader can see that the live and the `localhost` version with access to the Internet (Fig. 7a) are nearly identical, with the only difference being a missing banner at the top of the page. When observing the `localhost` version without access to the Internet (Fig. 7b), there are

(a) **(b)** **(c)** **(d)**

Fig. 7 As shown by these mementos, the site http://perezhilton.com/2009-06-11-world-premiere-gaga-manson is extremely difficult to archive **a** `Localhost` version captured with wget, with access to the Internet **b** `Localhost` version captured with wget, without access to the Internet **c** WebCite version **d** Wayback version

many more missing embedded resources. The missing content highlights the fact that a large number of files are not actually archived and cannot be loaded without the live web leaking into the memento. The WebCite (Fig. 7c) capture seems to be loaded properly. However, there is a significant amount of leakage including many of the JavaScript files that load content and several of the image files being imported into this page. The Wayback (Fig. 7) memento makes several requests for content that was not captured (as noted by the "Resource not in archive" messages), and the content on the page, particularly the personalized content, was corrupted during the archival process and displays as code. The resource consists almost entirely of JavaScript that loads content into the page dynamically, resulting in the reliance on leakage to load the requisite resources.

6 Experiment design

This experiment observes how well current web capture and archival tools can archive content and to identify characteristics of resources that are difficult to archive. Specifically, we study the completeness of the mementos captured by these tools and the impact that JavaScript has on archivability. We studied two sets of URIs with respect to archivability; the first is a set of Bitlys (shortened URIs shared in tweets) taken from Twitter, and the second is a set of human-curated URIs from Archive-It.

Each set of resources is captured with a set of archival tools, observed in their archived state, and each of the mementos is compared to the live *gold standard* version for completeness and, implicitly, archivability. The web page in its

live, native environment is the best version possible, and if an archival tool replicates the live web, it has perfectly captured and archived that resource. Each of the tools are compared to one another, and each of the datasets are compared and contrasted.

The second goal of our experiment is to study the evolution of archivability over time to determine the past and current trends. We observed the effects of JavaScript on archivability over time by loading mementos from the Internet Archive with PhantomJS and recording the HTTP response codes of the embedded resources. We determined whether the mementos were loaded from HTML or JavaScript. From this, we analyze, in depth, the nature of the memento and its archivability over time.

6.1 Datasets

This experiment utilizes two different datasets presented for comparison. The first dataset, the *Twitter* set, consists of Bitly URIs shared over Twitter. Shortened URIs are popular among social network services and the resources to which they redirect vary in expected life span [3]. The second dataset, the *Archive-It* set, was sampled from Archive-It collections. The Archive-It collections are created and curated by human users often corresponding to a certain event (e.g., National September 11 Memorial Museum) or a specific set of web sites (e.g., City of San Francisco). The Twitter and Archive-It sets are made up of fundamentally different resources, as shown in Tables 2 and 3; the differences are investigated in Sect. 6.1.3. There is no overlap between the two sets.

We discarded non-HTML representations (e.g., JPEG and PDF) from both sets. The purpose of this study is to observe how well the embedded resources are converted from live to archived. Non-HTML representations do not contribute to this study and are assumed to be perfectly archived by all tools.

6.1.1 Twitter

We collected the Twitter URIs through the Twitter Garden Hose[3] in October 2012. This set consists of 1,000 URIs and represents resources that Twitter users thought to be important enough to share with others on social media but which have not necessarily been actively archived. With the non-HTML representations removed, the Twitter set has 901 URIs.

6.1.2 Archive-It

The Archive-It set consists of the entire set of URIs belonging to the collections listed on the first page of collections at Archive-It.org[4] as of October 2012. This resulted in 2,093 URIs that represent a collection of previously archived and human-curated URIs. To make the datasets equal in size, we randomly sampled 1,000 URIs from the set of 2,093. As shown in Table 2, the Archive-It set has a lower proportion of non-HTML content than the Twitter set. With the non-HTML representations removed from the randomly sampled set of 1,000 this collection has 960 URIs.

Table 1 Example URIs

	URI Delimiter	Example URI
URI Fragment	#	http://www.example.com/index.html#fragment
URI Parameter	?	http://www.example.com/index.php?key1=value1
HashBang URI	#!	http://www.example.com/index.php#!state1

6.1.3 Collection differences

In addition to the different collection venues, the datasets also differ in the server-side and client-side values passed to the resource via URI as parameters and fragments, respectively. URI fragments (first row of Table 1) identify a point or portion of content within the resource and are processed by the client (after the HTTP response), not sent to the server when dereferencing the URI. Instead, they are an offset applied to the representation returned by the server and depend on MIME type, such as a tag within the HTML or a specific second in an audio file. Server-side parameters (second row of Table 1) identify values for use in constructing the representation (before the HTTP response) using key-value pairs and are passed to the server when dereferencing the URI.

The Twitter set has more URIs containing fragments than the Archive-It set. Additionally, there are far more URIs with parameters in the Twitter set. Our complexity measure (defined in Sect. 7.1) has $\overline{\text{UC}}_{\text{Twitter}} = 1.76$ and $\text{UC}_{\text{Twitter}_\sigma} = 0.312$ meaning there are nearly 2 URI parameters in the Twitter data set for each URI, while the complexity of Archive-It URIs has $\overline{\text{UC}}_{\text{Archive-It}} = 0.16$, $\text{UC}_{\text{Archive-It}_\sigma} = 0.174$ meaning they are mostly without parameters. Note that only 3 URIs from the Twitter data set had both parameters and fragment identifiers (0.3 % of the collection). Only 2 URIs from the Archive-It data set had both parameters and fragment identifiers (0.2 % of the collection).

Additionally, 93.1 % of URIs in the Archive-It set are top-level URIs whereas (in an aesthetically pleasing coincidence) 93.1 % of the Twitter set is several levels away—or

[3] https://dev.twitter.com/docs/streaming-apis/streams/public.

[4] http://www.archive-it.org/explore/?show=Collections.

Table 2 Content features of each collection

Collection	Statistical breakdown of content				
	PDF (%)	Images (%)	Other content (%)	HTML (%)	HTML content containing JavaScript (%)
Twitter $n = 901$	0.3	1.3	3.7	84.8	98.7
Archive-It $n = 960$	4.4	0.6	1.3	93.7	97.1

Table 3 URI features of each collection

Collection	Statistical breakdown of URIs							
	COM (%)	EDU (%)	ORG (%)	GOV (%)	Other domains (%)	With fragments (%)	With parameters (%)	Top-level URIs (%)
Twitter	81.6	0.3	3.5	0.2	14.1	1.4	30.5	6.9
Archive-It	47.9	7.7	10.9	12.7	20.6	0.8	41.2	93.1

deep links—from each URI's respective top level domain. We explore URI depth further in Sect. 6.3. The Twitter set contains slightly more personalized client-side content than the Archive-It set, as suggested by the features of each set's URIs. Twitter URIs have more client-side parameters as observed by the 0.6 % more fragments than the Archive-It set. These fragments specify client-side parameters that are presumably used for client-side processing. There are also 2 hash-bang URIs [69] (third row in Table 1) in the Twitter set and 5 in the Archive-It set. These URIs identify parameters for automatic client-side processing by JavaScript.

The HTML of each set is measured and compared using several different metrics as discussed in Sect. 6.3. Each set consists mostly of resources containing JavaScript (98.7 and 97.1 % of the Twitter and Archive-It sets, respectively, contain JavaScript) (Table 2).

While the .com, .edu, .org, and .gov top level domains (TLD) are defined in Table 3, the "Other Domains" are not. These include TLDs such as .tv or .im, as in these URIs:

```
http://www.geekbrief.tv/best-ad-from-microsoft-ever

http://tr.im/lPRA#user1244749838078
```

Governmental web sites comprise 57.6 % (533) of the Archive-It collection. Governmental URIs are not limited to the `.gov` TLD (such as http://www.ethics.alabama.gov) but can include other suffixes (such as http://ali.state.al.us/). The Twitter collection only has three government-owned sites (0.3 % of the collection). Governmental websites are mandated to conform to web accessibility standards for content but commercial websites do not have this restriction. The government and non-government sites perform nearly identically in our measurements—differing only by a maximum of 6 % across all metrics and a Chi-square test provided a $p = 0.22$ showing no statistical significance across our measurements—so we have opted to measure the collections in their entirety instead of measuring only the government and non-government URIs independently.

6.2 Archiving the resources

We archived each URI from the two datasets with a suite of tools, including submitting to WebCite, capturing with wget, and archiving with Heritrix. The method we used to access the resulting mementos depended on the archival tool. We viewed the WebCite-captured mementos using the WebCite web service, Heritrix-captured mementos via a local installation of the Wayback Machine, and wget-captured mementos through a client (browser) from the `localhost` server. We also captured each URI in its native live-web environment to establish a baseline.

In our figures, we label the WebCite mementos as "WebCite" (playback via the WebCite web service), wget mementos as "Disconnected" (playback via a server with no out-bound Internet access), Heritrix mementos as "Wayback" (viewed through the Wayback machine), and live-web resources a "live" (viewed through a client).

In the remainder of this section, we detail the methods used to archive the resources in the dataset and view the resulting mementos. We measure leakage in the mementos (Sect. 6.4), which only occurs when replaying the mementos in a client, and not during the archival process. Because a primary goal of this research is to establish the impact of JavaScript on the mementos created by the tools, we classify the mementos according to the environment in which they are viewed. That is, we label the mementos created by Heritrix as "Wayback" mementos because the mementos are viewed via the Wayback Machine. Similarly, we label the WebCite mementos (viewed in a client through the WebCite web service), live

(live-web resources viewed through a client), and localhost mementos (archived with wget and viewed through a client from the localhost server on which the resources are stored).

We loaded the Twitter and Archive-It live-web resources using PhantomJS to establish the baseline to which all our mementos were compared. WebCite and wget do not employ PhantomJS or other headless browsing techniques during archiving.

Using PhantomJS, we downloaded each of the resulting representations in its native format (such as HTML) and generated a PNG screenshot of the representation. We also captured and logged the HTTP response codes generated during the load and viewing of each URI. These logs establish a set of resources required to view the page and also to identify whether or not the capturing tool was successful in capturing all necessary resources to create a complete memento. The HTTP response logs from a memento are compared to the response logs from the live viewing to establish the degree of completeness of the memento.

6.2.1 Heritrix 3.0.1 and the Wayback Machine

We installed a local version of Heritrix 3.0.1 and used it to crawl each of the URIs in this study. A local instance of the Wayback Machine was installed on `localhost` so the resulting WARC files could be viewed. Archival resources should load embedded resources from the local instance of the Wayback Machine. In a scenario such as this, the proxy mode is normally run with the Wayback Machine to prevent leakage but was omitted from this experiment since we were monitoring header values for leakage at load time and this Wayback instance was limited to a target crawl.

6.2.2 wget

The wget application was used to archive content from the live web onto the `localhost` machine. Our goal was to make sure the entire site was captured, complete with requisite resources and converting the links to be relative to the local directory. The following command was used to download the content:

```
wget -k -p -E -H -w 10 -o ./logs/$i.log
     -D $theDomain $toGet
```

This command captures the header logs in the `./logs/$i.log` file (where `$i` is the identifier for the log file), limits the wget capture to the target domain identified by `$theDomain`, and captures the target site `$toGet`. The locally captured content can then be viewed from `localhost` and measured against the native, live version of the resource.

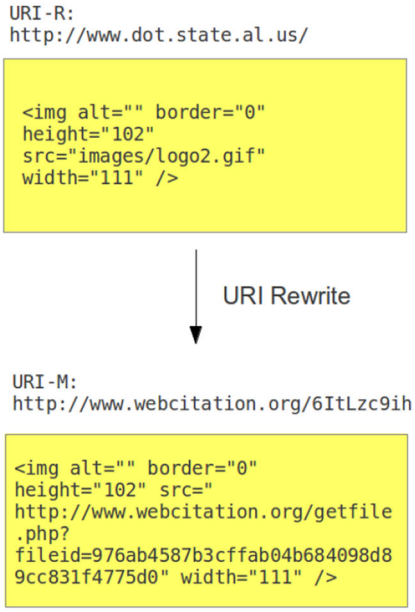


Fig. 8 URI re-writing in WebCite converts embedded URI-Rs to URI-Ms

6.2.3 WebCite

WebCite is an on-line, on-demand archival tool that offers real-time archiving of content. When WebCite archives a URI-R, it opaquely rewrites embedded URI-Rs to URI-Ms but does not maintain the semantics of the URI-Rs. The semantics are not required for WebCite's page-at-a-time archival practices to be successful. For example (Fig. 8), when WebCite archives a URI-R, the embedded image identified by a URI-R is rewritten to point to a URI-M.

6.3 Resource metrics

We measure resource complexity based on the HTML representation and the URI identifying the resource.

6.3.1 URI complexity

The complexity of a URI is measured through the depth (URI's number of levels down—or distance—from the TLD), the number of client-side parameters, and the number of server-side parameters. The client- and server-side parameters are measured by Eq. 1. The arithmetic mean of the sum of these measures is taken to give a URI complexity measure (UC) as noted in Eq. 2.

$$F = \max(|\text{client-side parameters}|, |\text{server-side parameters}|) \quad (1)$$

$$\text{UC} = \frac{|\text{Depth}| + F}{2} \quad (2)$$

For example, a URI such as

```
\url{http://www.tennessee.gov/ecd/}
```

from the Archive-It collection has a complexity of 0.5. The depth is 1 (`/ecd/`), and there are no server- or client-side parameters. Alternatively, a URI such as

```
http://edition.cnn.com/2009/SPORT/
football/
06/11/ronaldo.real.madrid.manchester/
index.html?eref=edition
```

from the Twitter collection has a complexity of 3.5. The depth is 6 (/2009/SPORT/football/06/11/ronaldo.real.madrid.manchester/), there is one server-side parameter but no client-side parameter ($\max(1, 0) = 1$), which totals to 7/2 = 3.5. The URI

```
http://www.tridentgum.com/#/trident
cares4kids/
```

has a client-side parameter that does not have a corresponding anchor in the HTML ($\max(1, 0) = 1$) and has a complexity of 3/2 = 1.5.

6.3.2 Content complexity

Content complexity can be measured in many different ways (e.g., number of HTML tags or presence of multimedia). This experiment focuses on JavaScript, since we suspect it is the major contributor to dynamically requested resources. Content complexity in this experiment is measured as the number of `< script >` tags loading JavaScript, defined as CC in Eq. 3. This is the number of JavaScript files or code segments being loaded and the number of occurrences of embedded JavaScript code occurring in the resource code. Intuition suggests that the more JavaScript embedded in a page, the more likely it will change its state on the client, and the more difficult it is to archive. Our previous attempts at assessing content complexity focused on lines of JavaScript, but this metric was misleading (for example, due to minimized files or from including external libraries like jQuery), so we opted instead for a count of `< script >` tags.

$$\text{CC} = \sum \text{script tags} \in \text{HTML} \tag{3}$$

6.4 Requests from HTML and JavaScript

We compare the number of requests for resources referenced in the HTML to the number of requests coming from run-time sources, such as JavaScript. We compared the URI-Ms referenced in the HTML and the requests observed by PhantomJS. Requests for URI-Ms referenced directly in the HTML are classified as HTML-loaded resources, while those not referenced in the HTML have been loaded at run-time by JavaScript. We define JavaScript-loaded resources according to Eq. 4 and use this method of measurement when referring to the number of resources loaded by JavaScript versus HTML.

$$\text{JS Resources} = \text{Resources Loaded} - \text{Resources in HTML Tags and CSS} \tag{4}$$

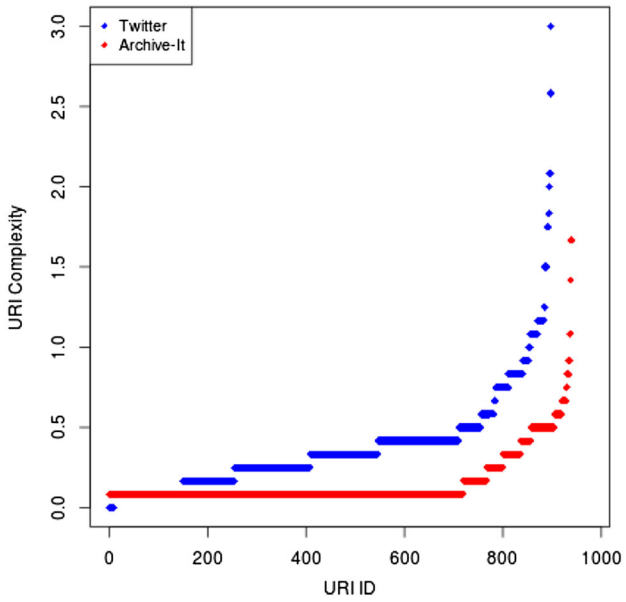


Fig. 9 URI complexity measure (UC)

7 Resource set analysis

The nature of the resources in each set is fundamentally different. We analyzed the complexities of the two datasets.

7.1 URI complexity

The links shared over Twitter tend to be *deep links*, with many layers of separation between the TLD and the shared resource. Additionally, the Twitter URIs tend to have client- or server-side parameters, which are assumed to identify a specific set of information or a special representation of the resource for the users' followers. Alternatively, the Archive-It collection, which has seed URIs chosen by human curators, tends to be made of more top-level domains and have fewer parameters and fragments.

The $\overline{\text{UC}}$ (Eq. 2) of the Twitter collection is 0.377 with $\text{UC}_\sigma = 0.3110$. The Archive-It collection has a $\overline{\text{UC}} = 0.166$ with $\text{UC}_\sigma = 0.2341$. The Archive-It collection is a lower $\overline{\text{UC}}$ than the Twitter collection (as seen in Fig. 9), supporting

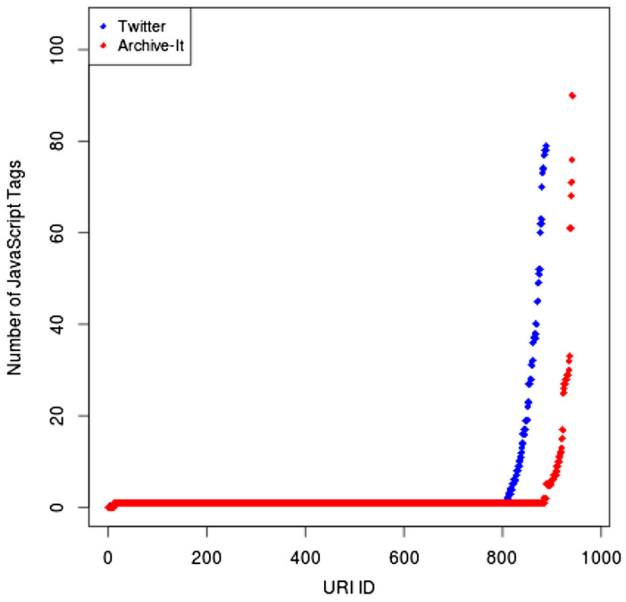


Fig. 10 CC of the HTML

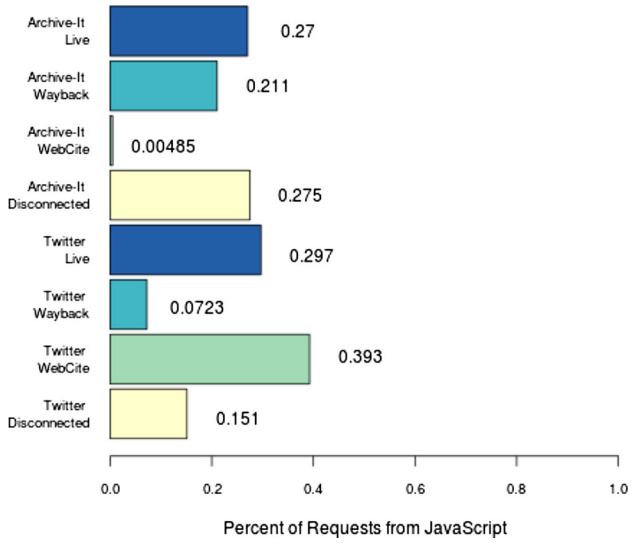


Fig. 11 Percentage of resource requests from JavaScript show patterns similar to CC

the theory that the human-curated Archive-It collection deals more with higher-level URIs than the shared links of Twitter.

7.2 Content complexity

The Twitter set has a $\overline{CC} = 4.78$ with $CC_\sigma = 16.23$. The Archive-It set has an average of $\overline{CC} = 2.16$ with $CC_\sigma = 6.87$. The Archive-It set has, on average, approximately half as many < `script` > tags as the Twitter set and a CC_σ that is half that of the Twitter set (as shown in Figs. 10, 11).

We created a list of resources referenced in the HTML tags and CSS. The difference between the total set of resources loaded and the resources referenced in the HTML and CSS are assumed to come from JavaScript. The comparison of JavaScript and HTML requested resources is shown by dataset and archival tool in Fig. 11. As expected, the CC measure is directly related to the number of JavaScript requests to external resources. By taking the average across all environments, we found that the Twitter set resources load 16.3 % of the requisite resources through JavaScript (presumably Ajax), whereas 18.7 % of resources are loaded via JavaScript in the Archive-It set. This is contrary to our hypothesis that increased CC will produce more resource requests from JavaScript. The Twitter set, which has more embedded JavaScript ($\overline{CC} = 4.78$), makes fewer requests to content with JavaScript than the seemingly less complex Archive-It set ($\overline{CC} = 2.16$). As discussed in Sect. 8.2, the nature of the JavaScript requests plays a more important role in archivability.

8 Archiving experiment discussion

In this study, a perfectly archived resource has all of its constituent resources copied into the archive environment and is independent from the live web. Failure to satisfy this capture will result in two possible outcomes. The first is an HTTP 400 or 500 class (non-200) of response header when accessing the missing resource(s). The second is leakage, which occurs when archived resources make requests to and include content from the live web when they should be accessing archived content only. We have provided an example in our research blog [15]. As such, the archivability of a resource is measured by the number of embedded resources that archival tools can capture.

8.1 Missing resources

The performance of each archival tool and each collection can be analyzed with several metrics. Most important is the number of resources missing from the mementos. The average number of missing responses is expressed as the percentage of non-200 responses returned when requesting the requisite resources. An HTTP 200 response indicates a successful load of a resource into the page, while a non-200 response indicates a missing resource (300-style redirects are omitted from this calculation). These averages for each collection and tool are shown in Fig. 12. Each bar represents a collection and an environment. The bars representing the live web establish the *gold standard*, or "best case", archival goal. Live resources are not perfect—they sometimes request resources that are unavailable and do not receive 200 responses for 100 % of their requests. However, they do establish a ceiling of possible performance. The Archive-It set receives an HTTP 200

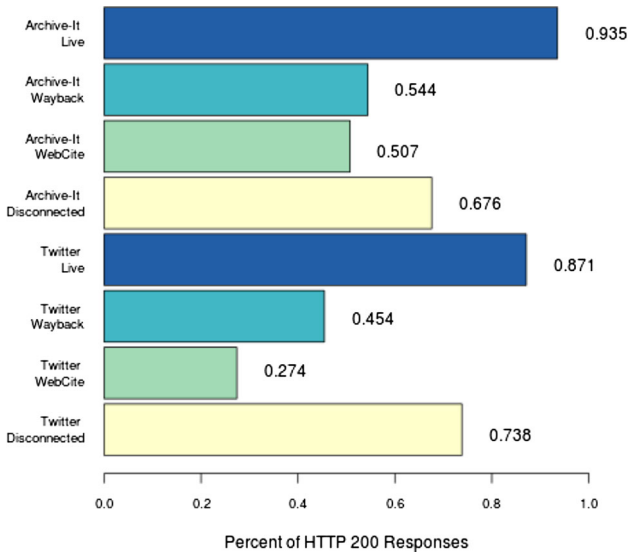


Fig. 12 The HTTP 200 response codes for embedded resources

response for 93.5 % of all requests and the Twitter set receives an HTTP 200 response for 87.1 % of all requests.

There is a clear difference between the archival tools used and the number of resources successfully loaded. The percent of HTTP 200 responses from the Wayback Machine across all collections is 69.1 % per page, meaning roughly two-thirds of all resources needed by the page to provide a complete representation (as compared to the live version) were archived by the Heritrix crawl and viewed in the Wayback Machine. The WebCite service results in 45.2 % of all resources being captured. The wget tool (labeled as "Disconneted") captures 51.8 % of the resources.

We can demonstrate trends when analyzing the statistics in further depth. The resources in the Twitter collection, on average across all archived environments, produced 38.2 % more HTTP non-200 response headers as mementos than the live state. The Archive-It set fares slightly better, with 37.7 % more HTTP non-200 responses in their archived state. However, the WebCite results show the greatest difference between the Twitter and Archive-It sets (59.7 % HTTP 200 responses vs. 42.8 % 200 responses, respectively). If this difference is excluded, the Archive-It set performs better with only a reduction of 30.9 % HTTP 200 responses, as compared to the Twitter collection's 36.5 % reduction.

8.2 Leakage

In an archive, mementos' HTTP requests differ from that of the live web; the WebCite, wget, and Wayback tools reduce the number of HTTP requests to different domains in order to change all of the external requests to requests for mementos within the archive. Resources referenced in HTML should be rewritten to reference a URI-M instead of a URI-R. Failure

Fig. 13 A temporal inconsistency created by leakage in this cnn.com memento

to rewrite the URI-R to a URI-M will result in a reference to an external domain when the memento is loaded. Archival services cannot always predict what embedded resources will be loaded (such as when loaded via JavaScript) and are unable to perform a rewrite of the URI-R to a URI-M, resulting in leakage. Leakage is particularly problematic since it portrays live resources as archived.

As an example, we can observe a cnn.com memento from the Wayback Machine at URI-M

`http://web.archive.org/web/20080903204222/http://www.cnn.com/`

(Fig. 13). This memento from September 2008 includes links to the 2008 presidential race between McCain–Palin and Obama–Biden. However, this memento was observed on September 28, 2012—during the 2012 presidential race between Romney–Ryan and Obama–Biden. The memento includes embedded JavaScript that pulls advertisements from the live web. The advertisement included in the memento is a "zombie" resource that promotes the 2012 presidential debate between Romney and Obama. The memento attempts to load 144 embedded resources, 133 of which (92.4 %) successfully return a 200 response. The memento also attempts to load four resources from the live web. This drift from the expected archived time seems to provide a prophetic look at the 2012 presidential candidates in a 2008 resource.

The proportion of requests to different domains decreases when an archival tool captures the embedded resources and makes them local to the archive. The amount of requests to different domains (and therefore leakage) in the Twitter sets decreases by 43.2, 22.5, and 71.2 % for the WebCite, wget, and Wayback tools, respectively. The Archive-It domain observes decreases of external requests of 32.3, 1.5, and 42.6 % for the WebCite, wget, and Wayback tools, respectively. This shows that the archival tool affects the number of external HTML requests, with the wget tool doing the least to mitigate leakage and the views in the Wayback Machine lim-

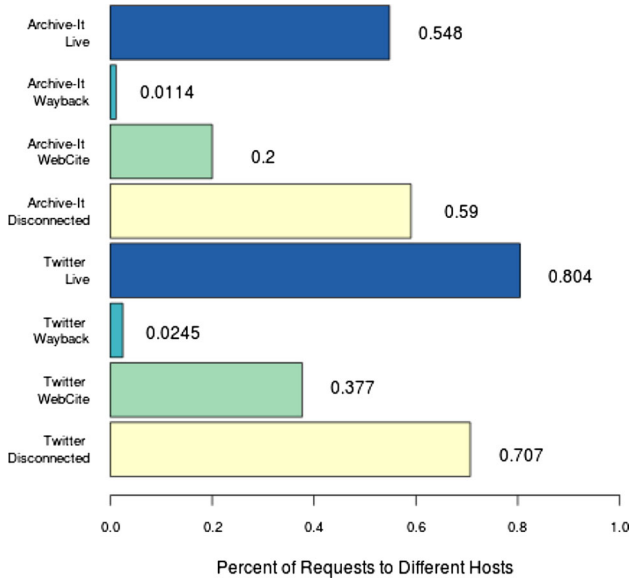


Fig. 14 Percentage of resource requests going to remote hosts from both HTML and JavaScript

iting leakage the most. As shown by Fig. 14, each archival tool impacts the target location of the requests for content the same way for each collection.

As in Sect. 7.2, we averaged across all environments shown in Fig. 14 and found that 62.0 % of Twitter resources loaded by HTML come from the host domain, while 38.0 % come from an external source. This is roughly similar to the Archive-It set, which loads 74.7 % from the host domain and 25.3 % from external domains. The Archive-It resources are more centrally organized, loading fewer resources from external domains. This analysis of resource requests shows that JavaScript and HTML make requests for content in similar patterns, with the ratio of local versus external content following the same pattern. Additionally, the increased JavaScript requests to external content increases the opportunity for leakage and therefore decreases the archivability of a resources.

The JavaScript requests to local vs. external resources are similar to that of the HTML requests (Fig. 11). The Twitter set gets 52.2 % of its JavaScript-requested resources from external domains, while the Archive-It set gets 70.1 % of its JavaScript-requested resources from external domains, suggesting that JavaScript is a source of leakage. We observe the modification of external requests for resources in the Twitter set as a reduction by 42.8, 9.8, and 78.0 %, respectively. The Archive-It set's capture by the WebCite, wget, and Wayback tools reduce the number of JavaScript requests to different domains by 33.5, 0.6, and 48.0 %, respectively. Again, this shows that the wget tool does little to mitigate leakage, and the Wayback Machine performs best by reducing requests to the live web. Heritrix provides the most complete collection and performs the best in this analysis. Since Heritrix uses PhantomJS to identify resources for archiving, it can limit leakage by anticipating JavaScript requests from mementos.

The percentage of external requests is highest in resources captured with WebCite with 14.6 % of all resource requests resulting in leakage. This holds true regardless of dataset. The wget tool captures are second with 14.2 % of all resource requests resulting in leakage. The Wayback captures experience the least leakage with only 7.1 % of all resources requests resulting in leakage.

The Twitter set observes the most leakage (as seen in Fig. 14). When the Twitter resources were captured with the wget, WebCite, and Heritrix tools, the leakage observed is (on average), 49.9, 8.6, and 1.3 %, respectively. The Archive-It set only sees an average leakage of 44.1, 4.7, and 1.9 % for each tool.

As shown, the Archive-It set is more easily archived, provides more complete mementos, and has fewer instances of leakage than the Twitter set.

The Archive-It set has many resources that are perfectly archived. The Twitter set only had 4.2 % of the resources that were perfectly archived by each tool, while 34.2 % of the Archive-It set was perfectly archived by each tool. This shows that the Archive-It set is much more archivable than the Twitter set. The resources shared over Twitter were much more difficult to capture than the content humans identify as historically or socially important. Twitter-shared resources are important enough to share with online acquaintances but cannot be effectively captured with current technologies. As shown, the Twitter set contained more JavaScript than the Archive-It set. As such, we conclude that resources relying on JavaScript to render a final representation are less archivable than those that are pure HTML.

9 Archivability over time

As we begin to discuss our second contribution—measuring the migration toward JavaScript and its impact on archiving over time—we must first understand the evolution of JavaScript. Web browsers render the structural HTML of a resource, a stylistic portion (implemented via CSS) and enable client-side behavior and state change (implemented via JavaScript). JavaScript code—which is embedded in the HTML—must be executed by the client after the representation has been returned by the server. To optimize execution, many crawlers do not include a JavaScript rendering engine, focusing only on the structural HTML and other embedded resources. This is problematic in that some resources' URIs might be determined at runtime or included in the page because of JavaScript's manipulation of the Document Object Model (DOM), and crawlers may not archive these resources because they were the output of unexecuted code.

Early versions of Heritrix had limited support for JavaScript. The crawler's parsing engine attempted to detect URIs in scripts, fetch the scripts, and recursively repeat this process with the intention of ensuring maximum coverage with the archive creation process. Recently, Heritrix was rewritten to be built on top of PhantomJS, a headless WebKit with JavaScript support, increasing the archiving resources loaded through JavaScript. At the time of authoring this paper, version 3.0.1 was currently in production at the Internet Archive and targeting the incorporation of a headless browser like PhantomJS [61].

With the exception of the recent versions of Heritrix (i.e., the inclusion of Umbra and production version 3.0.1), crawlers do not execute the client-side JavaScript and, therefore, have no way of determining what resources JavaScript will load. We investigate the performance of Heritrix over time by collecting and measuring the completeness of mementos in the Wayback Machine.

9.1 Impact of accessibility

Many factors can impact archivability. As we have shown, current tools can capture Archive-It resources more completely than Twitter resources. The nature of the collections may impact this behavior. For example, the Archive-It set contains more government-owned URI-Rs. Government URIs are often perfectly archived, with 85 of the 124 government URIs being perfectly archived (68.5 %). This shows that government URIs are more frequently perfectly archived than the rest of the mementos measured (the entire Archive-It collection was archived perfectly 34.2 % of the time).

Section 508 [1] gives suggestions on how websites should comply to be considered accessible. Also, the World Wide Web Consortium (W3C) through the Web Content Accessibility Guidelines (WCAG) gives concrete ways for developers to evaluate their creations to make them more accessible [17]. Hackett et al. detail the accessibility of archives on a pre-Ajax corpus (from 2002) and enumerate specific features that make a page accessible in the legal sense [30]. Much of this information is beyond the scope of this study (e.g., animated GIFs). Other efforts were made following the document's release on a metric to evaluate web accessibility [57], but also fail to consider an asynchronous web, which was less prevalent at the time.

All United States government sites are advised to comply with Section 508. These mandates guide content authors by limiting embedded programming languages and multi-state client-side representations. It stands to reason that pages adhering to Section 508 are easier for tools like Heritrix to capture in their entirety, which may influence the archivability of the Archive-It set.

In much of the same way that accessibility guidelines encourage content to be accessible to the end-user, complying with the guidelines also facilitates accessibility of the content (displayed by default or as the result of a JavaScript invocation) to the user agent. Conventional web browsers (e.g., Google Chrome, Mozilla Firefox) are capable of retaining this content in memory, even if not displayed. Browsers render a representation of a web resource. Contrary to modern browsers (e.g., Safari), archival crawlers like Heritrix; are not equipped with the capability to access, process, and capture JavaScript-dependent representations that use potentially inaccessible (in the WCAG sense) features.

Conventional web browsers are usually the first to implement inaccessible features. These features often are implemented in conventional web browsers using JavaScript and often disregard accessibility standards for the goal of enticing users with more personalzied and interactive representations. Because the content reliant on JavaScript is not accessible to archival crawlers, requiring content on the web to be more accessible would prevent JavaScript-dependent content and newer browser features from being used. Additionally, avoiding JavaScript-dependent content will reduce the ability for web resources to provide personalized and interactive content to users, presumably resulting in a less appealing representation.

9.2 Fetching data

We investigate the archivability of our resources over time by looking at the captures by the Internet Archive. In effect, this is a black-box test of archiving by examining the output of the archival process over time. Since application technologies have increased in prevalence over time, we expect to see memento completeness decrease (and therefore, memento quality decrease) as time progresses.

When considering the age and longevity of resources in the archives, it is useful to understand how the different collections differ in age. We used the carbon dating service [62] to estimate the age of the URI-Rs in our collections. The results are presented in Fig. 15. The carbon dating service was unable to reliably determine a creation date for 35.5 % of our collection, so the sample size in this graph has been reduced to include only the URI-Rs with reliably determined creation dates. Note that the Archive-It set is evenly distributed over time, while the Twitter resources are younger. This is intuitively satisfying given the different purposes of the Archive-It and Twitter services.

To increase the heterogeneity of our dataset (mitigating the impact of accessibility standards and the age of URI-Rs), from here on we will combine the Twitter and Archive-It data sets.

To empirically observe the change in archivability over time of each of the URI-Rs in our collections, we acquired the TimeMap of each URI from the Internet Archive to produce output like Fig. 16. We captured screen shots and HTTP

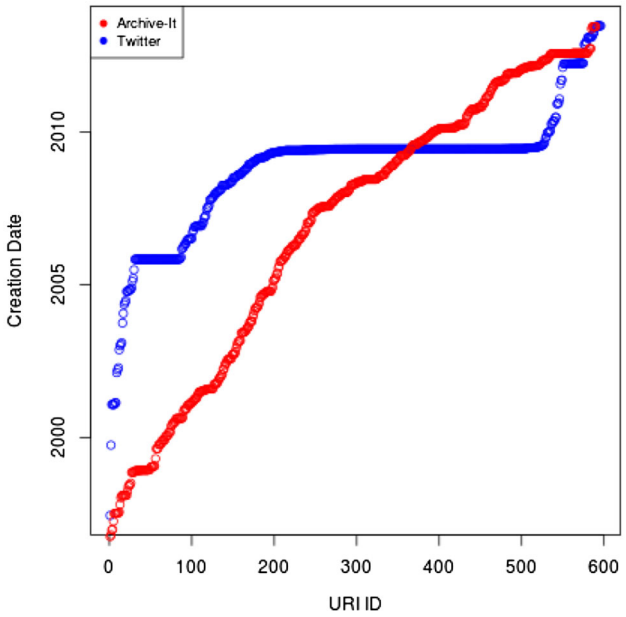


Fig. 15 The Twitter collection ($n = 596$) is, on average, younger than the Archive-It collection ($n = 590$)

requests for one memento per year of each URI in our collections in an effort to analyze the results of archival efforts performed in the past.

The http://www.doc.alabama.gov/ resource (CC = 0.43) appears to be perfectly archived throughout time in the Internet Archive. This holds true when accessing the mementos with and without JavaScript enabled. The mementos are visible in Fig. 17. All mementos from 2007 to 2013 neither request a resource nor receive a non-200 HTTP response through JavaScript with the exception of a single memento with an archival date time of October 21st, 2011 that requests and misses 6 resources via JavaScript.

The http://www.cmt.com/ resource (CC = 0.87) varies in archivability (seen in Fig. 18l), with an increase in missing resources over time. The mementos from 2009–2013 (Fig. 18l–p) are missing almost all central article and image content; this is when CMT.com implemented content loading through jQuery and Ajax, resulting in an average of 9.7 % of all URI-Ms requested to be missed because of JavaScript. CMT.com's addition of Ajax and associated technologies drastically increased the feature gap between the resource and the crawlers performing the archiving, while the doc.alabama.gov site limits its use of JavaScript and is more complete over time.

9.3 Challenges in measuring past performance

In our research, we have omitted calculating CC and other archivability metrics from mementos in our collections since we do not have the corresponding URI-R from which to compute a baseline. Several scenarios could take place that impact our archivability metrics:

- mementos may have been damaged (missing embedded resources) after the archival process took place due to implementations of the robots protocol (eliminating mementos from external domains) or limited by policy (such as the case with Archive-It);
- inavailability of embedded resources due to server-side failures or transient errors in the archives;
- embedded resources may not have been available to the crawler at archival time.

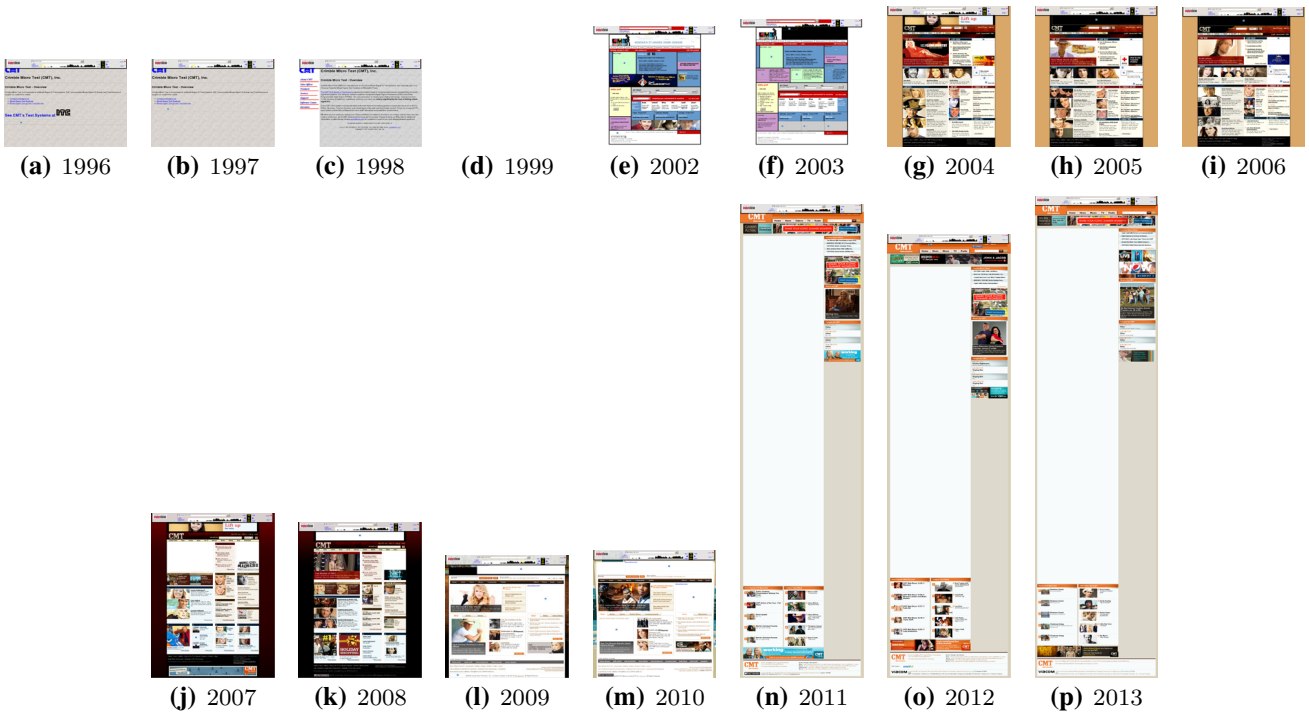
Without an understanding of the state of the live resource at archival time, we cannot accurately assign archivability metrics to the mementos.

```
<http:///www.doc.alabama.gov/>; rel="original",
<http://web.archive.org/web/timemap/link/http:///www.doc.alabama.gov/>; rel="self";
 type="application/link-format"; from="Mon, 08 Jan 2007 17:48:19 GMT"; until="Sun, 28 Jul 2013 15:16:29 GMT",
<http://web.archive.org/web/http:///www.doc.alabama.gov/>; rel="timegate",
<http://web.archive.org/web/20070108174819/http://www.doc.alabama.gov/>; rel="first memento";
 datetime="Mon, 08 Jan 2007 17:48:19 GMT",
<http://web.archive.org/web/20070113182156/http://www.doc.alabama.gov/>; rel="memento";
 datetime="Sat, 13 Jan 2007 18:21:56 GMT",
<http://web.archive.org/web/20070118175605/http://www.doc.alabama.gov/>; rel="memento";
 datetime="Thu, 18 Jan 2007 17:56:05 GMT",
<http://web.archive.org/web/20070123202638/http://www.doc.alabama.gov/>; rel="memento";
 datetime="Tue, 23 Jan 2007 20:26:38 GMT",
<http://web.archive.org/web/20070519200310/http://www.doc.alabama.gov/>; rel="memento";
 datetime="Sat, 19 May 2007 20:03:10 GMT",
<http://web.archive.org/web/20070617053244/http://www.doc.alabama.gov/>; rel="memento";
 datetime="Sun, 17 Jun 2007 05:32:44 GMT",
<http://web.archive.org/web/20070621140945/http://www.doc.alabama.gov/>; rel="memento";
 datetime="Thu, 21 Jun 2007 14:09:45 GMT",
```

Fig. 16 An abbreviated TimeMap for http://www.doc.alabama.gov/

(a) 2007 **(b)** 2008 **(c)** 2009 **(d)** 2010 **(e)** 2011 **(f)** 2012 **(g)** 2013

Fig. 17 The http://www.doc.alabama.gov/ mementos are perfectly archived through time since they limit their reliance on JavaScript to load embedded resources **a** 2007 **b** 2008 **c** 2009 **d** 2010 **e** 2011 **f** 2012 **g** 2013



(a) 1996 **(b)** 1997 **(c)** 1998 **(d)** 1999 **(e)** 2002 **(f)** 2003 **(g)** 2004 **(h)** 2005 **(i)** 2006

(j) 2007 **(k)** 2008 **(l)** 2009 **(m)** 2010 **(n)** 2011 **(o)** 2012 **(p)** 2013

Fig. 18 CMT.com over time. Changes in design and thus the technologies used are easily observable after mementos archived in 2009–2013 (Fig. 18l–p), which is when jQuery is introduced into the page and used to load embedded resources **a** 1996 **b** 1997 **c** 1998 **d** 1999 **e** 2002 **f** 2003 **g** 2004 **h** 2005 **i** 2006 **j** 2007 **k** 2008 **l** 2009 **m** 2010 **n** 2011 **o** 2012 **p** 2013

While calculating archivability metrics, we must also consider the practices used by each archival tool. For example, the Wayback Machine inserts additional JavaScript and other content to generate the banner and headers to appear in the mementos. Other archival services have similar practices for which we must account. It becomes difficult to measure the original resource when HTML is modified after archiving and during the dereferencing of mementos. This after-archiving injection and loading of scripts and other embedded resources by the archives will impact the calculations we have performed in this research such as CC. The live resources may have also been bound to different standards, and we have no a priori knowledge of these standards or their implementations. As such, we have omitted these metrics and measurements.

We instead investigate the availability of constituent mementos within our collection over time. The completeness of the memento as it existed at observation time and the source of the embedded resources (either from the HTML or JavaScript) provide a measurement of how JavaScript impacts memento archivability. We loaded all URI-Ms listed in the Internet Archive TimeMaps of URI-Rs in our collection and recorded the resources loaded and their HTTP response codes. We also recorded the origin of the request; specifically, we note whether the resources were loaded because of inclusion in the HTML or as a request from JavaScript. We did not track availability of each embedded resource within a URI-M over time during this experiment, but rather considered just the response codes generated from individual URI-M accesses.

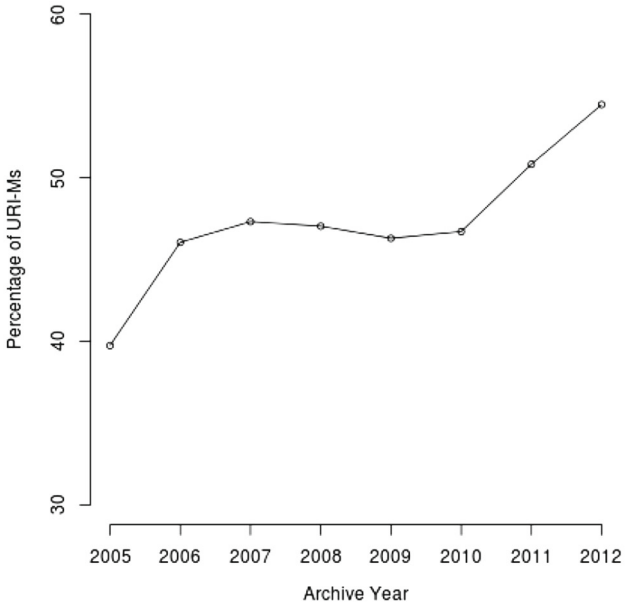
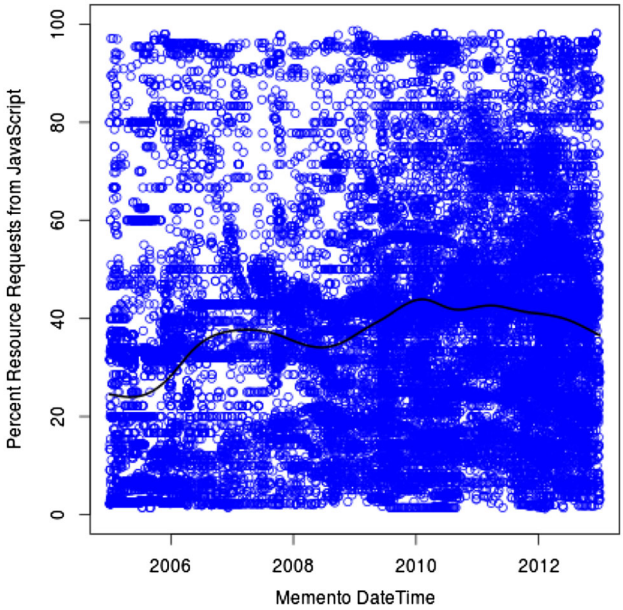
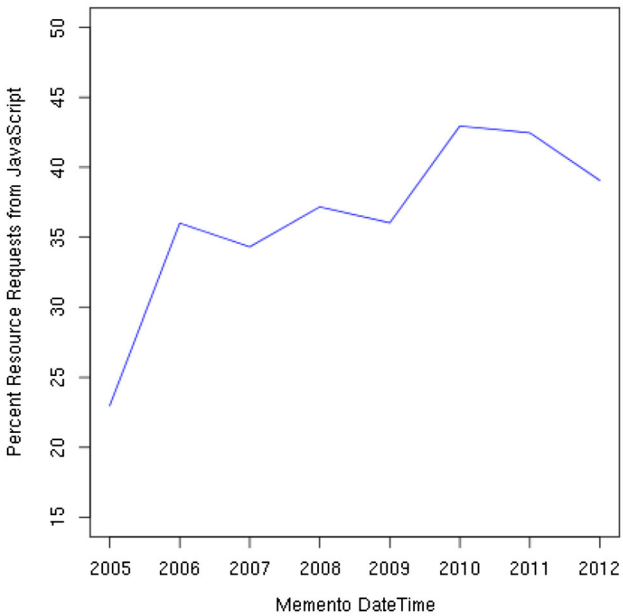


Fig. 19 Resources are using more JavaScript to load embedded resources over time



(a) More embedded resources are being loaded via JavaScript over time.



(b) On average, resources are increasingly being loaded by JavaScript per year.

Fig. 20 JavaScript is responsible for loading an increasing proportion of mementos over time. **a** More embedded resources are being loaded via JavaScript over time. **b** On average, resources are increasingly being loaded by JavaScript per year

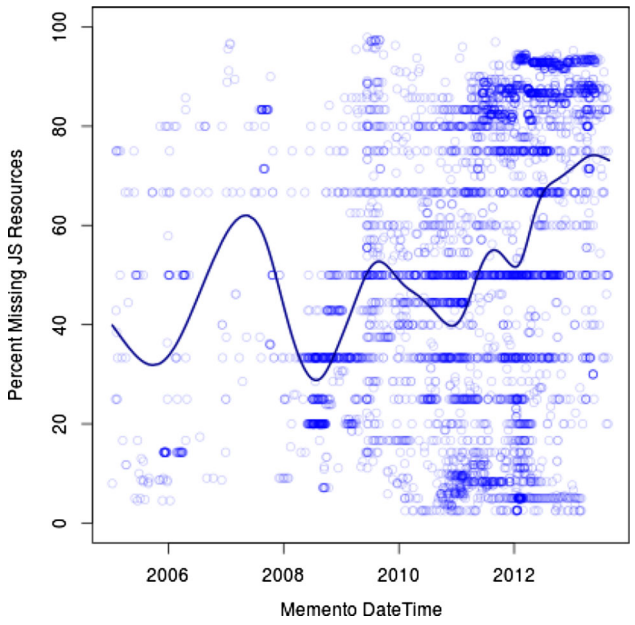
10 The impact of JavaScript on memento completeness

JavaScript is increasing in prevalence over time. We observed the embedded resources loaded from each memento in our collection from 2005–2012 to determine how many resources are being loaded from JavaScript. As shown in Fig. 19, more resources are using JavaScript to load embedded resources over time. In 2005, 39.7 % of our collection uses JavaScript to load at least one embedded resource, and continues to increase to 54.5 % of the collection using JavaScript to load at least one embedded resource. That is an increase of 14.7 % of resources using JavaScript between 2005 and 2012.

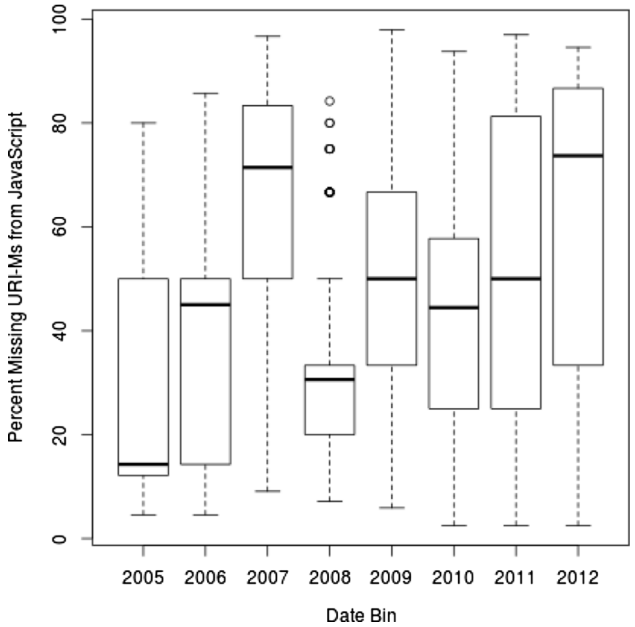
These resources are not only increasingly using JavaScript, but also are more heavily relying on JavaScript to load embedded resources. In Fig. 20a, we plot the percent of requests that come from JavaScript from each URI-M in blue as well as a fitted curve in black. Mementos load 24.5 % of all embedded resources through JavaScript in 2005, and 36.5 % in 2012 for an increase of 12.0 % in 7 years. As shown in Fig. 20b, yearly averages also increase from 2005 to 2012. Over our seven year window, mementos load 37.7 % of all resources via JavaScript ($\sigma = 23.6\%$).

These increases show that resources are loading a higher proportion of their embedded resources by JavaScript instead of HTML (potentially to provide a more personalized experience to the users). These embedded resources are loaded at run-time by JavaScript instead of during the dereferencing process when loaded by HTML. Embedded resources loaded by JavaScript are harder for Heritrix to archive (since they may not be known during crawl-time), and potentially result in missing content when accessing the mementos from the Wayback Machine.

As we discussed in Sect. 8.1, not all resources are loaded properly when archived. JavaScript is responsible for an increasing proportion of missing embedded resources



(a) JavaScript is responsible for an increasing proportion of missing embedded resources over time.



(b) Mementos are binned by archive year to provide a yearly summary of archive performance.

Fig. 21 JavaScript is responsible for an increasing proportion of missing resources. **a** JavaScript is responsible for an increasing proportion of missing embedded resources over time. **b** Mementos are binned by archive year to provide a yearly summary of archive performance

(Fig. 21b). We observed that JavaScript accounts for 39.9 % of all missing embedded resources in 2005, but is responsible for 73.1 % of all embedded resources in 2012. A more detailed analysis is provided in the box plots of Fig. 21b with the missing resources binned according the memento archive year. Box plots provide the calculated 25th percentile as the bottom of the box, the 50th percentile as the horizontal line within the box, and the 75th percentile as the top line of the box. The "whiskers" establish the lower and upper bounds of the data. The dots (for example, those in the box plot of 2008 in Fig. 21b) in a box plot are outliers in the data.

Table 4 Number of requests per memento by archive year

Request type	2005	2006	2007	2008	2009	2010	2011	2012
JavaScript misses	1.7	2.3	2.1	2.3	3.3	3.5	5.8	4.9
HTML misses	0.9	1.9	1.3	1.5	1.5	2.3	3.0	2.3
All HTTP 200s	19.4	21.7	30.8	45.1	31.1	30.6	29.8	38.1

JavaScript is responsible for 52.7 % of all missing embedded resources from 2005 to 2012 ($\sigma = 28.9$) which is 33.2 % more missing resources in 2012 than in 2005, showing JavaScript is responsible for an increasing proportion of the embedded resources unsuccessfully loaded by mementos. JavaScript is increasingly requesting mementos that are not in the archives and is responsible for attempting to load over half of all missing resources in our collection. This suggests that further increasing utilization of mementos to load embedded resources via JavaScript, and further increasing failures to dereference those resources will result in further reduction in memento completeness.

Over time, mementos are increasingly constructed with embedded resources and are requesting and missing an increasing number of embedded resources to render a final representation. Table 4 and Fig. 22 provide the number of successfully dereferenced URI-Ms, unsuccessful dereferences originating in HTML, and unsuccessful dereferences originating from JavaScript for each memento for each year.

While it is uninteresting that more total attempted dereferences result in more successful and more failed dereferences, the breakdown of how the failed dereferences were loaded provides insight into the source of reduced memento completeness. As the number of requested resources increases, the number of resources unsuccessfully loaded by HTML increases, and the number of unsuccessfully loaded JavaScript resources increases, as well. However, the JavaScript requested resources increase at a higher rate ($\Delta_{\text{HTML}} = 1.4$ requests vs $\Delta_{\text{JavaScript}} = 3.2$ requests from 2005 to 2012). This suggests that resources will continue to unsuccessfully load embedded resources via JavaScript as time progresses.

The collections are not dominated by perfect (0 % missing resources from JavaScript) mementos or completely JavaScript reliant (100 % missing resources from JavaScript) as shown in Fig. 23. The 0 and 100 % mementos are approximately equal to each other for every year, meaning the sta-

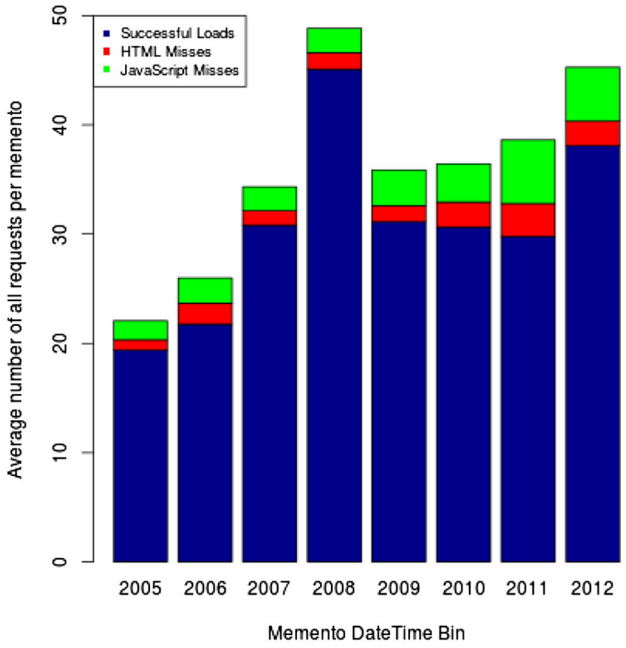


Fig. 22 Number of requests per memento by archive year

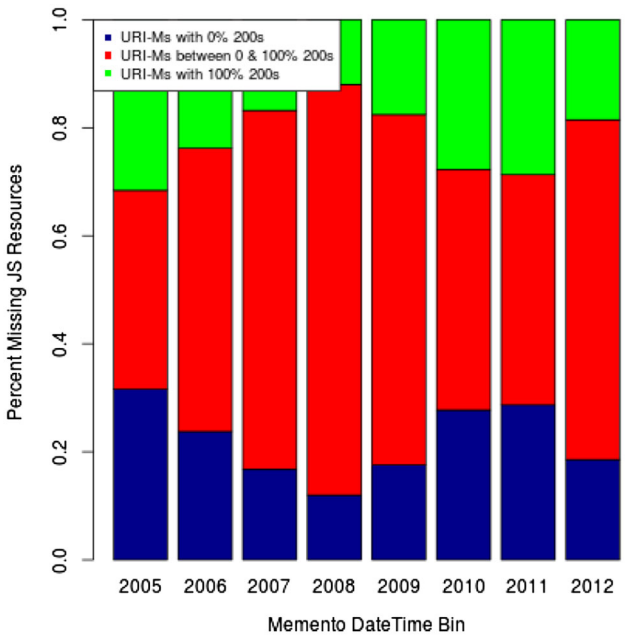


Fig. 23 Percent of missing resources from JavaScript by year

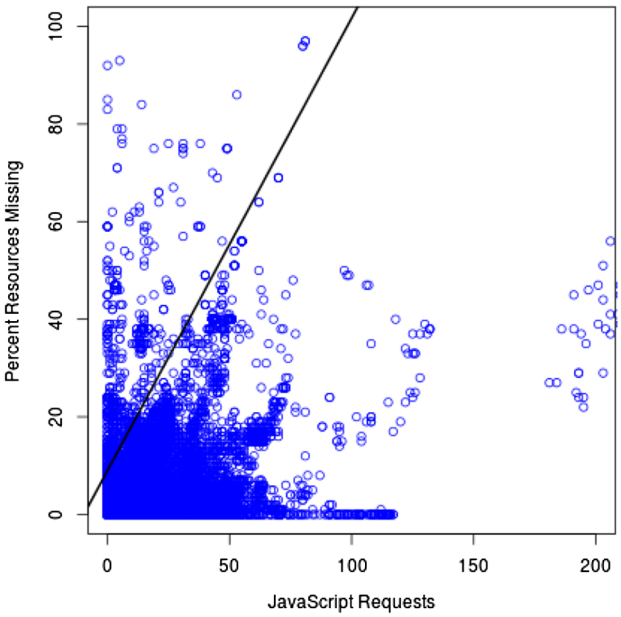


Fig. 24 As JavaScript is relied on more heavily to load embedded resources, more resources are missed

tistics reflect the performance of those resources that load at least (but limited to) a portion of their embedded resources from JavaScript.

As we have hypothesized previously, an increasing number of requests originating from JavaScript will result in an increased proportion of missing embedded resources. We show the number of requests from JavaScript and the resulting proportion of missing mementos for each URI-M in our collection in Fig. 24. The fitted line in black shows the correlation between the two statistics. As the number of requests from JavaScript increases, the number of missing embedded resources increases. This trend is supported by a moderate Kendall Tau-b correlation with $\tau = 0.36$ and $p < 0.01$. This correlation suggests that the current trend of relying increasingly on JavaScript to load embedded resources will result in a higher proportion of missing embedded resources and reduced memento completeness.

We began this contribution with the hypothesis that increasing reliance on JavaScript results in an increase in missing embedded resources in our archives. From the graphs in Figs. 19, 20, 21, 22, 23, 24 we made four findings:

- More resources are using JavaScript over time, and those resources are more heavily relied on by mementos to load embedded resources as time progresses;
- JavaScript is responsible for an increasing proportion of missing resources in the archives.
- The number of missing resources loaded by JavaScript is increasing at a higher rate than the number of missing resources loaded by HTML over time;
- The proportion of missing embedded resources is moderately correlated to the number of resources loaded from JavaScript.

These findings support our hypothesis and suggest that increased JavaScript in the archives will result in decreased memento completeness. As time progresses, an increasing number of resources will rely more heavily on JavaScript,

resulting in more missed content in the archives. Based on these findings, we recommend that crawlers adapt to handle embedded JavaScript to increase the completeness of mementos. A two-tiered crawling approach should be studied further, as recommended in Sect. 4.5.

11 Conclusions

This work observed a set of URIs shared over Twitter and a set of URIs previously archived by Archive-It. The Twitter URIs are twice as complex as the Archive-It URIs; the Archive-It URIs are closer to top-level domains ($\overline{\text{UC}}_{\text{Archive-It}} = 0.166$, $\overline{\text{UC}}_{\text{Twitter}} = 0.377$). The Twitter content contains twice as much JavaScript as the Archive-It set ($\overline{\text{CC}}_{\text{Archive-It}} = 2.16$, $\overline{\text{CC}}_{\text{Twitter}} = 4.78$). This shows that the resources humans actively curate and archive are easier to capture with today's tools than URI-Rs being shared over Twitter. The personalized, shared resources are more difficult to archive and experience much more leakage (between 0.6 % if using Heritrix to 5.8 % more if using wget) than their Archive-It counterparts. Only 4.2 % of the Twitter collection was perfectly archived by each archival tool in our experiment, while 34.2 % of the Archive-It set was perfectly archived by each tool.

The archivability of websites is changing over time because of an increasing reliance on JavaScript to load resources. JavaScript is responsible for 33.2 % more missing resources in 2012 than in 2005 meaning JavaScript is responsible for an increasing proportion of the embedded resources unsuccessfully loaded by mementos. Javascript is also responsible for 52.7 % of all missing content in our collection. This trend is expected to increase as time progresses since the number of embedded resources loaded via JavaScript is moderately correlated to the proportion of missing content in mementos. This supports our theory that as resources continue to more heavily utilize and rely on JavaScript to load resources, the completeness of mementos will decrease.

Acknowledgments This work was supported in part by the NSF (IIS 1009392) and the Library of Congress.

References

1. Access Board: The Rehabilitation Act Amendments (Section 508). http://www.access-board.gov/sec508/guide/act.htm (1998)
2. Ainsworth, S., Alsum, A., SalahEldeen, H., Weigle, M.C., Nelson, M.L.: How much of the Web is archived? In: Proceedings of the 2011 IEEE/ACM Joint Conference on Digital Libraries (JCDL), pp. 133–136 (2011). doi:10.1145/1998076.1998100
3. Antoniades, D., Polakis, I., Kontaxis, G., Athanasopoulos, E., Ioannidis, S., Markatos, E.P., Karagiannis, T.: we.b: the web of short URLs. In: Proceedings of the 20th International Conference on World Wide Web, WWW '11, pp. 715–724 (2011). doi:10.1145/1963405.1963505
4. Archive.today: Archive.today. http://archive.today/ (2013). http://archive.today/
5. Ast, P., Kapfenberger, M., Hauswiesner, S.: Crawler Approaches And Technology. (online). Graz University of Technology, Styria, Austria (2008). http://www.iicm.tugraz.at/cguetl/courses/isr/uearchive/uews2008/Ue01
6. Banos, V., Yunhyong, K., Ross, S., Manolopoulos, Y.: CLEAR: a credible method to evaluate website archivability. In: Proceedings of the 9th International Conference on Preservation of Digital Objects (2013)
7. Benjamin, K., von Bochmann, G., Dincturk, M., Jourdan, G.V., Onut, I.: A strategy for efficient crawling of rich internet applications. In: Proceedings of Web Engineering, Lecture Notes in Computer Science, vol. 6757, pp. 74–89. Springer, Berlin (2011). doi:10.1007/978-3-642-22233-7_6
8. Benson, E., Marcus, A., Karger, D., Madden, S.: Sync kit: a persistent client-side database caching toolkit for data intensive websites. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 121–130 (2010). doi:10.1145/1772690.1772704
9. Bergman, M.K.: Deep web: Surfacing hidden value. J. Electron. Publ. 7(1) (2001). doi:10.3998/3336451.0007.104
10. Berners-Lee, T.: Information management: a proposal. http://www.w3.org/History/1989/proposal.html (1990)
11. Bragg, M., Rollason-Cass, S.: Archiving Social Networking Sites w/ Archive-It. https://webarchive.jira.com/wiki/pages/viewpage.action?pageId=3113092 (2014)
12. Brunelle, J.F.: Google and JavaScript. http://ws-dl.blogspot.com/2014/06/2014-06-18-google-and-javascript.html (2014)
13. Brunelle, J.F., Kelly, M., SalahEldeen, H., Weigle, M.C., Nelson, M.L.: Not all mementos are created equal: measuring the impact of missing resources. In: Proceedings of the 2014 IEEE/ACM Joint Conference on Digital Libraries (JCDL), pp. 321–330 (2014). doi:10.1109/JCDL.2014.6970187
14. Brunelle, J.F., Kelly, M., SalahEldeen, H., Weigle, M.C., Nelson, M.L.: Not all mementos are created equal: measuring the impact of missing resources. Int. J. Digit. Libr. (2014) (accepted for publication)
15. Brunelle, J.F., Nelson, M.L.: Zombies in the archives. http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html (2012)
16. Chakrabarti, S., Srivastava, S., Subramanyam, M., Tiwari, M.: Memex: A browsing assistant for collaborative archiving and mining of surf trails. In: Proceedings of the 26th VLDB Conference, 26th VLDB (2000)
17. Chisholm, W., Vanderheiden, G., Jacobs, I.: Web content accessibility guidelines 1.0. Interactions **8**(4), 35–54 (2001). doi:10.1145/379537.379550
18. Crook, E.: Web archiving in a Web 2.0 world. In: Proceedings of the Australian Library and Information Association Biennial Conference, pp. 1–9 (2008)
19. Davis, R.C.: Five tips for designing preservable websites. http://blog.photography.si.edu/2011/08/02/five-tips-for-designing-preservable-websites/ (2011)
20. Dincturk, M.E., Jourdan, G.V., Bochmann, G.V., Onut, I.V.: A model-based approach for crawling rich internet applications. ACM Trans. Web **8**(3), 19:1–19:39 (2014). doi:10.1145/2626371
21. Duda, C., Frey, G., Kossmann, D., Zhou, C.: AjaxSearch: crawling, indexing and searching Web 2.0 applications. In: The Proceedings of the Very Large Database Endowment (VLDB) Endowment (PVLDB) **1**, 1440–1443 (2008). doi:10.14778/1454159.1454195
22. Eysenbach, G., Trudel, M.: Going, going, still there: using the WebCite service to permanently archive cited web pages. J. Med. Internet Res. **7**(5) (2005). doi:10.2196/jmir.7.5.e60

23. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: RFC 2616. http://tools.ietf.org/html/rfc2616 (1999)
24. Firefox: Firefox. http://www.mozilla.org/en-US/firefox/new/ (2013)
25. Flanagan, D.: JavaScript: the definitive guide. O'Reilly Media (2001)
26. Fleiss, B.: SEO in the web 2.0 era: the evolution of search engine optimization. http://www.bkv.com/redpapers-media/SEO-in-the-Web-2.0-Era (2007)
27. Fuhrig, L.S.: The Smithsonian: using and archiving Facebook. http://blog.photography.si.edu/2011/05/31/smithsonian-using-and-archiving-facebook/ (2011)
28. Garrett, J., et al.: Ajax: a new approach to web applications. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications (2005)
29. GNU: Introduction to GNU wget. http://www.gnu.org/software/wget/ (2013)
30. Hackett, S., Parmanto, B., Zeng, X.: Accessibility of internet websites through time. In: Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility, (77–78), pp. 32–39 (2003). doi:10.1145/1029014.1028638
31. Jack, P.: Extractorhtml extract-javascript. https://webarchive.jira.com/wiki/display/Heritrix/ExtractorHTML+extract-javascript (2014)
32. Jacobs, I., Walsh, N.: Architecture of the world wide web, vol. 1. In: Proceedings of Technical Report W3C Recommendation 15 December 2004, W3C (2004). http://www.w3.org/TR/webarch/
33. Kelly, M., Brunelle, J.F., Weigle, M.C., Nelson, M.L.: On the change in archivability of websites over time. In: Proceedings of the Third International Conference on Theory and Practice of Digital Libraries, pp. 35–47 (2013). doi:10.1007/978-3-642-40501-3_5
34. Kelly, M., Nelson, M.L., Weigle, M.C.: The archival acid test: evaluating archive performance on advanced HTML and JavaScript. In: Proceedings of the 2014 IEEE/ACM Joint Conference on Digital Libraries (JCDL), pp. 25–28 (2014). doi:10.1109/JCDL.2014.6970146
35. Kenney, A.R., McGovern, N.Y., Botticelli, P., Entlich, R., Lagoze, C., Payette, S.: Preservation risk management for web resources. D-Lib Mag. **8**(1) (2002). doi:10.1045/january2002-kenney
36. Kiciman, E., Livshits, B.: AjaxScope: a platform for remotely monitoring the client-side behavior of web 2.0 applications. In: Proceedings of The 21st ACM Symposium on Operating Systems Principles, SOSP '07 (2007). doi:10.1145/1841909.1841910
37. Vikram, K., Prateek, A., Livshits, B.: Ripley: Automatically securing web 2.0 applications through replicated execution. In: Proceedings of the Conference on Computer and Communications Security (2009)
38. Likarish, P., Jung, E.: A targeted web crawling for building malicious javascript collection. In: Proceedings of the ACM First International Workshop on Data-Intensive Software Management and Mining, DSMM '09, pp. 23–26. ACM, New York (2009). doi:10.1145/1651309.1651317
39. Livshits, B., Guarnieri, S.: Gulfstream: incremental static analysis for streaming JavaScript applications. In: Proceedings of Technical Report MSR-TR-2010-4, Microsoft (2010)
40. Marshall, C.C., Shipman, F.M.: On the institutional archiving of social media. In: Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 1–10 (2012). doi:10.1145/2232817.2232819
41. McCown, F., Brunelle, J.F.: Warrick. http://warrick.cs.odu.edu/ (2013)
42. McCown, F., Diawara, N., Nelson, M.L.: Factors affecting website reconstruction from the web infrastructure. In: JCDL '07: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 39–48 (2007). doi:10.1145/1255175.1255182
43. McCown, F., Marshall, C.C., Nelson, M.L.: Why web sites are lost (and how they're sometimes found). Commun. ACM **52**(11), 141–145 (2009). doi:10.1145/1592761.1592794
44. McGovern, N.Y., Kenney, A.R., Entlich, R., Kehoe, W.R., Buckley, E.: Virtual remote control. D-Lib Mag. **10**(4) (2004). doi:10.1045/april2004-mcgovern
45. Mesbah, A., Bozdag, E., van Deursen, A.: Crawling Ajax by inferring user interface state changes. In: Proceedings of Web Engineering, 2008. ICWE '08. Eighth International Conference, pp. 122–134 (2008). doi:10.1109/ICWE.2008.24
46. Mesbah, A., van Deursen, A.: An architectural style for ajax. In: Proceedings of Software Architecture, Working IEEE/IFIP Conference, pp. 1–9 (2007). doi:10.1109/WICSA.2007.7
47. Mesbah, A., van Deursen, A.: Migrating multi-page web applications to single-page ajax interfaces. In: Proceedings of the 11th European Conference on Software Maintenance and Reengineering, CSMR '07, pp. 181–190. IEEE Computer Society, Washington, DC, USA (2007). doi:10.1109/CSMR.2007.33
48. Mesbah, A., van Deursen, A., Lenselink, S.: Crawling ajax-based web applications through dynamic analysis of user interface state changes. ACM Trans. Web **6**(1), 3:1–3:30 (2012). doi:10.1145/2109205.2109208
49. Meyerovich, L.A., Livshits, B.: Conscript: Specifying and enforcing fine-grained security policies for javascript in the browser. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy, SP '10, pp. 481–496. IEEE Computer Society, Washington, DC, USA (2010). doi:10.1109/SP.2010.36
50. Mickens, J., Elson, J., Howell, J.: Mugshot: deterministic capture and replay for JavaScript applications. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10, pp. 159–173 (2010)
51. Mohr, G., Kimpton, M., Stack, M., Ranitovic, I.: Introduction to Heritrix, an archival quality web crawler. In: Proceedings of the 4th International Web Archiving Workshop (2004)
52. National Archives and Records Administration: NARA code of federal regulations-36 CFR subchapter B: records management. http://www.archives.gov/about/regulations/subchapter/b.html (2011)
53. National Archives and Records Administration: NARA code of federal regulations-36 CFR subchapter B part 1236: Electronic Records Management. http://www.archives.gov/about/regulations/part-1236.html (2011)
54. Negulescu, K.C.: Web Archiving @ the internet archive. Presentation at the 2010 Digital Preservation Partners Meeting (2010). http://www.digitalpreservation.gov/meetings/documents/ndiipp10/NDIIPP072110FinalIA.ppt
55. Nelson, M.L.: 2013–07-09: Archive.is supports memento. http://ws-dl.blogspot.com/2013/07/2013-07-09-archiveis-supports-memento.html (2013)
56. @NesbittBrian: Play framework sample application with JWebUnit and synchronous ajax (2011). http://nesbot.com/2011/10/16/play-framework-sample-app-JWebUnit-synchronous-ajax
57. Parmanto, B., Zeng, X.: Metric for web accessibility evaluation. J. Am. Soc. Inf. Sci. Technol. **56**(13), 1394–1404 (2005). doi:10.1002/asi.20233
58. Pierce, M.E., Fox, G., Yuan, H., Deng, Y.: Cyberinfrastructure and web 2.0. In: Proceedings of High Performance Computing and Grids in Action, pp. 265–287 (2008)
59. Reed, S.: Introduction to Umbra. https://webarchive.jira.com/wiki/display/ARIH/Introduction+to+Umbra (2014)
60. Rosenthal, D.S.H.: Talk on harvesting the future web at IIPC2013. http://blog.dshr.org/2013/04/talk-on-harvesting-future-web-at.html (2013)

61. Rossi, A.: 80 Terabytes of archived web crawl data available for research. http://blog.archive.org/2012/10/26/80-terabytes-of-archived-web-crawl-data-available-for-research/ (2012)
62. SalahEldeen, H.: Carbon dating the web. http://ws-dl.blogspot.com/2013/04/2013-04-19-carbon-dating-web.html (2013)
63. SalahEldeen, H.M., Nelson, M.L.: Losing my revolution: how many resources shared on social media have been lost? In: Proceedings of the Second international conference on Theory and Practice of Digital Libraries, pp. 125–137 (2012). doi:10.1007/978-3-642-33290-6_14
64. SalahEldeen, H.M., Nelson, M.L.: Resurrecting my revolution: using social link neighborhood in bringing context to the disappearing web. In: Proceedings of the Third International Conference on Theory and Practice of Digital Libraries, pp. 333–345 (2013). doi:10.1007/978-3-642-40501-3_34
65. Sigursson, K.: Incremental crawling with Heritrix. In: Proceedings of the 5th International Web Archiving Workshop (2005)
66. Thibodeau, K.: Building the archives of the future: advances in preserving electronic records at the national archives and records administration. D-Lib Mag. 7(2) (2001). doi:10.1045/february2001-thibodeau. http://www.dlib.org/dlib/february01/thibodeau/02thibodeau.html
67. Tofel, B.: 'Wayback' for accessing web archives. In: Proceedings of the 7th International Web Archiving Workshop (2007)
68. Van de Sompel, H., Nelson, M.L., Sanderson, R., Balakireva, L.L., Ainsworth, S., Shankar, H.: Memento: time travel for the web. In: Proceedings of Technical Report, Los Alamos National Laboratory (2009). arXiv:0911.1112
69. W3C staff and working group participants: hash URIs. http://www.w3.org/QA/2011/05/hash_uris.html (2011)
70. Wikipedia: ajax (programming). http://en.wikipedia.org/wiki/Ajax_(programming) (2013)
71. Zucker, D.F.: What does ajax mean for you? Interactions **14**, 10–12 (2007). doi:10.1145/1288515.1288523