On the Change in Archivability of Websites Over Time

Mat Kelly, Justin F. Brunelle, Michael L. Nelson, and Michele C. Weigle
Old Dominion University
Department of Computer Science
Norfolk, Virginia 23529 USA
{mkelly,jbrunelle,mln,mweigle}@cs.odu.edu

ABSTRACT

As web technologies evolve, web archivists work to keep up so that our digital history is preserved. Recent advances in web technologies have introduced client-side executed scripts that load data (Ajax) without a referencable identifier (e.g., URI) and others that require user interaction (e.g., content loading when the page has scrolled). These advances have made automating methods for capturing web pages more difficult. Because of the evolving schemes of publishing web pages along with the progressive capability of web preservation tools, the archivability of pages on the web has varied over time. In this paper we show that the *archivability* of a web page can be deduced from the type of page being archived, which aligns with that page's accessibility in respect to dynamic content. We show concrete examples of when these technologies were introduced by referencing temporally tagged captures (mementos) of pages that have persisted through a long evolution of available technologies. Identifying these reasons for the inability of these web pages to be archived in the past in respect to accessibility serves as a guide for ensuring that content that has longevity is published using good practice methods that make it available for preservation.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries; D.3.3 [**Programming Languages**]: Language Constructs an Constraints

General Terms

Design, Experimentation

Keywords

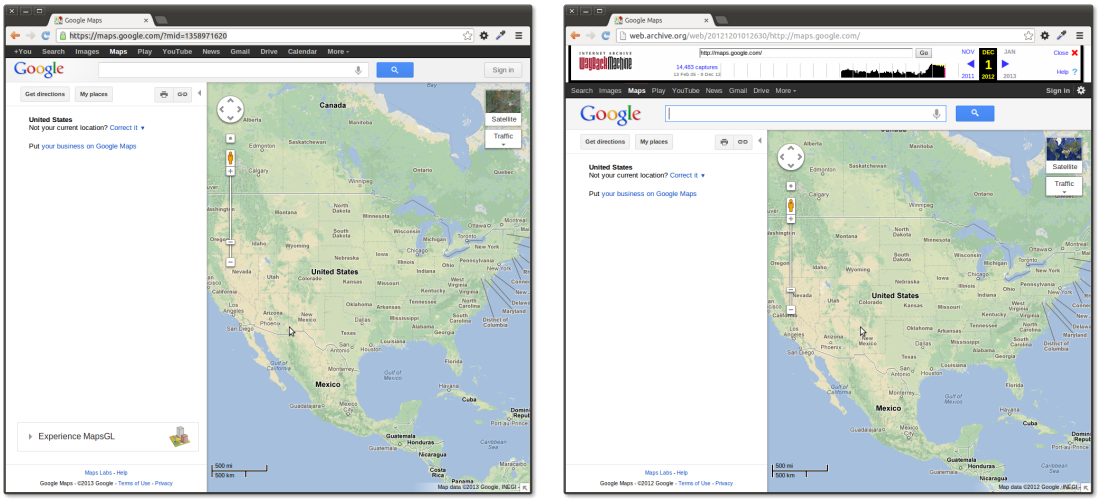
Web Archiving, Digital Preservation, JavaScript, Ajax, Memento

1. INTRODUCTION

The web has gone through a gradient yet demarcated series of phases in which interactivity has become more fluid to the end-user. Early websites were relatively static. Early adoption of JavaScript allowed the components on a web page to respond to users' actions or be manipulated in ways that made the page more (and ironically, often less) usable. A further extension to user interactivity came with the introduction of Ajax [11], which combined multiple web technologies to give web pages the ability to perform operations asynchronously. The adoption of Ajax by web developers facilitated the fluidity of user interaction on the web. Through each phase in the progression of the web, the ability to preserve the content displayed to the user has also progressed but in a less linear trend.

A large amount of the difficulty in web archiving stems from the crawler's insufficient ability to capture content related to JavaScript. Because JavaScript is executed on the client side (i.e., within the browser after the page has loaded), it should follow that the archivability could be evaluated using a consistent replay medium. The medium used to archive (normally a web crawler tailored for archiving, e.g., Heritrix, among others [24, 18, 14]) is frequently different from the medium used to replay the archive (henceforth, the *web browser*, the predominant means of replay). The crawler creates the web archive, which is processed by a replay system (e.g., Internet Archive's Wayback Machine [30]), which is then accessed by an end-user through the web browser medium (e.g., the user accesses Wayback's web interface). This inconsistency between the perspective used to capture the pages versus the perspective used to view the stored content [13] introduces difficulty in evaluating the potential to be archived (henceforth the *archivability*) of web resources. Further discrepancies in the capabilities of the crawler versus the web browser, namely in capturing versus displaying the representation of a resource, make archivability difficult to measure without manual inspection.

The success of preservation of a web page is defined by how much of the originally displayed content is displayed on replay. The success of a consistent replay experience when the original experience contained a large amount of potential user interactivity (and more importantly, the loading of external resources not initially loaded) might not rely on the level of interactivity able to be re-experienced at replay as long as all of the resources to properly display the web page on replay were captured and loaded when the archive



(a) Live version

(b) Archived Dec. 1, 2012

(c) Archived Apr. 30, 2012

Figure 1: Google Maps is a typical example where the lack of completeness of the archive is easy to observe. Figure 1(a) shows how the web page should look. The map in the middle is draggable, allowing the user to pan, among other interactive UI elements. An archived version (from April 30, 2012) of this page Figure 1(c) is missing UI elements (circled), and the interaction does not function. This is due to resources that would be loaded on user interaction, which are not preserved by the crawler. Figure 1(b) shows a recently archived (December 1, 2012) version that gives the façade of functionality when, in fact, resources on the live web are being loaded.

is replayed.

The nature of the execution of the archiving procedure is often to blame for not capturing resources loaded at runtime that not only manipulate the Document Object Model (DOM) but also load subsequent representations of resources. These subsequently loaded representations are often captured [24] if their location is able to be extracted from the static code by a crawler, but a problem occurs when their loading is latent or triggered by user interaction. An example of this can be seen in the Internet Archive's capture of Google Maps[1] (Figure 1). When this archive is replayed, everything that was presented to the user (i.e., the crawler) at time of archiving is displayed. None of the trademark panning or user interface (UI) elements function in a manner similar to the live version (Figure 1(a)) of the same page. The resources, however, appear to be correctly loaded (Figure 1(c)), though the asynchronous JavaScript calls are never fired because of the broken UI elements. Versions of a page recently archived (Figure 1(b)) appear to have all resources required for the page to be interactive, and the page performs as expected, but upon inspection of the URIs, all

[1] `http://maps.google.com`

reference the live web and not the resources at the archive.

Contrasting the completeness of the archive of an interactive website with one from a simpler website that does not contain interactive elements or the loading of external resources via JavaScript further exemplifies that this trait is to blame for archive incompleteness. For example, a single page webpage containing only HTML, images, and CSS is likely to be completely represented in the archive when preserved with an archival tool like Heritrix. Further enforcement of accessibility of all content on a web page as mandated by Section 508 [1], with which all governmental websites are directed to comply, increases the likelihood that complying websites are completely archived. In this paper we will show that the archivability of a website, given the state of the art of archiving tools, has changed over time with the increased usage of resource-loading JavaScript and the increased accessibility of websites. Further, we will examine the incapability of crawlers and archiving tools in capturing this content and what can be done to remedy their shortcomings and increase the archivability of problematic webpages.

2. RELATED WORK

Many components contribute to the archivability of a web page ranging from reliability of the mechanism used to archive to the frequency at which the mechanism is run. Ainsworth et al. utilized web directories like DMOZ, Delicious, Bitly and search engine results to determine how much of the web is archived [4]. McCown et al., in earlier work, developed strategies for resurrecting web pages from the archives but mainly considered those with static resources (including JavaScript) [19]. In more recent work [20], McCown touched on the sources used to recreate lost websites (of particular interest, using Internet Archive's) and the long tail effect on the unlikelihood of domain specific sites to be able to be resurrected using the larger archives as a source. Mohr's much earlier work set the basis for the tool used by Internet Archive to preserve webpages, Heritrix, while introducing incremental crawling into the tool's repertoire of capability [29].

As JavaScript has been the source of many problems in archiving, particularly since the web has become more dynamic, it is useful to note prior attempts relating to JavaScript and archivability. Likarish [16] developed a means of detecting JavaScript with certain facets (namely malicious code via deobfuscation) using Machine Learning techniques and Heritrix. Vikram, Kiciman, and Meyerovich have all collaborated in separate works with Livshits on taking on some of the complexities of JavaScript, including attributes only available at runtime and thus normally limited to be experience by the client [15, 32, 23, 17]. Bergman, in a seminal work, described the quantity of resources we are unable to index (the "deep Web") [7], which has significantly increased with the advent of resources being loaded only at runtime. Ast extended on Bergman's work by proposing approaches to capture Ajax content using a conventional web crawler (i.e., one not used for preservation) [5].

3. WHY JAVASCRIPT MAKES IT DIFFICULT

A user or script normally browses the web using a user agent. In the case of a user, this is normally a web browser. Initially, web browsers were limited in capability and were inconsistent in implementation. This inconsistency eventually was the impetus for creating web standards to remedy the guesswork developers had to do to ensure that the display was as desired. The layout engine is the component of a web browser that is responsible for rendering HTML, the structural portion of a web page. Along with the structure, there is also a stylistic portion (implemented via CSS) and a behavioral portion (implemented in JavaScript) on the client-side.

As the layout engines of modern browsers evolved, the JavaScript rendering engine lagged behind, causing the behavioral functionality of web pages to perform inconsistently among users. This was particularly noticable when Ajax-based websites became common. An example of this inconsistency that requires a different implementation per browser is the need to use the Microsoft.XMLHTTP[2] ActiveXObject for Microsoft Internet Explorer and the standard XMLHTTPRequest[3] object for most other browsers to execute Ajax code.

To simplify the process of obtaining the data quickly and without worry about behavior, many crawlers and scrapers do not include a JavaScript rendering engine. Some, in fact, just grab the HTML and any embedded resources and rely on the user's layout engine to render the fetched data at a later date. This is problematic in that some resources' location on the web might be built at runtime or included in the page because of JavaScript DOM manipulation. In the case of a crawler, this might be negligible, as the resource will still be hot-linked and thus included when the web page is "replayed". For crawlers intended for preservation, however, the archive must be self-contained and thus, for a complete archive, these resources must be captured and their locations rewritten to be accessible at time of replay.

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 of the archive creation process. Recently, Heritrix was rewritten to be built on top of PhantomJS[4], a headless WebKit (the layout engine used by Google Chrome, Apple Safari, etc.) with JavaScript support. This version is currently in production [3] at Internet Archive. This advancement greatly increases the potential for accurate JavaScript processing and the likelihood that all resources required to replay a web page are captured by a crawler.

4. ARCHIVABILITY OF SITES IN RESPECT TO TYPE

We stated in our recent work that web pages of links shared over Twitter were less archivable than those selected to be preserved by the collection-based Archive-It service [8]. Examining the general nature of the web pages on each of these respective realms surfaces an interesting correlation. Many of the websites on Archive-It are governmental. Many of those on Twitter are commercially-based (e.g., CNN.com).

[2]http://msdn.microsoft.com/en-us/library/ms537505(v=vs.85).aspx
[3]http://www.w3.org/TR/XMLHttpRequest/
[4]http://phantomjs.org/

Governmental websites are mandated to conform to web accessibility standards for content. Commercial websites do not have this restriction. From this, it can be extrapolated that the more accessible a website is, the more archivable it is. Further, certain features of JavaScript (e.g., the reliance of it being enabled to show content) may make a page generally less accessible.

The markup (HTML) of a web page is rarely a hindrance to a website being captured. Because the evaluation of the preservation of the markup is reliant on the replay system being used, it is sufficient for the crawler to store the markup without regard to the accuracy its eventual rendering. The difference in interpretation of markup among various users, for instance, does not make the code more or less accessible. Semantic markup, as encouraged by Section 508, WAI specifications, and other organizations that advocate accessible web development practices does affect how end-users see the content. Even if the content displayed is hidden from view, it is still likely present and thus preserved, making variance in markup replay a moot point.

In contrast to markup, behavior (usually JavaScript) can contend what content resides in the markup. If certain behaviors are not invoked, certain content may never make it to the markup, thus compromising the degree at which the content that should be archived is archived.

5. EXPERIMENTAL SETUP

Observing the effects of JavaScript on archivability over time first required a means of fetching archives with a reliable datetime association. The Memento Framework [31] provides exactly this. Using a URI convention and the HTTP Accept-Datetime header, archives can be queried for the approximate time desired and the closest result will be returned by means of the framework implementation. Sites existed prior to archival efforts and though the Internet Archive (the primary agent in putting forth these efforts) has made strides in archiving since 1996 (it now contains over 4.5 petabytes of data [25]), the web is still not comprehensively archived [4].

5.1 Gauging Accessibility

While Section 508 gives suggestions on how websites should comply to be considered accessible, other organizations (namely the World Wide Web Consortium (W3C) through the Web Content Accessibility Guidelines (WCAG)) give concrete ways (14 guidelines that translate into 91 checkpoints) for developers to evaluate their creations to make them more accessible [9]. Hackett et al. go in-depth on the accessibility of archives on a pre-Ajax corpus (from 2002) and enumerate very specific features that make a page accessible in the legal sense. Much of this information is beyond the scope of this study (e.g., animated GIFs) or of little interest to contemporary web archivists (e.g., Java Applets). Other efforts were made following the document's release on a metric to evaluate web accessibility [26], but also fail to consider an asynchronous web, which did not exist to today's degree at the time.

As stated in Section 4, the markup of a web page is not problematic for crawlers to capture. To understand the role that JavaScript plays in hindering comprehensive web archiving by crawlers, it is useful to examine the WCAG Principles of Accessibility [9] and remark on where issues would occur for a crawler. JavaScript, specifically, affects a page's accessibility by hiding information from view (perceivability), only working if all components of the script are present (operability), and frequently playing a critical role in a page being useable (robustness).

```
1: function GetMemsWithYearInterval(mementoURIs)
2:     M ← mementoURIs[1]                    ▷ Get First Memento
3:     lastDate ← extractDate(M)
4:     lastDateTest ← lastDate
5:     for m = 2 → length(mementoURIs) do
6:         testingDate ← extractDate(mementoURIs[m])
7:         testingDate = extractDate(mementoURIs[m])
8:         if lastDate + oneYear ≤ testingDate then
9:             if |lastDate − testingDate + oneYear| ≥
10:                |lastDateTested − lastDate + oneYear|  then
11:                lastDate ← mementoURIs[m − 1]
12:            else
13:                lastDate ← mementoURIs[m]
14:            end if
15:            push(M,mementoURIs[m])
16:        else
17:            lastDateTested ← testingDate
18:        end if
19:    end for
20:    return M
21: end function
```

Figure 2: Pseudocode to get mementos from a timegate at a temporal distance of one year per memento or as close as possible with the first memento as the pivot.

5.2 Fetching Data

The initial experiment was to test the archivability of web sites whose presence in the archive has persisted over a long period of time. These can be described as the "stubby head" juxtaposed to McCown's "long tail" of sites that are preserved. For a long period of time, the website Alexa[5] has gathered the traffic of many of these sites and ranked them in descending order. This ranking currently exists as Alexa's Top 500 Global Sites[6]. Even within the Top 25, websites from Japan[7], China[8],and Russia[9] are present with the majority being based in the United States. We first attempted an approach at gathering data by querying the archives (namely, Internet Archive's Wayback) for past Top lists[10] but found that the location of this list was inconsistent to the present one and some of the sites in past top 10 lists remained present in the current list. We used a simple scraping scheme to grab the paginated list but that turned up pornographic sites by the third page on the 2012 list so we kept it to the top few sites to remain representative, unbiased, and to reduce the likelihood of including sites without longevity.

For each of these web sites, the TimeMap was acquired from

[5] http://www.alexa.com
[6] http://www.alexa.com/topsites
[7] http://yahoo.co.jp
[8] http://sina.com.cn
[9] http://vk.com
[10] http://web.archive.org/web/20090315000000*/http://www.alexa.com/topsites

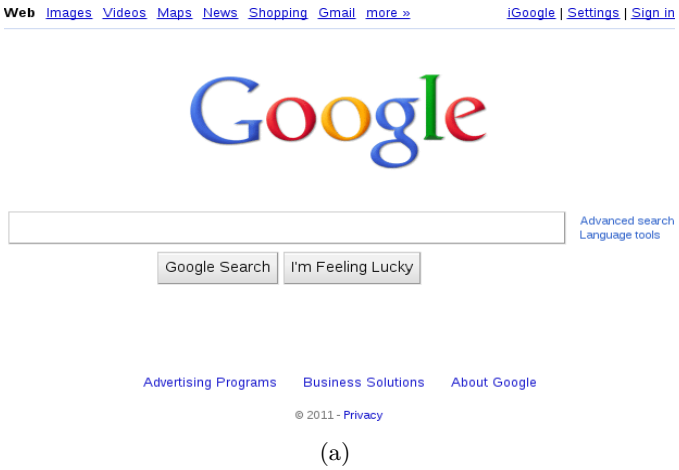
```
<http://api.wayback.archive.org/list/timebundle/http://cnn.com>; rel="timebundle",
<http://cnn.com>; rel="original",
<http://api.wayback.archive.org/list/timemap/link/http://cnn.com>; rel="timemap"; type="application/link-format",
<http://api.wayback.archive.org/list/timegate/http://cnn.com>; rel="timegate",
<http://api.wayback.archive.org/memento/20000620180259/http://cnn.com/>; rel="first memento"; datetime="Tue, 20 Jun 2000 18:02:59 GMT",
<http://api.wayback.archive.org/memento/20000621011731/http://cnn.com/>; rel="memento"; datetime="Wed, 21 Jun 2000 01:17:31 GMT",
<http://api.wayback.archive.org/memento/20000621140928/http://cnn.com/>; rel="memento"; datetime="Wed, 21 Jun 2000 14:09:28 GMT",
...
<http://api.wayback.archive.org/memento/20061227222050/http://www.cnn.com>; rel="memento"; datetime="Wed, 27 Dec 2006 22:20:50 GMT",
<http://api.wayback.archive.org/memento/20061227222134/http://www.cnn.com/>; rel="memento"; datetime="Wed, 27 Dec 2006 22:21:34 GMT",
<http://api.wayback.archive.org/memento/20061228024612/http://www.cnn.com/>; rel="memento"; datetime="Thu, 28 Dec 2006 02:46:12 GMT",
...
<http://api.wayback.archive.org/memento/20121209174923/http://www.cnn.com/>; rel="memento"; datetime="Sun, 09 Dec 2012 17:49:23 GMT",
<http://api.wayback.archive.org/memento/20121209174944/http://www.cnn.com/>; rel="memento"; datetime="Sun, 09 Dec 2012 17:49:44 GMT",
<http://api.wayback.archive.org/memento/20121209201112/http://www.cnn.com/>; rel="last memento"; datetime="Sun, 09 Dec 2012 20:11:12 GMT"
```

Figure 3: A sample abbreviated (for space) TimeMap for cnn.com

Internet Archive using the URI convention "`http://api.wayback.archive.org/list/timemap/link/`*URI*" (where *URI* is the Fully Qualified Domain Name of the target site) to produce output like Figure 3. From this list of mementos we use an algorithm to choose mementos with a one year spread (Figure 2).

The rudimentary algorithm described in Figure 2 fails in instances where there are fewer mementos or where there is a large gap in mementos. In the resulting TimeMap (Figure 4), the URI `http://matkelly.com` is present and returns a TimeMap with mementos going back to 2006. The algorithm chooses the first memento (with Memento-Datetime 20060514123511) as the seed on which to base the annual interval. The next memento retained would have Memento-Datetime 20070514123511, as close as possible to exactly one year. Here, it would have Memento-Datetimes 20060717055501 and 20090505173357. The 2006 date would be chosen and used as the subsequent pivot even though it is only about 2 months from the seed pivot. The previously discarded 2009 date would be assigned the next "annual memento", indicating that this algorithm, though appropriate for web pages with many mementos, is unsuitable for those web sites with few mementos.

The URI of each memento was passed to a PhantomJS script, and the HTTP codes of each resource as well as a snapshot were taken. A snapshot of the memento[11] for `google.com` with the Memento-Datetime 20110731003335, for example, produces the results of Figure 5(a) for the snapshot and a line break delimited list of the subsequent HTTP codes and respective URI dereferenced to assemble the page. Here, we noticed that subsequent requests for the resources yielded resources from the live web. We tailored the PhantomJS script to rewrite the URIs to hit the Wayback Machine instead[12]. This produces an identical display (part of the Wayback UI was programatically hidden) but with resource requests that access archived content (Figure 5(c)). The last step was repeated but with PhantomJS sent the directive to capture the page with JavaScript off.



(a)

```
302 http://api.wayback.archive.org/memento/20110731003335/http://google.com
200 http://api.wayback.archive.org/memento/20110731111142/http://www.google
    .com/
200 http://www.google.com/intl/en_ALL/images/srpr/logo1w.png
404 http://www.google.com/static/js/disclaim.js
200 http://www.google.com/images/srpr/nav_logo80.png
...
```

(b)

```
302 http://web.archive.org/web/20110731003335/http://google.com
200 http://web.archive.org/web/20110731111142/http://www.google.com/
200 http://web.archive.org/static/images/toolbar/wm_tb_prv_on.png
200 http://web.archive.org/static/images/toolbar/wm_tb_nxt_on.png
200 http://web.archive.org/static/js/graph-calc.js
200 http://web.archive.org/static/images/toolbar/wayback-toolbar-logo.png
200 http://web.archive.org/static/js/disclaim-element.js
200 http://web.archive.org/static/jflot/jquery.min.js
200 http://web.archive.org/static/images/toolbar/transp-red-pixel.png
200 http://web.archive.org/static/images/toolbar/transp-yellow-pixel.png
...
302 http://web.archive.org/web/20110731170314im_/http://www.google.com/intl
    /en_ALL/images/srpr/logo1w.png
302 http://web.archive.org/web/20110731111142/http://www.google.com/images/
    srpr/nav_logo80.png
200 http://web.archive.org/liveweb/http://www.google.com/intl/en_ALL/images
    /srpr/logo1w.png
200 http://web.archive.org/liveweb/http://www.google.com/intl/en_ALL/images
    /srpr/logo1w.png
...
200 http://web.archive.org/liveweb/http://www.google.com/images/srpr/
    nav_logo80.png
302 http://web.archive.org/web/20110731111142/http://www.google.com/images/
    srpr/nav_logo80.png
200 http://web.archive.org/liveweb/http://www.google.com/images/srpr/
    nav_logo80.png
200 http://web.archive.org/liveweb/http://www.google.com/images/srpr/
    nav_logo80.png
```

(c)

Figure 5: If all resources are accessible by a memento, fetching that memento from the Wayback API will result in the display of the archived web page. Figure 5(a) shows Google's homepage from memento with Memento-Datetime 20110731003335. The URIs we receive when requesting the TimeMap consists of URIs that are prefixed with the API's URI. When we utilize the API for subsequent resources, these resources are retrieved from the live web (Figure 5(b)). We transform these URIs once we have the timestamps from the TimeMap to access the standard Wayback Machine interface. This produces URI requests that indicate that the resources are being fetched from the live web (Figure 5(c)).

[11] `http://api.wayback.archive.org/memento/20110731003335/http://google.com`

[12] e.g., `http://web.archive.org/web/20110731003335/http://google.com`

```
<http://api.wayback.archive.org/list/timebundle/http://matkelly.com>; rel="timebundle",
<http://matkelly.com>; rel="original",
<http://api.wayback.archive.org/list/timemap/link/http://matkelly.com>; rel="timemap"; type="application/link-format",
<http://api.wayback.archive.org/list/timegate/http://matkelly.com>; rel="timegate",
<http://api.wayback.archive.org/memento/20060514123511/http://www.matkelly.com/>; rel="first memento"; datetime="Sun, 14 May 2006 12:35:11 GMT",
<http://api.wayback.archive.org/memento/20060516213852/http://www.matkelly.com/>; rel="memento"; datetime="Tue, 16 May 2006 21:38:52 GMT",
...
<http://api.wayback.archive.org/memento/20060717055501/http://www.matkelly.com/>; rel="memento"; datetime="Mon, 17 Jul 2006 05:55:01 GMT",
<http://api.wayback.archive.org/memento/20090505173357/http://matkelly.com/>; rel="memento"; datetime="Tue, 05 May 2009 17:33:57 GMT",
<http://api.wayback.archive.org/memento/20090506201837/http://matkelly.com/>; rel="memento"; datetime="Wed, 06 May 2009 20:18:37 GMT",
<http://api.wayback.archive.org/memento/20100114060027/http://matkelly.com/>; rel="memento"; datetime="Thu, 14 Jan 2010 06:00:27 GMT",
<http://api.wayback.archive.org/memento/20100516045729/http://matkelly.com/>; rel="memento"; datetime="Sun, 16 May 2010 04:57:29 GMT",
<http://api.wayback.archive.org/memento/20110208114156/http://matkelly.com/>; rel="memento"; datetime="Tue, 08 Feb 2011 11:41:56 GMT",
<http://api.wayback.archive.org/memento/20110507001028/http://www.matkelly.com/>; rel="memento"; datetime="Sat, 07 May 2011 00:10:28 GMT",
...
<http://api.wayback.archive.org/memento/20121127044548/http://matkelly.com/>; rel="last memento"; datetime="Tue, 27 Nov 2012 04:45:48 GMT"
```

Figure 4: A TimeMap from a website that has fewer mementos. Such a TimeMap is problematic when the algorithm in Figure 2 is applied. In the case of the websites examined here, however, this is moot, as the examined websites contain a sufficient number of mementos without any large temporal gaps.

From the top 10 websites on Alexa for 2012, some websites had a robots.txt restriction. Heritrix, by default, obeys this and does not crawl websites that have a robots exclusion policy [24]. Internet Archive's production Heritrix maintains this setting. The number of mementos obtained by using the code in Figure 2 and applying the URI transformation in Figure 5 produces the following quantity of mementos, ordered corresponding to Alexa's 2012 ranking (Table 1).

Alexa Rank	Web Site Name	Available Mementos
1	Facebook.com	no mementos robots.txt exclusion
2	Google.com	15 mementos 1998 to 2012
3	YouTube.com	7 mementos 2006 to 2012
4	Yahoo.com	16 mementos 1997 to 2012
5	Baidu.com	no mementos robots.txt exclusion
6	Wikipedia.org	12 mementos 2001 to 2012
7	Live.com	15 mementos 1999 to 2012[13]
8	Amazon.com	14 mementos 1999 to 2012
9	QQ.com	15 mementos 1998 to 2012
10	Twitter.com	no mementos robots.txt exclusion

Table 1: Alexa's 2012 Top 10 websites and available mementos.

That the content of some of these websites is not preserved (namely Facebook and Twitter) by institutions has been addressed by multiple parties [21, 14, 2, 6, 22]. As evidenced in Figure 5, all of these websites may not exhibit traits of un-archivability, as previously imagined. The root domain, in this case, may not be representative of the extent at which a web site (contrasted to web page) utilized un-archivable practices. One particular site that has succumbed to the effects of unarchivability due partially to both its longevity and publishing medium is YouTube. Crook [10] went into detail about the issues in preserving multimedia resources on the web, and Prellwitz documented how quickly this multimedia degrades [27], so highlighting this website for analysis would be useful in remedying one of the many reasons that it is not sufficiently preserved.

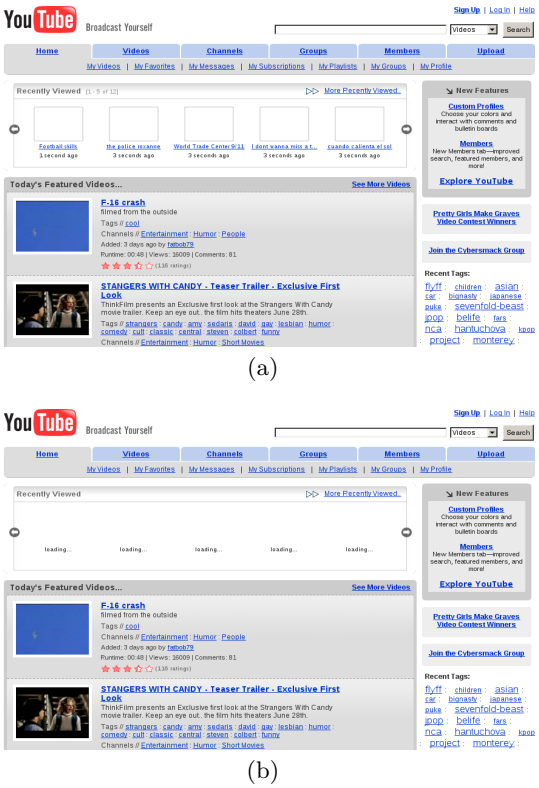


Figure 6: A YouTube memento from 2006 shows a subtle distinction in display when JavaScript is enabled (Figure 6(a)) and disabled (Figure 6(b)) at the time of capture. The Ajax spinner (above each "loading" message in Figure 6(b) is never replaced with content, which would be done were JavaScript enabled on capture. When it was enabled, the script that gathers the resources to display (blank squares in the same section of the site in Figure 6(a)) is unable to fetch the resources it needs in the context of the archive. The URIs of each of these resources (the image source) is present as an attribute of the DOM element but because it is generated post load, the crawler never fetches the resource for preservation.

Using the procedure described earlier in this section, we captured screen shots and HTTP requests for one memento per year of YouTube.com. One of the problematic aspects of YouTube, normally, is retention and playback of the multimedia content. While there have been efforts in attempting to capture the multimedia on this site in a reliable way (e.g., TubeKit[28]), our concern is less about executing a focused crawl and more on analyzing the results of what has been done in the past. The simpler case here of lack of archivability is observable from the homepage. In each of the cases of capturing a screen shot (Figure 6) of the memento with and without JavaScript, there is variance on the "Recently Viewed" section of the website. This part of the website is Ajax-driven, i.e., after the page has loaded, the content is fetched (Figure 7). A crawler could retain the JavaScript that fetches the resources and attempt to grab a copy of the resources contained within and loaded at runtime but this particular script takes a moment post-load to load and display the images that represent links to videos. This is better explained by Figure 6(a), which is representative of the memento with JavaScript enabled. The content necessary to display this section was preserved due to its reliance on runtime execution. Figure 6(b) shows the same memento fetched with JavaScript off. The place-holder Ajax "spinner" demonstrates that the JavaScript to overwrite the DOM elements is present in the archive and executable but the resources needed to fully build this webpage do not exist in the archive.

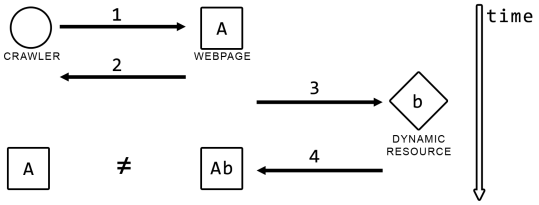
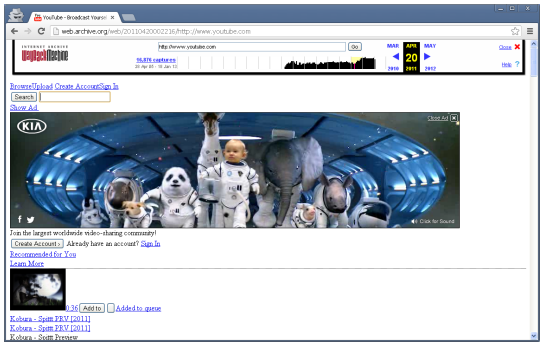


Figure 7: When a crawler meant for web archiving fetches a web page (1), it waits until the page and all of the resources to consider it fully rendered to load. At this point, the crawler preserves (2) all of the content on the web page (A). Meanwhile, after the page has loaded, the page can request 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 full loaded (it triggered on a simpler mechanism), the secondary content is not preserved and the version of the web page preserved is not completely archived.



(a)

```
GET http://web.archive.org/web/20121208145112cs_/http://s.ytimg.com/yt/
      cssbin/www-core-vfl_0JqFG.css 404 (Not Found) www.youtube.com:15
GET http://web.archive.org/web/20121208145115js_/http://s.ytimg.com/yt/
      jsbin/www-core-vfl8PDcRe.js 404 (Not Found) www.youtube.com:45
Uncaught TypeError: Object #<Object> has no method 'setConfig' www.youtube.
      com:56
Uncaught TypeError: Cannot read property 'home' of undefined www.youtube.
      com:76
Uncaught TypeError: Cannot read property 'ajax' of undefined www.youtube.
      com:86
Uncaught TypeError: Cannot read property 'ajax' of undefined www.youtube.
      com:96
Uncaught TypeError: Object #<Object> has no method 'setConfig' www.youtube.
      com:101
Uncaught TypeError: Cannot read property 'home' of undefined www.youtube.
      com:204
Uncaught TypeError: Object #<Object> has no method 'setMsg' www.youtube.com
      :496
Uncaught ReferenceError: _gel is not defined www.youtube.com:1784
Uncaught TypeError: Object #<Object> has no method 'setConfig' www.youtube.
      com:1929
Uncaught TypeError: Object #<Object> has no method 'setMsg' www.youtube.com
      :1938
Uncaught TypeError: Cannot read property 'home' of undefined www.youtube.
      com:524
GET http://web.archive.org/web/20130101024721im_/http://i2.ytimg.com/vi/1
      f7neSzDqvc/default.jpg 404 (Not Found)
```

(b)

Figure 8: The 2011 capture of this YouTube.com memento shows that in addition to the screen shot (Figure 8(a)) indicating that a CSS file was missing, a chain of Ajax-driven resource calls failed (Figure 8(b), thereby preventing all resources from being included in the web page's preservation process.

Contrast this to five years later (2011) when a redesign of YouTube that is heavily reliant on Ajax fails. When loading this memento[14] into Wayback via a web browser, the JavaScript errors in Figure 8(b) appear in the console. This memento (Figure 8(a)) exhibits even more problems in that it contains "Zombies in the Archives" [12], i.e., resources on the live web are displayed embedded within archived pages [8].

The lack of aesthetic of the 2011 YouTube memento is a result of the CSS files (first line of Figure 8(b)) returning an HTTP 302 (redirect) error with the final URI resulting in a 404, as evidence in the log file that accompanies the Figure 8(a) screenshot when the aforementioned annual memento collection process occurs. The unarchivability of this page is not evident with the screenshot alone. By examining the JavaScript log, we noted that a causal chain prevented resource fetching from further down in the execution from completing successfully. JavaScript is fairly resilient to runtime errors and oftentimes will continue executing so long as

[14] http://api.wayback.archive.org/memento/20110420002216/http://youtube.com

a resource dependency is not hit[15]. The progressive increase of Ajax on YouTube over time has caused a longer chain of failures than the 2006 example. Testing this same procedure on a website that persisted from before Ajax existed until today yet chose to rely on it at one time would test whether its inclusion greatly reduce the archivability.

6. A REINFORCING CASE

A second example where the change in archivability over time is much more dramatic can be found in the NASA website[16]. As a government funded agency, NASA is advised to comply with the aforementioned accessibility standards. The same procedure (Section 5) of creating a collection of annual mementos was used to obtain screenshots (Figure 11), HTTP logs and additionally capturing the HTML of the memento. Mementos ranging from 1996-2006 were available and retained, a sampling that sufficiently spanned the introduction of dynamism into the web.

```
<script language="javascript" type="text/javascript" src="flash.js"></
      script>

<script language="javascript" type="text/javascript">
function flashURL(id) {
  //Flash Redirect URL
  window.location.href = 'index.html';
}
</script>
```

(a)

```
var fstr = '';

if(hasFlash(6)) {
        fstr+='<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
               codebase="http://download.macromedia.com/pub/shockwave/cabs/
              flash/swflash.cab#version=6" width="200" height="100" id="
              screenreader.swf"><param name="movie" value="screenreader.swf
              "/><param name="quality" value="high"/><param name="bgcolor"
              value=""/><embed src="screenreader.swf" quality="high"
              bgcolor="#000000" width="200" height="100" name=".swf" type="
              application/x-shockwave-flash" pluginspage="http://www.
              macromedia.com/go/getflashplayer"></embed></object>';
        window.status = 'Flash 6 Detected...';
} else {
        fstr+='<br><br><br><br> <center><font size="2">  <font color="#
              FFFFFF">  <font face="Arial, Helvetica, sans-serif">To view
              the enhanced version of NASA.gov, you must have Flash 6
              installed.</font></font></font></center>';

        fstr+='<br><br><br> <center><font color="#0099FF" size="2" face="
              Arial, Helvetica, sans-serif"><strong>+  </strong
              ></font><strong><font color="#FFFFFF" size="2" face="Arial,
              Helvetica, sans-serif"><a href="http://www.macromedia.com/go/
              getflashplayer/" target="_new">Install Flash 6 now</a>  
              ;         <font color
              ="#0099FF">+  </font><a href="http://www.nasa.gov/
              home/index.html">Enter NASA.gov</a> </font></strong> </center
              >';
}
with(document) { open('text/html'); write(fstr); close(); }
```

Figure 9: In 2003, nasa.gov introduced code into their website that checked the capability of the user's web browser and showed or hid content. A highly discouraged practice that hinders accessibility is to only show content that is available through scripting means. The link to enter the website regardless of the user's browser capability, here, is generated with JavaScript. This would cause the content to not be displayed were the user's browser incapable or if client-side scripting were disabled in the user's browser preferences.

[15]This is by design of the interpreted language but appears to go against the fast-fail software philosophy.

[16]http://www.nasa.gov

The mementos from 1996 through 2002 show table-based websites devoid of JavaScript. In 2003, JavaScript was introduced into the markup (Figure 9(a)). Checkpoint 6.3[17] of the Web Content Accessibility Guidelines [9] states, "Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page." Checkpoint 6.3 is a Priority 1 checkpoint meaning, "A Web content developer must satisfy this checkpoint. Otherwise, one or more groups will find it impossible to access information in the document. Satisfying this checkpoint is a basic requirement for some groups to be able to use Web documents." Here, lack of accessibility directly correlates with unarchivability. Normally, providing an alternate means of viewing the page's content would suffice were the link to "Enter NASA" regardless of the incompatibility, but even the single relevant link on the page (with the other being a link to install Flash) is generated by a script (Figure 9(b)). From the 2004 to 2006 snapshots on, in lieu of testing for Flash, the ability to progress into the site is no longer offered but rather a message stating that JavaScript is required and a means to access instructions to enable it is the sole content supplied to the crawler (and those that had a less capable user agent). Observing the count of the resources required to construct a memento (Figure 10) gives further evidence that things went awry in terms of both accessibility and archivability between 2004 and 2006.

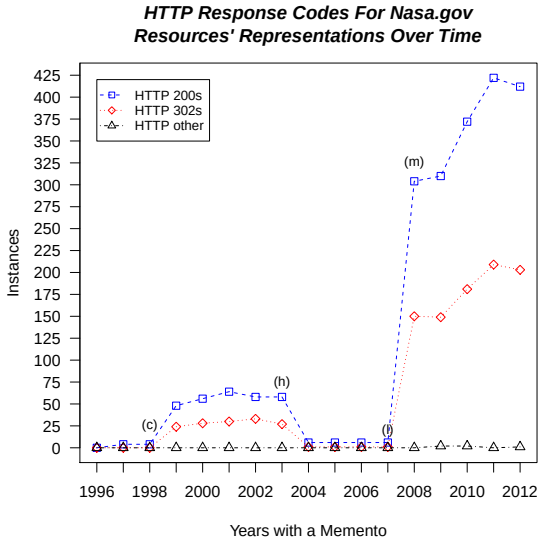


Figure 10: The number of resources required to construct the page has a noticeably absent lull that corresponds to Figure 11.

Relying solely on the number of resources fetched to determine where a site's reliance on unarchivable technologies lies is not foolproof, but it is a good guide to identify problematic pages. Were a crawler to encounter this drastic change and if the changed count was sustained, this should be noted

[17]http://www.w3.org/TR/WCAG10/wai-pageauth.html#tech-scripts

as evidence of potential problems. On a comparable note, the same procedure was run on another government website where this deviation from web standards would be the least likely to surface, `whitehouse.gov`. A similar dip can be seen in Figure 12 in 2010. Examining the screen shot and log of HTTP codes, akin to Figure 5(c), it is evident that a subset of CSS files were not preserved by the crawler. A preserved web page resembling this problem is not one necessarily related to the crawler's inabaility to fetch components of a page embedded in JavaScript but rather, the URI was not persistent enough to endure the time to reach Heritrix's horizon (the point at which it is preserved) once placed on the frontier (list of URIs to be preserved).

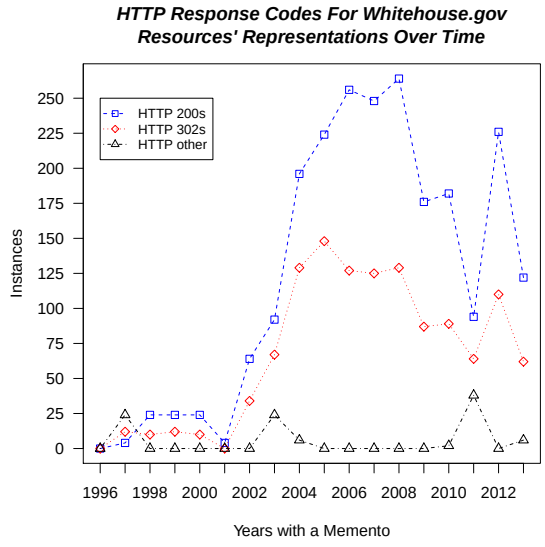


Figure 12: The preservation of the White House web page exhibits a different problem yet is briefly similar in that the count drastically changed. The sudden change in 2011 is the result of a set of CSS files not reaching the crawler horizon, which may have had implications on subsequent resource representations (embedded within the CSS) from being preserved.

7. CONCLUSIONS

The archivability of websites has changed over time in different ways for different classes of websites. While JavaScript is partially to blame for this, it is more a problem that content is not accessible. Lack of accessibility makes content more difficult for crawlers to capture. Websites that are trend leaders, unfortunately, set a bad premise for facilitating archivability. As this trend will likely continue, tools are being created (Heritrix 3) that are instead becoming more capable at working with inaccessible websites. Recognizing techniques to make the archiving process easier by those that want their content preserved is a first step in guiding web development practices into producing web sites that are easier to preserve.

8. REFERENCES

[1] The Rehabilitation Act Amendments (Section 508). `http://www.access-board.gov/sec508/guide/act.htm`, 1998.

[2] Twitter Donates Entire Tweet Archive to Library of Congress. `http://www.loc.gov/today/pr/2010/10-081.html`, 2010.

[3] 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.

[4] S. G. Ainsworth, A. Alsum, H. SalahEldeen, M. C. Weigle, and M. L. Nelson. How Much of the Web is Archived? In *Proceeding of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, JCDL '11, pages 133–136, New York, NY, USA, 2011. ACM.

[5] P. Ast, M. Kapfenberger, and S. Hauswiesner. Crawler Approaches And Technology. *[Online]. Graz University of Technology, Styria, Austria*, 2008. `http://www.iicm.tugraz.at/cguetl/courses/isr/uearchive/uews2008/Ue01%20-%20Crawler-Approaches-And-Technology.pdf`.

[6] J. Bass. Getting Personal: Confronting the Challenges of Archiving Personal Records in the Digital Age. Master's thesis, University of Winnipeg, 2012.

[7] M. Bergman. White Paper: the Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, 7(1), 2001.

[8] J. F. Brunelle, M. Kelly, M. C. Weigle, and M. L. Nelson. Losing the Moment: The Unarchivability of Shared Links. *Submitted for Publication.*

[9] W. Chisholm, G. Vanderheiden, and I. Jacobs. Web Content Accessibility Guidelines 1.0. *Interactions*, 8(4):35–54, July 2001.

[10] E. Crook. Web Archiving in a Web 2.0 World. *Electronic Library, The*, 27(5):831–836, 2009.

[11] J. Garrett et al. Ajax: A New Approach to Web Applications. `http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications`, 2005.

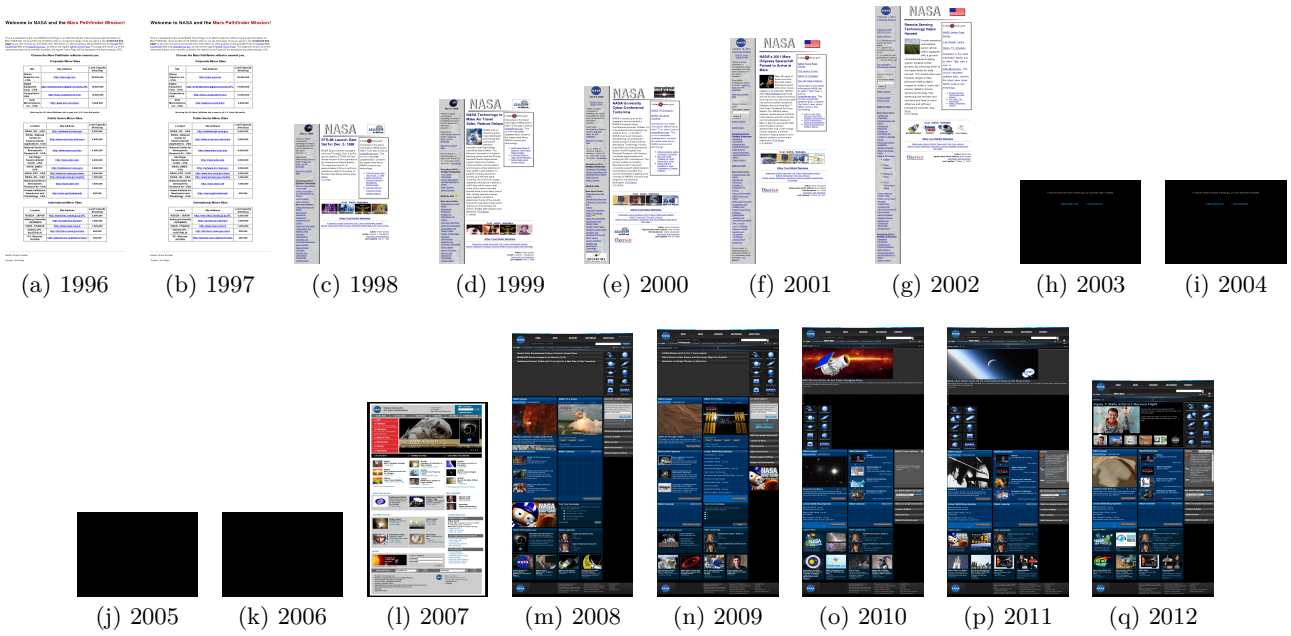
[12] Justin F. Brunelle. Zombies in the Archives. `http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html`, 2012.

[13] M. Kelly. An Extensible Framework For Creating Personal Archives Of Web Resources Requiring Authentication. Master's thesis, Old Dominion University, 2012.

[14] M. Kelly and M. C. Weigle. WARCreate - Create Wayback-Consumable WARC Files from Any Webpage. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, pages 437–438, Washington, DC, June 2012.

[15] E. Kıcıman and B. Livshits. AjaxScope: A Platform for Remotely Monitoring the Client-Side Behavior of Web 2.0 Applications. In *Proceedings of Symposium on Operating Systems Principles*, 2007.

[16] P. Likarish and E. Jung. 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, pages 23–26, New York, NY, USA, 2009. ACM.



(a) 1996 (b) 1997 (c) 1998 (d) 1999 (e) 2000 (f) 2001 (g) 2002 (h) 2003 (i) 2004

(j) 2005 (k) 2006 (l) 2007 (m) 2008 (n) 2009 (o) 2010 (p) 2011 (q) 2012

Figure 11: NASA over time. Changes in design and thus the technologies used is easily observable between Figures 11(b) and 11(c), 11(g) and 11(h), 11(k) and 11(k), and 11(l) and 11(m)

[17] B. Livshits and S. Guarnieri. Gulfstream: Incremental Static Analysis for Streaming JavaScript Applications. Technical Report MSR-TR-2010-4, Microsoft, January 2010.

[18] J. Masanès. Web Archiving Methods and Approaches: A Comparative Study. *Library Trends*, 54(1):72–90, 2005.

[19] F. McCown, N. Diawara, and M. L. Nelson. Factors Affecting Website Reconstruction from the Web Infrastructure. In *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, pages 39–48, 2007.

[20] F. McCown, C. C. Marshall, and M. L. Nelson. Why Websites Are Lost (and How They're Sometimes Found). *Communications of the ACM*, 52(11):141–145, 2009.

[21] F. McCown and M. L. Nelson. What Happens When Facebook is Gone? In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, pages 251–254, New York, NY, USA, 2009. ACM.

[22] E. Meyer. Researcher Engagement with Web Archives-Challenges and Opportunities. Technical report, University of Oxford, 2010.

[23] L. Meyerovich and B. Livshits. Conscript: Specifying and Enforcing Fine-Grained Security Policies for Javascript in the Browser. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 481–496. IEEE, 2010.

[24] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to Heritrix, an Archival Quality Web Crawler. In *Proceedings of the 4th International Web Archiving Workshop (IWAW'04)*, September 2004.

[25] K. C. Negulescu. Web Archiving @ the Internet Archive. Presentation at the 2010 Digital Preservation Partners Meeting, `http://www.digitalpreservation.gov/meetings/documents/ndiipp10/NDIIPP072110FinalIA.ppt`, 2010.

[26] B. Parmanto and X. Zeng. Metric for Web Accessibility Evaluation. *Journal of the American Society for Information Science and Technology*, 56(13):1394–1404, 2005.

[27] M. Prellwitz and M. L. Nelson. Music Video Redundancy and Half-Life in YouTube. In *TPDL*, pages 143–150, 2011.

[28] C. Shah. Tubekit: a Query-based YouTube Crawling Toolkit. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 433–433. ACM, 2008.

[29] K. Sigurðsson. Incremental Crawling with Heritrix. In *Proceedings of the 5th International Web Archiving Workshop (IWAW'05)*, 2005.

[30] B. Tofel. 'Wayback' for Accessing Web Archives. In *Proceedings of the 7th International Web Archiving Workshop (IWAW'07)*, 2007.

[31] H. Van de Sompel, M. L. Nelson, R. Sanderson, L. L. Balakireva, S. Ainsworth, and H. Shankar. Memento: Time Travel for the Web. Technical Report arXiv:0911.1112, 2009.

[32] K. Vikram, A. Prateek, and B. Livshits. Ripley: Automatically Securing Web 2.0 Applications Through Replicated Execution. In *Proceedings of the 16th ACM conference on Computer and Communications Security*, pages 173–186. ACM, 2009.