



Server-Side Web Archiving

Version 1.0 | April 2021

Eoin O'Donohoe (Netherlands Institute for Sound and Vision)



**dutch digital
heritage
network**

Contents

1. Introduction	3
1.1 Background	3
1.2 Server-Side	4
1.3 The Dynamic Web	4
1.4 Past Research	4
1.5 Methodology	4
1.6 Goal	5
1.7 Scope	5
2. Tools	6
2.1 Conifer	6
2.2 Rezip	6
3. Use Cases	7
3.1 Last Hijack	7
3.2 Collapsus	12
4. Application of Tools	16
5. Results	22
6. Recommendations and Conclusions	24
7. Glossary	26
Credits	30

1. Introduction

This report explores the area of server-side web preservation in an archival context by examining the significant properties of the dynamic web, providing an overview of the tools currently available for capturing such projects, and offering some examples and use cases that can display where server-side archiving can be employed as a strategy for heritage institutions.

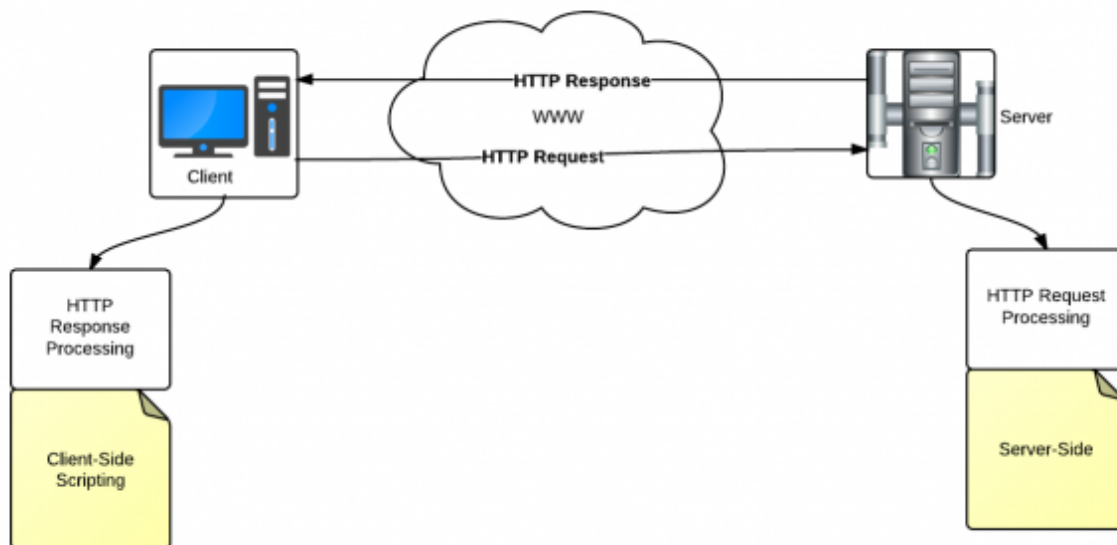
1.1 Background

The NDE Software Archiving project, conducted during the “intensiveringsperiode” 2019–2020, brings together research, best practices, and guidelines for Dutch heritage institutions looking to start with or intensify software preservation. Alongside traditional software packages, this research also aims to encompass web technologies and web based applications that don't neatly fit into regular web crawling workflows. Dynamic web content and interactive experiences online are ever increasing and the processes involved are essential for the successful preservation of such projects. There are several initiatives in place that focus on capturing and safeguarding the web, whether by creating snapshots of specific websites at particular times or through the recording of individual, subjective walkthroughs of a user's website interactions. Yet there is further scope for institutions to identify, document and preserve the collective assets that combine to make up such projects. The Internet Archive's *Wayback Machine* and Rhizome's *Conifer* (previously *WebRecorder*) offer invaluable tools for the capture of and access to client-side content but even those don't always succeed in capturing a fully functioning interactive website. Through focusing efforts on preserving the server-side files, rather than crawling the client-side representation of a website, some of the pitfalls of crawling tools can be avoided by offering a more holistic method. Acquiring and preserving server-side data for complex websites is not a straightforward task due to a reliance on project makers and website hosters to provide access to files and the diverse nature of the websites in question. However, and what to do with such data in order to make it more accessible is still being investigated. This report will answer some of these questions and offer recommendations on getting started with preservation of dynamic web content. This will hopefully be beneficial for institutions and individuals looking to make the first steps in this area.

1.2 Server-side

Many websites depend on server-side operations to retrieve, manipulate and store data that is requested or provided by the user on the client side. Server-side content can include all kinds of assets such as scripts, media, databases that are managed by the website creators. Many of the operations performed on the server end would be too slow and insecure to achieve on the client-side. Transaction-based and server-side approaches require active collaboration with the server owners and need to be implemented on a case-by-case basis.¹

¹ <https://nationalarchives.gov.uk/documents/information-management/web-archiving-guidance.pdf> pg. 5



Client side and server side communication

1.3 The Dynamic Web

Websites can be divided into two types; static websites and dynamic websites. Static websites offer all users the same content and same experience by displaying information using HTML and CSS. Dynamic websites on the other hand, involve a dialogue between the user, via their browser, and the website server, where data is stored. This transactionary nature creates a unique experience for each user through deferred representations, whereby client-side technologies such as JavaScript are used to change the client-side state of a representation after it has been initially loaded.² Dynamic websites can be difficult for a traditional web crawler to capture due to the scripts and databases which are encapsulated on the server-side. The option to recreate the original environment of the website would ensure the preservation of the performative nature of such projects.

1.4 Past Research

Previous research into this topic has been conducted by Sound and Vision colleague Rasa Bocyte in their 2018 paper *Server-side Preservation of Dynamic Websites*³. A lot of groundwork has been laid out here in terms of the needs of server-side preservation as well as the tools that can be used to achieve them. By testing these tools in practice on a few use cases, and providing more information on how to deploy these tools this report brings server side archiving to a new audience.

Work on the ReproZip Web tool, as a prototype, has been presented at iPres 2019⁴. In this research we looked at the degree to which this tool is now ready for use in archival practice.

1.5 Methodology

The goal of this study is to provide knowledge and recommendations on the steps required for Dutch heritage institutions as they set out to collect and preserve complex websites, such as interactive documentaries. Specifically, it focuses on the available use cases from the

² Brunelle et al (2015). Archiving Deferred Representations Using a Two-Tiered Crawling Approach, iPres 2015. <https://arxiv.org/pdf/1508.02315.pdf> pg 1

³ Bocyte, R. (2018). Server-side Preservation of Dynamic Websites. Netherlands Institute for Sound and Vision. <https://publications.beeldengeluid.nl/pub/633>

⁴ Boss et al (2019), Saving Data Journalism Using ReproZip-Web to Capture Dynamic Websites for Future Reuse, iPres 2019. <https://ipres2019.org/static/proceedings/iPRES2019.pdf>

Submarine studio and aims to distill some best practices for getting started. The overview of available tools should present some options for what can be done with acquired projects.

1.6 Goal

For the purpose of this report, digital preservation analyst Eoin O'Donohoe of the Netherlands Institute for Sound and Vision (Sound and Vision) carried out the research, installation, testing, and evaluation of several tools for the preservation of dynamic websites. Focussing on particular examples, provided by Amsterdam based production studio Submarine, this research identifies commonalities between significant properties, suitability of available tools, and necessary collaboration for the successful preservation of such projects. The work explores the current state of the web based productions in the Netherlands and the most feasible methods available for their preservation. The process consisted of:

1. Familiarisation with tools and technologies available for the long-term preservation of web based projects.
2. Research into the individual case studies from Submarine including technologies used, identification of server files, local redeployment of websites, troubleshooting with website creators.
3. Installing/deploying the dynamic website (case study) on a local machine. Then packaging with reprozip-web and re-installing it from the reprozip package.
4. Testing of Reprozip tool and analysis of where it fails and where it succeeds in capturing the use-cases.

1.7 Scope

The scope of this report is to investigate the specific tooling for capturing dynamic websites, with the main focus being on a server-side approach. The examination of these tools can be seen as a use case in its own right and will serve to highlight the potential barriers when it comes to planning for the preservation of such assets. Outside the scope of this report is the establishment of a concrete procedure for capturing all dynamic websites, as each example will likely present it's own unique issues and challenges.

2. Tools

For the purposes of this report, a variety of tools were investigated for their suitability. Two of the more notable options that we wanted to use for capturing and packaging our use cases were Conifer and a reproducibility tool called Rezip.



Tools used as part of testing

2.1 Conifer

By its own description, Conifer (previously known as WebRecorder) offers the possibility of creating high fidelity, interactive captures of the web from the perspective of a specific user, while also offering a platform where these captures can be made accessible and played back. Developed by Rhizome, the tool allows users to capture complex client-side interactions such as embedded media, 3D graphics and fancy navigation and save them as a collection. Using a dedicated UI, users can load up the URL of a website and begin the capture process, simply navigating through the experience. This is highly subjective and thorough interrogation of each site is necessary to achieve a true picture. Nevertheless, the ease of use and generally high quality of the captures produced make Conifer a valuable alternative to automated crawling tools.

2.2 Rezip

Rezip is a reproducibility tool used to collect all necessary data, scripts, libraries and environment variables and package them into one self-contained bundle. The idea is that these bundles can then be redeployed on different machines and different environments but still work as originally intended. Rezip has been used to capture complex scientific data and processes as scripts are run but also allows for the capture of client-server applications (including databases). By running a web application locally from the command line it should be possible to trace the processes that are happening and automatically identify which files should be included in the final package. Having access to server files for a particular project and knowledge of its deployment is an important first step for using this tool.

3. Use Cases

For our two use cases we contacted Amsterdam based production studio Submarine, who have previously worked with Sound and Vision. The company is known for creating award winning features, animations, documentaries and digital content. Most notably they have produced several interactive and transmedia experiences for the web. Each project makes use of various web technologies to tell a story, and it is down to each user to navigate their own way through the information presented. In this study we decided to look at two specific example; *Last Hijack* and *Collapsus*

3.1 Last Hijack

Last Hijack is an interactive documentary that tells the story of piracy in Somalia. Through recorded interviews, animations, and data-visualisations the user can navigate through a timeline of events with new information and resources becoming available at different points.



Last Hijack by Submarine

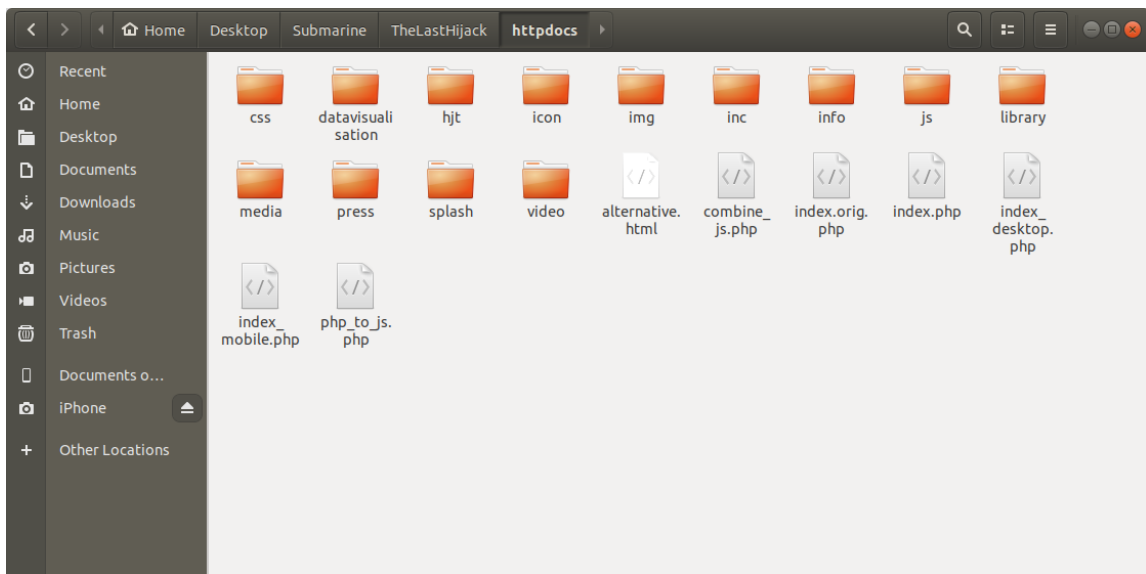
Initial Appraisal

Last Hijack is primarily written in PHP⁵ and features an interactive timeline with various content attached at various points. These take the form of embedded videos, text pop ups, data-visualisations and interactive maps all of which make extensive use of JavaScript in order to render. Further investigation into the folders of the server files show use of JQuery⁶. The total project folder received from Submarine consists of over five thousand files, separated into various folders. Many of these files are interconnected scripts and from the perspective of a non

⁵ PHP is a general-purpose scripting language especially suited to web development, usually processed on the server side.

⁶ JQuery is a JavaScript library used to simplify a variety of scripting operations including event handling, animation and HTML manipulation.

professional developer it is difficult to understand what is going on so it is beneficial to have someone knowledgeable in this area on hand to assist.



Screenshot of project files

Deployment

In order to see if the website would render correctly it was necessary to deploy the files on a server. As the project was written in PHP primarily it was decided to make use of the PHP built-in server which is generally used for local testing of web applications in development. I used an Ubuntu machine running version 18.04.4. Firstly, PHP was installed via the command line.

```
beng@beng:~$ sudo apt install php7.2-cli
```

```
beng@beng: ~
File Edit View Search Terminal Help
beng@beng:~$ php -v
Command 'php' not found, but can be installed with:
sudo apt install php7.2-cli
sudo apt install hhvm
beng@beng:~$ sudo apt install php7.2-cli
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
 libegl1-mesa libblvm8 libwayland-egl1-mesa linux-modules-5.3.0-26-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
 php-common php7.2-common php7.2-json php7.2-opcache php7.2-readline
Suggested packages:
 php-pear
The following NEW packages will be installed:
 php-common php7.2-cli php7.2-common php7.2-json php7.2-opcache
 php7.2-readline
0 upgraded, 6 newly installed, 0 to remove and 142 not upgraded.
Need to get 2505 kB of archives.
After this operation, 12,3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Example command line installation of PHP


```

beng@beng: ~
File Edit View Search Terminal Help
Setting up php-common (1:60ubuntu1) ...
Created symlink /etc/systemd/system/timers.target.wants/phpsessionclean.timer → /lib/systemd/system/phpsessionclean.timer.
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up php7.2-common (7.2.24-0ubuntu0.18.04.6) ...

Creating config file /etc/php/7.2/mods-available/calendar.ini with new version
Creating config file /etc/php/7.2/mods-available/ctype.ini with new version
Creating config file /etc/php/7.2/mods-available/exif.ini with new version
Creating config file /etc/php/7.2/mods-available/fileinfo.ini with new version
Creating config file /etc/php/7.2/mods-available/ftp.ini with new version
Creating config file /etc/php/7.2/mods-available/gettext.ini with new version
Creating config file /etc/php/7.2/mods-available/iconv.ini with new version
Creating config file /etc/php/7.2/mods-available/pdo.ini with new version
Creating config file /etc/php/7.2/mods-available/phar.ini with new version
Creating config file /etc/php/7.2/mods-available/posix.ini with new version
Creating config file /etc/php/7.2/mods-available/shmop.ini with new version
Creating config file /etc/php/7.2/mods-available/sockets.ini with new version
Creating config file /etc/php/7.2/mods-available/sysvmsg.ini with new version
Creating config file /etc/php/7.2/mods-available/sysvsem.ini with new version
Creating config file /etc/php/7.2/mods-available/sysvshm.ini with new version
Creating config file /etc/php/7.2/mods-available/tokenizer.ini with new version
Setting up php7.2-readline (7.2.24-0ubuntu0.18.04.6) ...

Creating config file /etc/php/7.2/mods-available/readline.ini with new version
Setting up php7.2-json (7.2.24-0ubuntu0.18.04.6) ...

Creating config file /etc/php/7.2/mods-available/json.ini with new version
Setting up php7.2-opcache (7.2.24-0ubuntu0.18.04.6) ...

Creating config file /etc/php/7.2/mods-available/opcache.ini with new version
Setting up php7.2-cli (7.2.24-0ubuntu0.18.04.6) ...
update-alternatives: using /usr/bin/php7.2 to provide /usr/bin/php (php) in auto mode
update-alternatives: using /usr/bin/phar7.2 to provide /usr/bin/phar (phar) in auto mode
update-alternatives: using /usr/bin/phar.phar7.2 to provide /usr/bin/phar.phar (phar.phar) in auto mode

Creating config file /etc/php/7.2/cli/php.ini with new version
beng@beng:~$

```

Example command line installation of PHP

This ensured that the built-in server was available for use. The following step was to identify where the access point to the website was. This is generally the index.html file or something similar. With many variations of this naming convention, across multiple different file extensions, it was a bit of trial and error to find the correct starting point but once identified the following command was run via the command line:

```

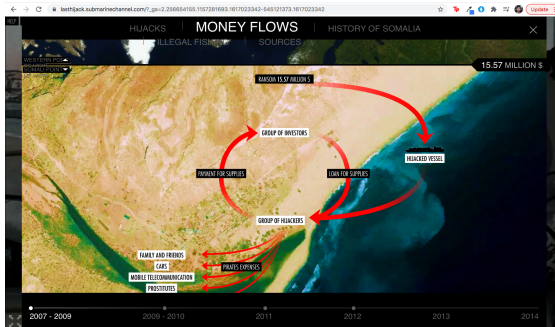
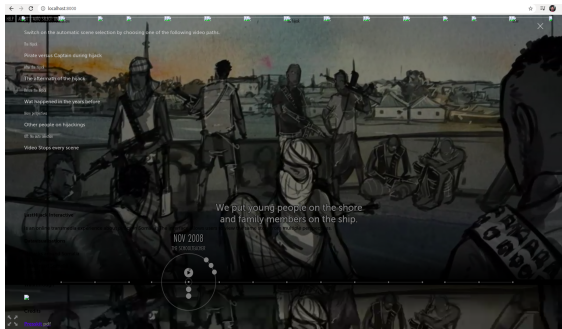
beng@beng: ~/Desktop/Submarine/TheLastHijack/httpdocs
File Edit View Search Terminal Help
beng@beng:~$ cd Desktop/Submarine/TheLastHijack/httpdocs/
beng@beng:~/Desktop/Submarine/TheLastHijack/httpdocs$ php -S localhost:8000
PHP 7.2.24-0ubuntu0.18.04.7 Development Server started at Tue Nov 3 15:30:31 2020
Listening on http://localhost:8000
Document root is /home/beng/Desktop/Submarine/TheLastHijack/httpdocs
Press Ctrl-C to quit.

```

Example command line deployment of project

This deployed the website on my machine’s local host, port 8000, where I could access it via a web browser. At first glance everything appeared to render correctly, with videos and text appearing correctly in the browsers. Navigation back and forth through the different options wasn’t a problem but when it came to some of the data-visualisation elements it was clear that there was a missing link somewhere. For the most part the folder structure of the project was clear and labelled and “Data Visualisations” was its own folder. On further inspection, and after reviewing the error messages in the console it became clear that the scripts were calling from a slightly different path. Whether this was a result of a mix up when the server files were

transferred, or a complication between relative and absolute path conventions⁷, is unsure but by tracing the folder location that was actually being called and moving the data visualisation files there the issue was partly resolved.



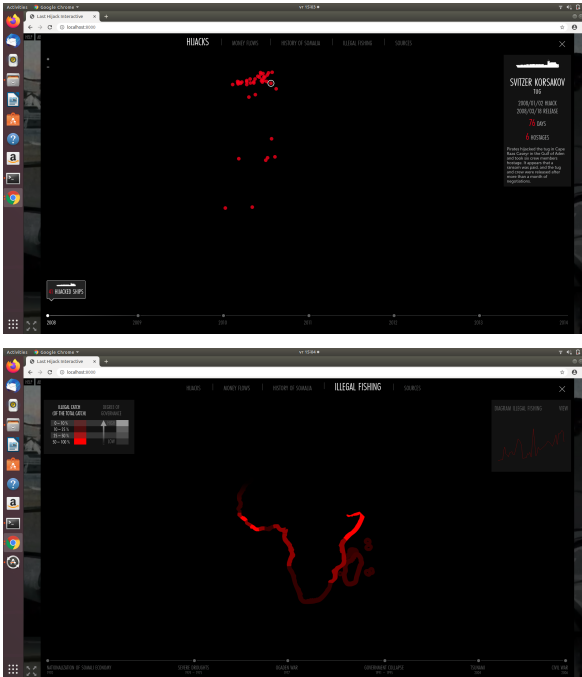
Data visualisation overlay with missing resources beside live version

```
[Mon Feb 1 18:07:17 2021] 127.0.0.1:46768 [404]: /datavisualisation/scripts/main.js - No such file or directory
[Mon Feb 1 18:07:26 2021] 127.0.0.1:46806 [200]: /common/styles/tooltipster.css
[Mon Feb 1 18:07:26 2021] 127.0.0.1:46810 [200]: /common/styles/main.css
[Mon Feb 1 18:07:50 2021] PHP Notice: Undefined property: stdClass::$dummy in /home/beng/Desktop/Submarine/TheLastHijack/httpdocs/money-flows/index.php on line 126
[Mon Feb 1 18:07:50 2021] PHP Notice: Undefined property: stdClass::$dummy in /home/beng/Desktop/Submarine/TheLastHijack/httpdocs/money-flows/index.php on line 126
[Mon Feb 1 18:07:50 2021] PHP Notice: Undefined property: stdClass::$dummy in /home/beng/Desktop/Submarine/TheLastHijack/httpdocs/money-flows/index.php on line 126
[Mon Feb 1 18:07:50 2021] PHP Notice: Undefined property: stdClass::$dummy in /home/beng/Desktop/Submarine/TheLastHijack/httpdocs/money-flows/index.php on line 126
```

Terminal displaying error message 404 where resources were missing

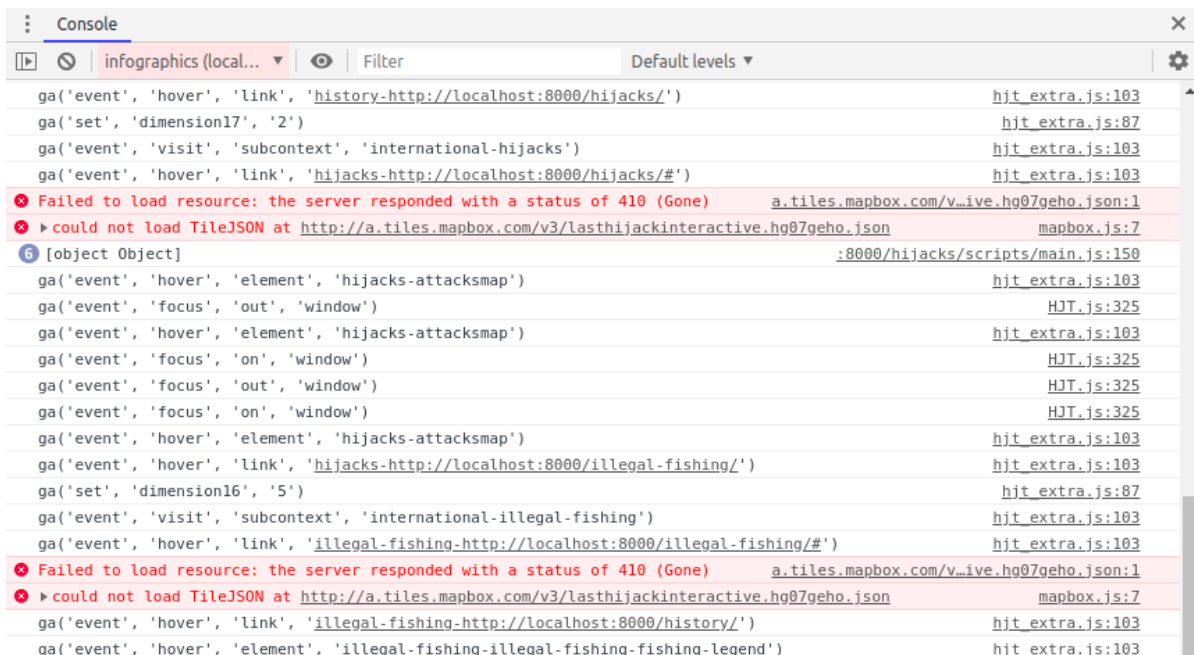
The remaining issue related to the website's use of an external map resource, which is meant to be incorporated into some of the data visualisations. When comparing the local deployment and the live version it became evident that both were experiencing the same issue.

⁷ A file is identified by its path through the file system, beginning from the root node. A path is either *relative* or *absolute*. An absolute path always contains the root element and the complete directory list required to locate the file whereas a relative path might just contain subdirectory information.



Maps not displaying on certain pages (This is also an issue with the live version)

When the console was consulted it displayed that the website was attempting to access a Mapbox⁸ resource which is responsible for importing the live map for use with the data visualisations. Discussing the issue with developers from Submarine it was mentioned that a possible cause for this problem was an outdated subscription to Mapbox. As this affects the live version it was taken as a priority to fix by the developers but it does further highlight the preservation issue of depending on external resources.



Console pointing to Mapbox issues

⁸ Mapbox is a developer's subscription platform that provides custom live maps for applications. <https://www.mapbox.com/>

3.2 Collapsus

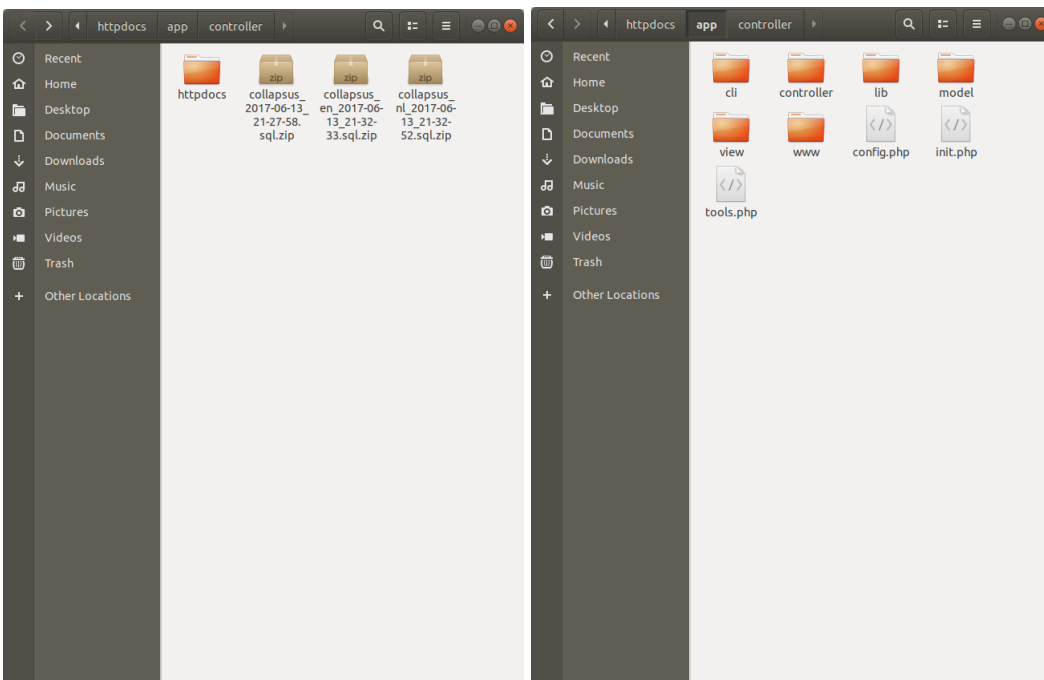
Collapsus is a transmedia storytelling experience that deals with issues surrounding global energy politics. It makes use of documentary footage, mini games and movie clips to put users in control of the unfolding story.

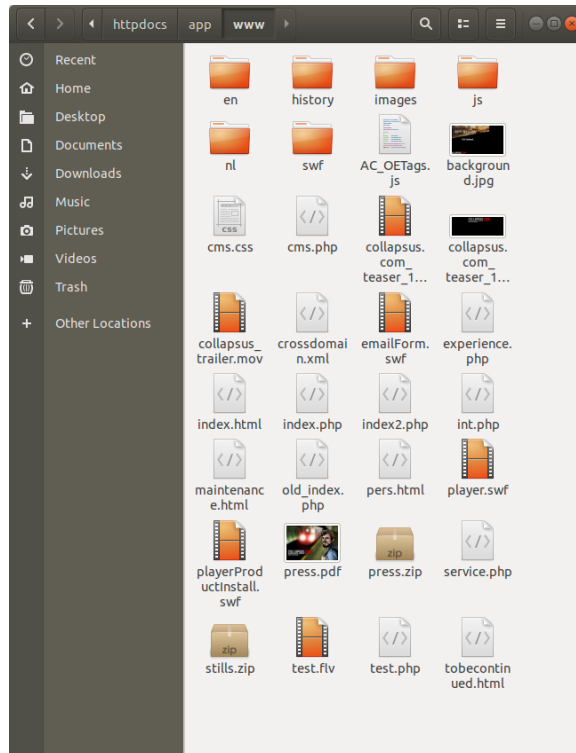


Collapsus by Submarine

Initial Appraisal

Collapsus, like *The Last Hijack*, is written in PHP but also includes an underlying SQL database and a collection of flash video that feeds the application. Once again, the interconnecting files sprawl across many folders with little in the way of documentation or instruction for deployment.





Folder structure of *Collapsus* source files

Deployment

Having located the “index.php” file in the “/www” folder this seemed like the natural option to point the built in PHP server towards. In a similar method to *The Last Hijack* example I navigated to the appropriate folder and launched the server on port 8000 using the same command as before in the terminal:

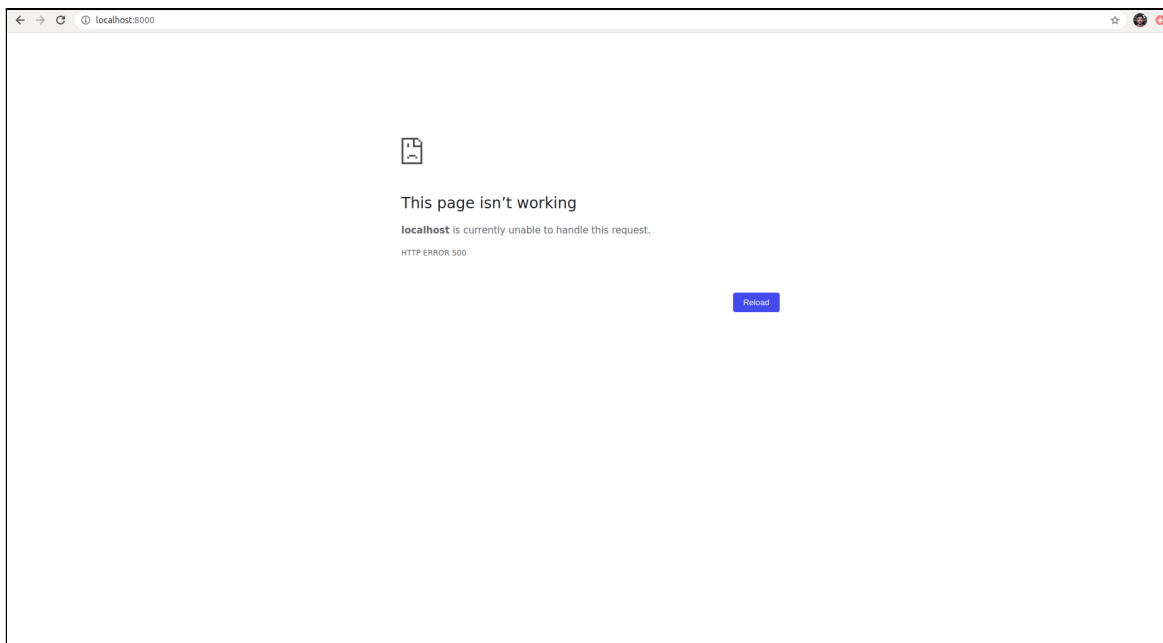
```
beng@beng:~/Desktop/collapsus/httpdocs/app/www$ php -S localhost:8000
```

With the server started I opened the browser to localhost:8000 but was presented with “HTTP Error 500” with the page failing to load. Returning to the terminal the error message displayed an issue with a `mysql_connect()` function in the “init.php” file.

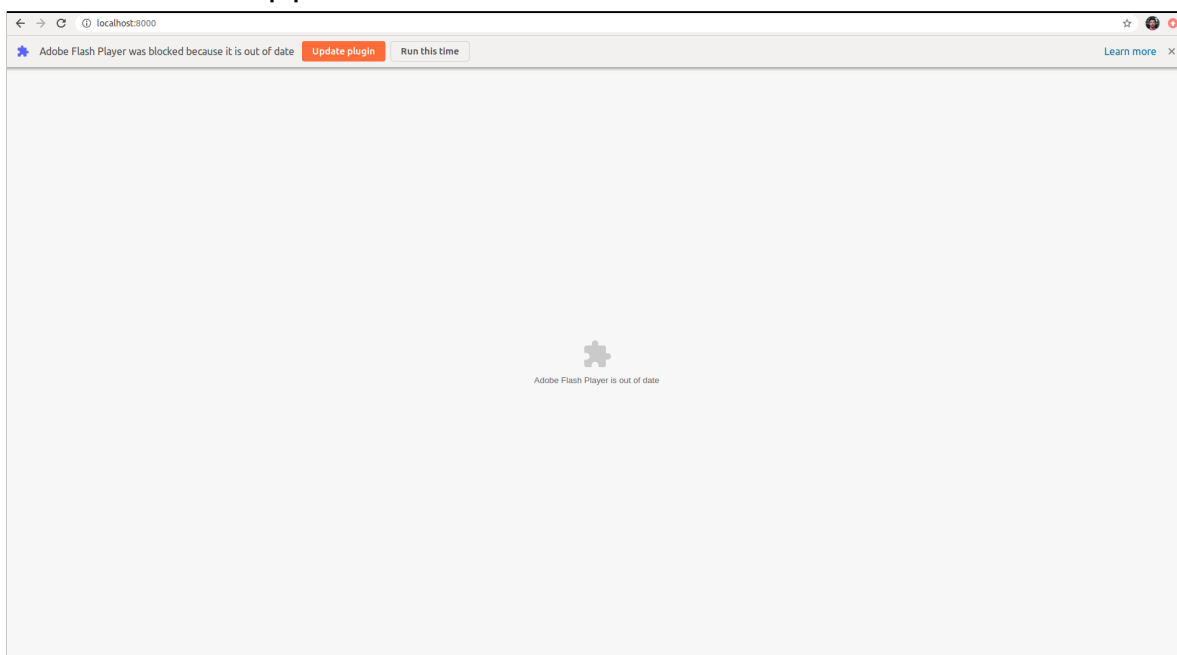
```
[Sun Mar 7 16:34:36 2021] PHP Fatal error: Uncaught Error: Call to undefined function mysql_connect() in /home/beng/Desktop/collapsus/httpdocs/app/init.php:7
Stack trace:
#0 /home/beng/Desktop/collapsus/httpdocs/app/www/index.php(3): require()
#1 {main}
   thrown in /home/beng/Desktop/collapsus/httpdocs/app/init.php on line 7
[Sun Mar 7 16:34:36 2021] 127.0.0.1:42698 [500]: / - Uncaught Error: Call to undefined function mysql_connect() in /home/beng/Desktop/collapsus/httpdocs/app/init.php:7
Stack trace:
#0 /home/beng/Desktop/collapsus/httpdocs/app/www/index.php(3): require()
#1 {main}
   thrown in /home/beng/Desktop/collapsus/httpdocs/app/init.php on line 7
```

Error message when trying to connect to *Collapsus* underlying database

When raised with the developers at Submarine it was determined that this was as a result of an outdated MySQL driver. A relatively quick fix, which resulted in an updated “init.php” file, this again highlights the potential for legacy issues affecting the long term sustainability of web applications. After replacing the “init.php” file and relaunching the server some progress was evident but the next issue presented itself.



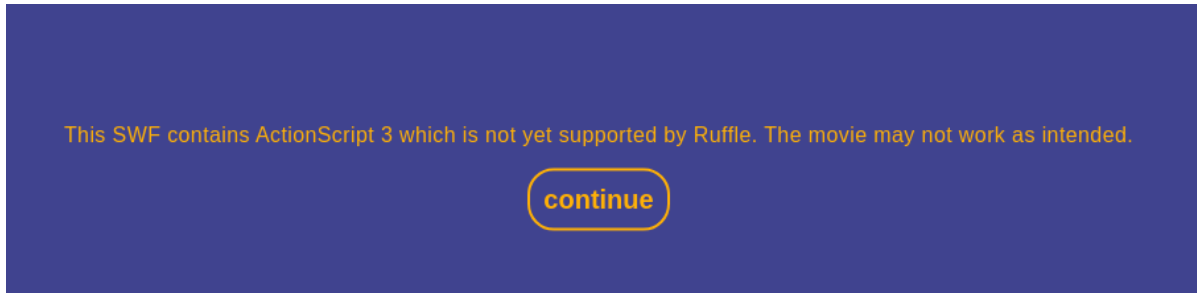
HTTP Error 500 before “init.php” fix



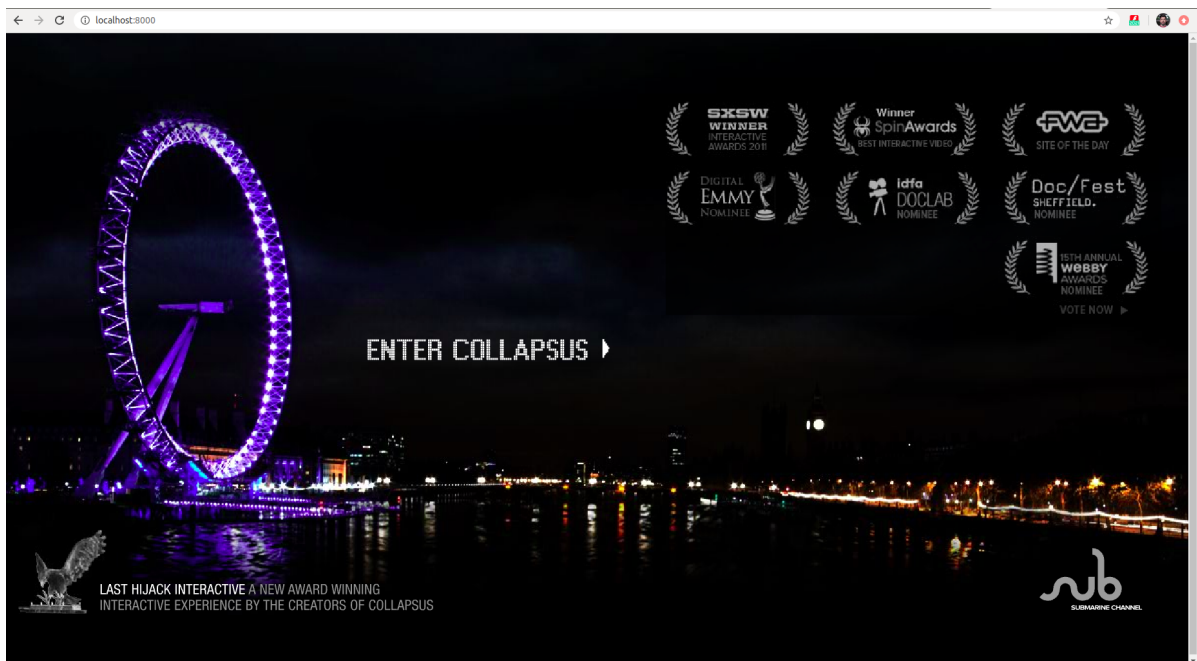
Prompt to enable Flash after “init.php” fix

This issue is probably the most difficult to deal with for any web application that relies on Flash post 2020 as the much publicised discontinuation of Flash support makes browser access to such sites impossible. This problem was discussed with the developers and the possibility of sourcing an old Flash plugin was suggested. This in turn would require an older version of a browser that is compatible with Flash. Due to time constraints, and the added layer of complexity, this approach was not tested in this study but work has been conducted in a

separate report on the topic of browser emulation⁹ which could aid in the development of an appropriate browser environment to run Flash reliant websites. As a final attempt to access the content I added the Flash Player for Web (update 2021) extension to chrome. This player aims to extend the use of some Flash content after 2020 by making use of the Ruffle Flash emulator¹⁰ to access content.



Ruffle warning when trying to access *Collapsus* Flash content



Home page rendered using Ruffle extension but lacks functionality when clicking on Enter

⁹ Claudia Roeck. (2021, January 28). Web browser characterisation, emulation, and preservation (Version 1.0). Zenodo. <http://doi.org/10.5281/zenodo.4476030>

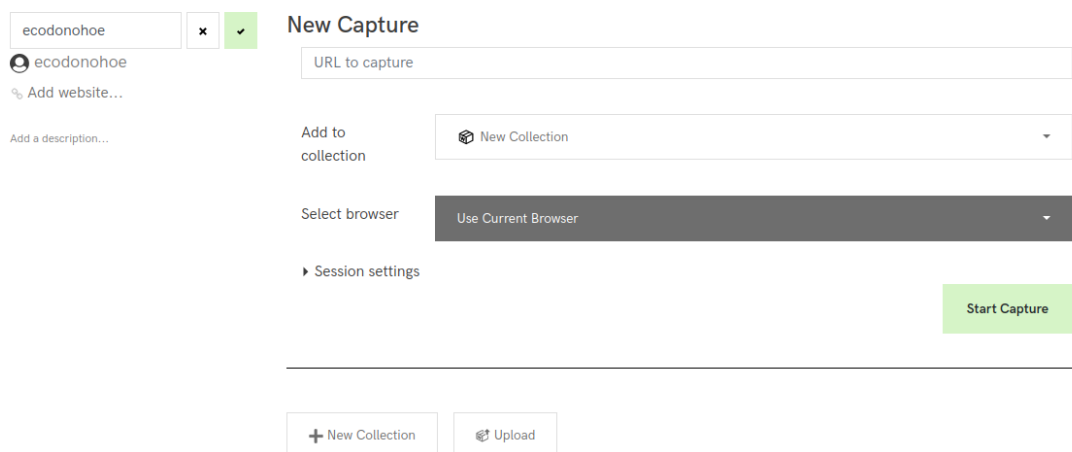
¹⁰ Ruffle is a Flash Player emulator written in Rust that runs natively on all modern operating systems as a standalone application, and on all modern browsers. <https://ruffle.rs/#>

4. Application of Tools

This section will describe the experiences experimenting and interacting with two specific tools: Conifer (previously Webrecorder) and Rezip.

Conifer

The Conifer web application is straightforward and intuitive offering the user the possibility to create an account where website captures can be saved in collections. For the purpose of this test I tried capturing *Last Hijack*, which relies heavily on user interaction to expose all possible content available. The below interface shows the starting point for capturing a Conifer recording. The user simply needs to include the target URL and click “Start Capture”.



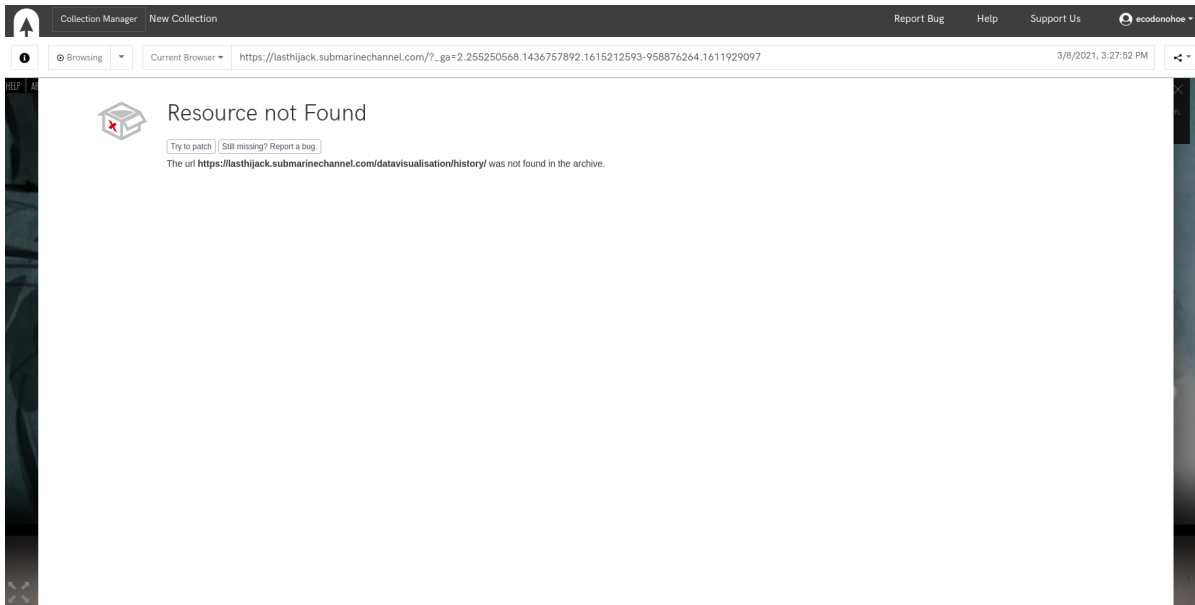
The screenshot shows the 'New Capture' interface in the Conifer application. On the left, there is a browser tab for 'ecodonohoe' with a close button (x) and a checkmark (✓). Below the tab are options to 'Add website...' and 'Add a description...'. The main form area is titled 'New Capture' and contains the following elements:

- A text input field labeled 'URL to capture'.
- A dropdown menu labeled 'Add to collection' with 'New Collection' selected.
- A dropdown menu labeled 'Select browser' with 'Use Current Browser' selected.
- A section for 'Session settings' with a right-pointing arrow.
- A green 'Start Capture' button.

At the bottom of the interface, there are two buttons: '+ New Collection' and 'Upload'.

Conifer capture page

This will launch the target website in the current window with the Conifer interface overlay visible. From here it is down to the user to navigate through the website and interact with the features. When finished, the user stops the capture and is presented with a WARC file that can be opened within the same Conifer interface and interacted with independently of the live website. In my first pass of the website I was interested in what information was and wasn't captured. While interacting with the site I purposely skipped sections or failed to click on links to see how this would affect the capture. As expected, these features did not display on playback, in some instances just appearing as blank boxes within the window and in others providing a “resource not found” message.



A page that was purposely skipped when capturing in Conifer

In a second pass, these same elements were interacted with and subsequently included in the final capture, allowing for playback. The overall quality of the capture was very high, only displaying minimal lag in loading some elements of the page. However, the reality of having to manually interact with each individual feature of the website does make the process of capturing rather time consuming. As this tool is likely to be used for specific, individual use cases on a small scale this may not be too much of an issue but it is worth planning and documenting what and where you are actually capturing when using the tool.

Though conifer focuses on capturing the performative nature of a website, as opposed to the server files themselves, several conversations with developers raised the importance of recording and preserving a snapshot of a website as it is intended to work, look and feel. In the long run, and in combination with the more complete server-side approach, efforts to redeploy old websites can benefit greatly from having some form of reference available.

Reprozip

In order to test and experiment with the reprozip tool some prior setup needed to be carried out. The Reprozip documentation¹¹ contains extensive information on setup, installation and use of the tool but there is still quite a technical barrier in terms of concepts discussed and language used. My own experiments with using the Reprozip tool led to plenty of troubleshooting and often far from desired results, but any failings in terms of reproducing use cases is not necessarily a reflection on the tool itself but rather the steep learning curve that is associated with such technology. Here I will document my steps taken while learning about the tool.

Installation

First up was installation. I used the same Ubuntu machine running version 18.04.4 as before and followed the instructions provided in the documentation:

1. Install prerequisite software packages

¹¹ <http://docs.reprozip.org/en/1.0.x/>

- Rezip requires Python and a pip installer, as well as the below packages by specific component:

Component / Plugin	Required Software Packages
<i>rezip</i>	SQLite, Python headers, a working C compiler
<i>rezip-unzip</i>	None
<i>rezip-vagrant</i>	Python headers, a working C compiler, SSL library [2], FFI library [2], Vagrant v1.1+,
<i>rezip-docker</i>	Docker
<i>rezip-vistrails</i>	None [3]

- All packages could be installed with the following command via the terminal:

```
beng@beng:~$ sudo apt-get install python
python-dev python-pip gcc libsqlite3-dev libssl-dev libffi-dev
```

2. Rezip itself was then installed with:

```
beng@beng:~$ pip install -U rezip
```

3. Lastly, Rezip, the component responsible for unpacking experiments was installed using:

```
beng@beng:~$ pip install -U rezip[all]
```

These steps ensured that all necessary components are present on the machine and the next stage was to trace and package a website.

Tracing and Packing

During the tracing and packing progress Rezip tracks the execution of the website deployment, in this case the files and folders of the target project and the commands to launch the server. The tracing stage identifies all the operating system calls used during deployment and captures the environment variables necessary for future re-deployment.

This is achieved by using the following command in conjunction with the original command line instructions used to deploy start the server:

```
beng@beng:~/Desktop/Submarine/TheLastHijack/httpdocs$ rezip trace
php -S localhost:8000
```

After accessing the deployed project at the specified port the server was then stopped and a configuration file was automatically created detailing the contents to be included in the packing stage. This file could be edited to include additional material where needed but in general should contain everything required for redeployment. Note, the configuration file on my machine existed in a hidden folder located in the same directory where the trace was run. To access it the command line and Gedit were used.

```
beng@beng:~/Desktop/Submarine/TheLastHijack/httpdocs/.rezip-trace$
gedit config.yml
```

```

config.yml
~/Desktop/Submarine/TheLastHijack/httpdocs/reprozip-trace
Save

# ReproZip configuration file
# This file was generated by reprozip 1.0.16 at 2021-03-10T17:59:05+01:00

# You might want to edit this file before running the packer
# See 'reprozip pack -h' for help

# Run info
version: "0.8"
runs:
# Run 0
- architecture: x86_64
  argv: [php, -S, 'localhost:8000']
  binary: /usr/bin/php
  distribution: [ubuntu, '18.04']
  environ: {CLUTTER_IM_MODULE: xim, COLORTERM: truecolor, DBUS_SESSION_BUS_ADDRESS: 'unix:path=/run/user/1000/bus',
    DESKTOP_SESSION: ubuntu, DISPLAY: ':0', GDMSESSION: ubuntu, GJS_DEBUG_OUTPUT: stderr,
    GJS_DEBUG_TOPICS: JS ERROR;JS LOG, GNOME_DESKTOP_SESSION_ID: this-is-deprecated,
    GNOME_SHELL_SESSION_MODE: ubuntu, GNOME_TERMINAL_SCREEN: /org/gnome/Terminal/screen/
0f7a0149_75c5_4e4e_8a8e_aae8bbdc793f,
    GNOME_TERMINAL_SERVICE: ':1.225', GPG_AGENT_INFO: '/run/user/1000/gnupg/S.gpg-agent:0:1',
    GTK_IM_MODULE: ibus, GTK_MODULES: 'gail:atk-bridge', HOME: /home/beng, IM_CONFIG_PHASE: '2',
    LANG: en_US.UTF-8, LC_ADDRESS: nL_NL.UTF-8, LC_IDENTIFICATION: nL_NL.UTF-8, LC_MEASUREMENT:
nL_NL.UTF-8,
    LC_MONETARY: nL_NL.UTF-8, LC_NAME: nL_NL.UTF-8, LC_NUMERIC: nL_NL.UTF-8, LC_PAPER: nL_NL.UTF-8,
    LC_TELEPHONE: nL_NL.UTF-8, LC_TIME: nL_NL.UTF-8, LESSCLOSE: /usr/bin/lesspipe
%s %s, LESSOPEN: '| /usr/bin/lesspipe %s', LOGNAME: beng, LS_COLORS:
'rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su
7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01
OLDPWD: /home/beng, PATH: '/home/beng/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin',
    PWD: /home/beng/Desktop/Submarine/TheLastHijack/httpdocs, QT4_IM_MODULE: xim,
    QT_ACCESSIBILITY: '1', QT_IM_MODULE: ibus, SESSION_MANAGER: 'local/beng:@/tmp/.ICE-unix/
3651,unix/beng:/tmp/.ICE-unix/3651',
    SHELL: /bin/bash, SHLVL: '1', SSH_AGENT_PID: '3747', SSH_AUTH_SOCK: /run/user/1000/keyring/ssh,
    TERM: xterm-256color, TEXTDOMAIN: im-config, TEXTDOMAINDIR: /usr/share/locale/,
    USER: beng, USERNAME: beng, VTE_VERSION: '5202', WINDOWPATH: '2', XAUTHORITY: /run/user/1000/
adm/Xauthority
}
YAML Tab Width: 8 Ln 1, Col 1 INS

```

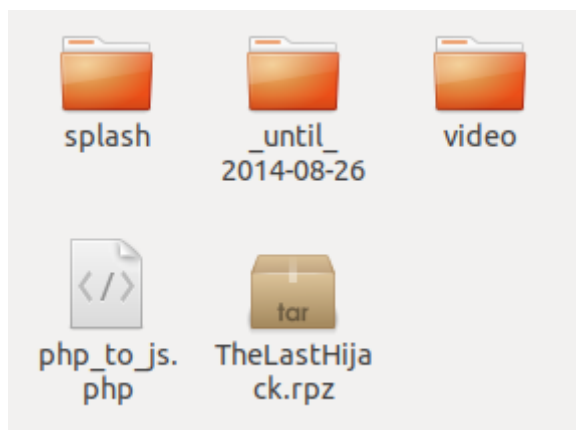
Extract from Config.yml file

Once the configuration file was ready, the final step was to pack the files, detailing the name of the new .rpz file that was to be created. This would then appear in the same target directory as the trace.

```

beng@beng:~/Desktop/Submarine/TheLastHijack/httpdocs$ reprozip pack
TheLastHijack

```



TheLastHijack.rpz file

Repronzip and Unpacking

The Reprounzip component is used to unpack and redeploy the Reprozip package. The documentation¹² details several possibilities for this process and describes which unpackers to use in different scenarios. A number of these options were tested, on a variety of operating systems, and to a varying degree of success.

Firstly, as the original machine used to create and package the Reprozip file was Linux, I looked at the three unpackers that come with Reprozip, and that are only compatible with a linux operating system. These were *reprozip directory*, *reprozip chroot*, and *reprozip installpkgs*. The commands for the unpacking step are the same across each of the unpackers but the only one I experienced any success with was the *reprozip directory* option. Below details the steps I took.

The *directory* Unpacker: Unpacking as a Plain Directory

The *directory* unpicker (`repronzip directory`) allows users to unpack the entire experiment (including library dependencies) in a single directory, and to reproduce the experiment directly from that directory. It does so by automatically setting up environment variables (e.g.: `PATH`, `HOME`, and `LD_LIBRARY_PATH`) that point the experiment execution to the created directory, which has the same structure as in the packing environment.

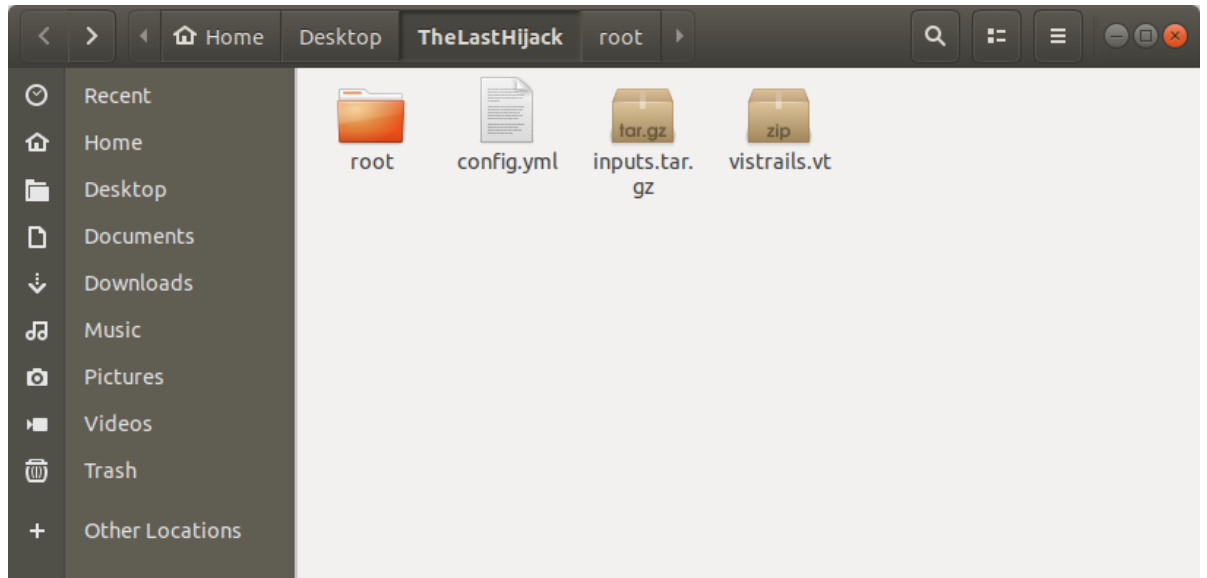
1. The Reprozip “.rpz” file created in the packing step was first copied to a Desktop location to test the unpacking from an entirely new path.
2. After moving into the new current working directory for this package the following command was run to setup the unpack the experiment:

```
beng@beng:~/Desktop$ reprozip directory setup TheLastHijack.rpz TheLastHijack
```

Here, the *reprozip directory setup* command is followed by the name of the package and the name of the target directory.

3. This stage may take some time to complete but when ready the target directory will contain the unpacked files and folder structure and configuration file from the original Reprozip package.

¹² <https://reprozip.readthedocs.io/en/1.0.x/unpacking.html>



4. To run the newly unpacked experiment again the following command was used:

```
beng@beng:~/Desktop$ reprunzip directory run TheLastHijack
```

This initiates the original command line instructions that were run when tracing the deployment of the website, in this case running the website on the PHP built-in server on localhost:8000.

```
PHP 7.2.24-0ubuntu0.18.04.7 Development Server started at Fri Mar 12 15:17:39 2021
Listening on http://localhost:8000
Document root is /home/beng/Desktop/TheLastHijack/root/home/beng/Desktop/Submarine/TheLastHijack/httpdocs
Press Ctrl-C to quit.
```

From here the project can be interacted with via the browser at the designated port. In the following section I will briefly describe the results of the unpack and redeployment as well as some of the issues encountered when attempting to use other unpacking options.

5. Results

Upon redeployment of the *Last Hijack* example it was clear that information was missing. The website loaded as normal but when navigating through sections of the site links became broken or pages didn't display. The packing and unpacking step was repeated several times and it was made clear that the manual traversal of website elements was needed in order to capture those files. Alternatively, this would be possible by including all the necessary files in the configuration file at the editing stage. Due to time constraints this wasn't possible but is interesting to be aware of the added manual intervention that needs to be carried out in order to pack a complete version of the website. Aside from this, the files and elements of the website that were captured, unpacked and redeployed worked seamlessly, as if deploying directly from the original source folder. This displays great promise for the tool and its ability to compress and extract the vital information needed to redeploy the project.

Of course, the use of the *reprounzip directory* unpacker in its native Linux environment had the potential to make efforts more straightforward. Other options for unpacking in alternative environments, i.e Windows and MacOS, include Vagrant¹³ and Docker¹⁴. Both options were explored but added their own level of complexity that comes with using a new technology. The Vagrant unpacker allows experiments to be unpacked in a virtual machine and run in an emulated environment while the Docker version can extract and reproduce experiments as docker containers, meaning that any environment where these technologies are available should work. These unpackers follow the same commands for setup and running of a Reprozip package, as seen from the MacOS terminal

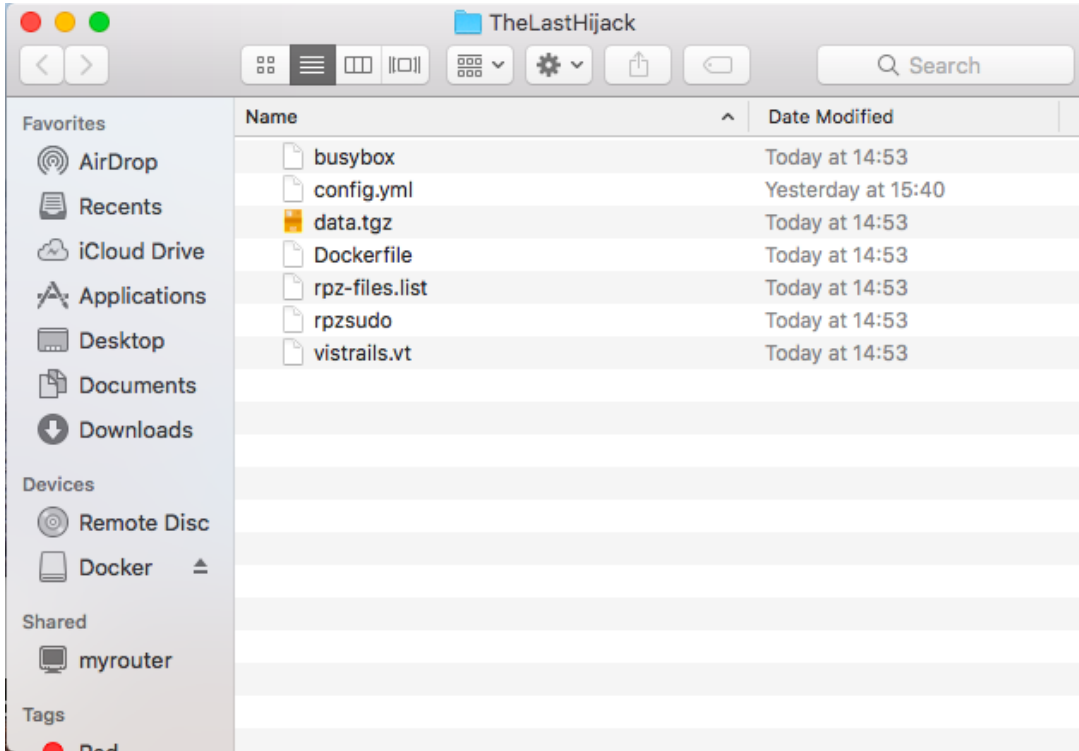
```
EoinsMacBookAir:Desktop ecodonohoe$ reprounzip docker setup TheLastHijack.rpz  
TheLastHijack
```

```
EoinsMacBookAir:Desktop ecodonohoe$ reprounzip docker run TheLastHijack
```

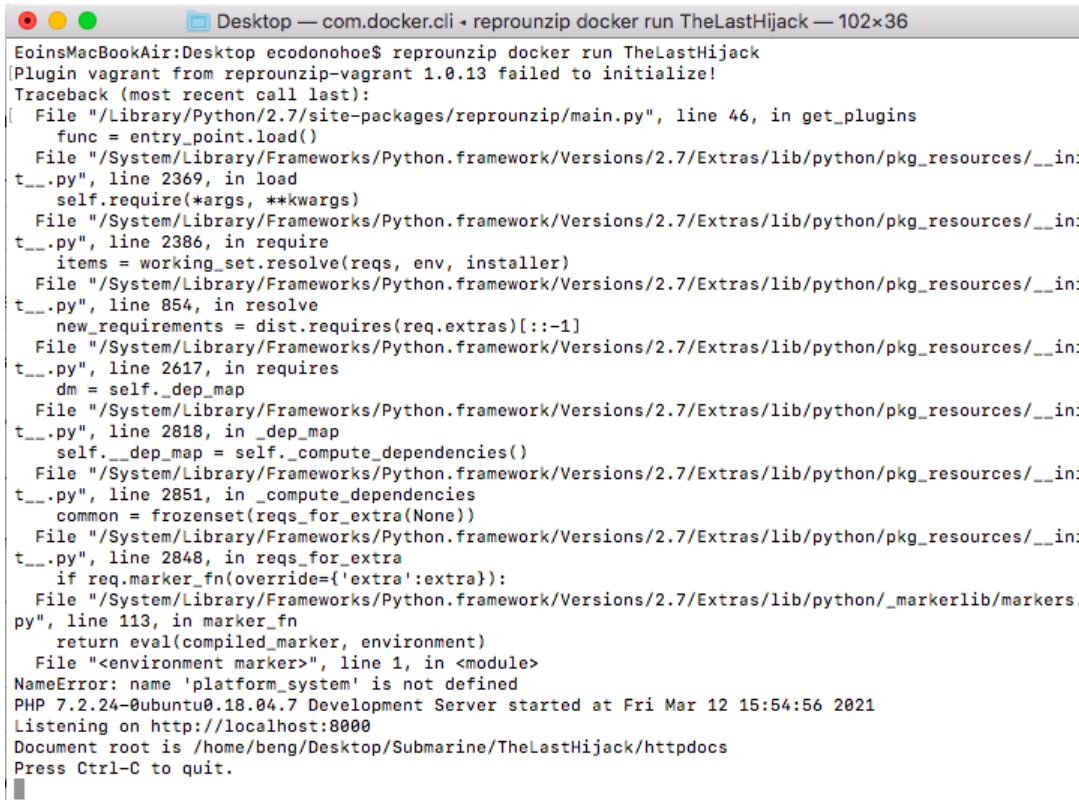
The unpacked project was created with all the same contents and some additional Docker specific files while the run command encountered what appears to be some Python related errors that I was unable to interpret. The same issue was encountered while working on a Windows 10 machine. The execution of and redeployment of Reprozip packages in different environments is a major draw for the technology. The promising results of the native Linux *Directory* unpacker hopefully show that with a greater understanding of the peripheral tools, such as Vagrant and Docker, greater success could be achieved in other environments.

¹³ <https://www.vagrantup.com/>

¹⁴ <https://www.docker.com/>



The Last Hijack unpacked using docker on MacOS 10.13.6



Terminal display when running `repronzip docker run TheLastHijack`

6. Recommendations and Conclusions

Experimentation with the different tools in this report has highlighted the technical barrier facing heritage institutions when it comes to dealing with such complex examples. Not only does the target of preservation itself present unique challenges and barriers when it comes to understanding its construction, the tools used to document and capture such sites also require a specific technical knowledge for setup, use and troubleshooting. This report aimed to show the challenges faced while testing such tools and though the results did not display clear success across multiple platforms there were some promising possibilities to build on going forward. It should be noted what Rezip can and cannot capture:

Can capture

- Currently, experiments originally run on Linux
- Binaries, files, dependencies and environment variables that are required to run an experiment
- Databases and server-client interactions

Cannot capture

- Missing or outdated dependencies: if the website relies on an older version of a piece of software that is no longer supported, Rezip cannot fix this issue itself.
- Browser specific requirements: similarly, Rezip only captures the operating system environment information for experiments. If a website depends on a specific browser version this will need to be sourced elsewhere and likely redeployed in an emulated environment

In light of this, below are the proposed steps for the documentation and capture of dynamic websites.

1. Selection

- Due to the time consuming and labour intensive nature of capturing dynamic websites, the selection criteria used when choosing projects to preserve should be considered carefully.
- At risk technologies should be identified and prioritised with websites using these technologies a primary focus.

2. Collaboration with Creators

- Website creators should be involved from as early as possible and can act as a valuable source of technical information.
- Expectations and division of labour, where necessary, should be laid out at an early stage. Knowing what each party is set to gain and what they should bring to the table is essential for clear and focussed communication.

3. Documentation

- As early as possible, efforts should be made to document the website in operation in its optimal setting. This can be as simple as a video walk through of the website or, as detailed in this report, a high fidelity capture of a full site walkthrough using the Conifer tool.
- Technical listing of environment and software dependencies, programming languages and version information. A technical interview with creators can help here.

4. Server-side Capture

- Transfer, setup and deployment of website server side files on a suitable system or, ideally, access to the original server environment where the files are stored
- Installation and execution of Rezip tool, editing of configuration file, packing and quality assurance of package.
- The Rezip package should be tested on a variety of operating systems using different unpackers.
- Troubleshooting and recapture as needed.

The preservation of dynamic websites is a difficult task due to the complex environments of such projects. The basic groundwork of identifying what should be captured and preserved as well as the execution of such a goal is always going to be a more time consuming and labour intensive task than traditional web archiving but the need to save these works is as important for both creators and cultural institutions. Both will need to work closely to set goals and achieve results. If they do, server-side web archiving will provide another preservation strategy for dynamic web content.

7. Glossary of Web Browser – Related Terms

Add-on

See [browser add-on](#)

Application Programming Interface (API)

An Application Programming Interface (API) is a kind of hyper-programming language or library (or glue) that serves to link external services and programs to a specific platform. For instance, Firefox created an API in order to enable external programmers to create their add-ons for the Firefox browser. The APIs are usually specific to a browser. For instance, the Firefox API cannot be used for Google Chrome.

Bandwidth

In the context of the internet, bandwidth means the highest rate of data transfer possible on a specific communication path. It is also called network bandwidth and is usually stated in bits per second.

Browser

See [web browser](#).

Browser add-on

A browser add-on (also known as a browser extension) is a small programming module that extends a web browser's functionality—for instance, by adding a toolbar, enabling or disabling plug-ins, or integrating notebooks or video calling. Browser add-ons are written in the same programming language as websites (HTML, Javascript, CSS) and need to use the browser-specific API.

Browser emulation

A browser emulation serves to run an obsolete [web browser](#) in order to access obsolete websites and certain web archives. For instance, websites that need plug-ins can only be rendered in obsolete browser environments. As obsolete web browser environments only run on obsolete computer hardware, this hardware needs to be emulated on any client computer with current hardware. The emulation of the obsolete computer hardware, including the installation of the obsolete browser environment, is called "browser emulation."

Browser extension

See [Browser add-on](#).

Browser plug-in

A browser plug-in is an executable, usually made and licensed by an external party in order to render video, audio, and web animations. It is a deprecated technology. Before the introduction of HTML5 in 2014, web browsers were not able to render audiovisual content except for still images. It took several years until web designers adopted HTML5 for video and audio. This is the reason why browser plug-ins play an important role in rendering obsolete websites. Examples of browser plug-ins include the [Flash plug-in](#), [Shockwave plug-in](#), and [Java plug-in](#).

Cascading Style Sheets (CSS)

CSS is a language that serves to create a template for the layout of a website. In this way, the content of the website is separated from its layout. This procedure makes it more efficient to add new web pages or more content to a website.

Client-side

A website is hosted on a web server and accessed through a client computer. The client computer holds the web browser that is needed to render the website. The web server and client computer are connected through the internet. “Client-side” refers to the computer with the web browser.

DNS-Server

The Domain Name System (DNS) is a decentralised inventory of URLs and the corresponding IP addresses of the web servers where the websites are hosted. A DNS-Server contains such an inventory. A user’s internet service provider provides such DNS-Servers. DNS-servers are organized hierarchically.

Document Object Model (DOM)

A Document Object Model is a programming interface used to structure websites in order to enable the web browser to render them efficiently. The idea is that web pages with embedded scripts do not have to be reloaded from the web server when executing the script. To achieve that the DOM structures a web page as a logical tree.

Dynamic web page

A dynamic web page is a web page whose content changes dynamically, usually as a reaction to user input or to other inputs. There are web pages whose dynamics are embedded in [client-side](#) code, such as Javascript, and can be interpreted by the web browser. Other websites contain [server-side](#) code such as PHP, Python, or Ruby, which are executed on the web server.

Emulation as a Service (EaaS) / Emulation as a Service Infrastructure (EaaSI)

Emulation as a Service is an emulation platform developed by the bwFLA team at the University of Freiburg (D) that is now distributed by OpenSLX. The platform offers a collection of emulators and tools to manage created environments. The EaaS platform can be made accessible in a web browser with internet access. Emulation as a Service Infrastructure ([EaaSI](#)¹⁵) is an implementation of the EaaS platform at Yale University. The university and OpenSLX made a user documentation¹⁶ of EaaSI available.

Executable

An executable is a file that needs neither a compiler nor an interpreter to be executed. In contrast to [source code](#), it is usually not human readable but contains machine code that is executable on a specific operating system and computer architecture.

Extension

See [browser extension](#) and [browser add-on](#)

Flash plug-in

The Flash plug-in is a browser plug-in that is able to render vector-based animations with the file extension SWF. It was developed for the World Wide Web. From 2005 until about 2017, Adobe

¹⁵ <https://www.softwarepreservationnetwork.org/eaasi-gitlab/> (accessed october 2020)

¹⁶ https://eaasi.gitlab.io/eaasi_user_handbook/ (accessed october 2020)

Systems developed Flash and [Shockwave](#). Flash files load more quickly whereas Shockwave is more versatile. Flash and Shockwave are both closed source.

GIF

The Graphics Interchange Format (GIF) is an image format based on bitmap. It can store several images in one file and is usually used for short animations. GIFs have been popular since the beginning of the internet.

Graphical User Interface (GUI):

A graphical user interface can be used instead of command line or text interfaces when interacting with electronic devices like computers. A GUI application usually opens in a window that consists of graphical elements such as icons and menus. The user interacts with the computer or application with the mouse by manipulating certain graphical elements instead of entering a command line in text form. A GUI is usually considered more user friendly than a command line interface.

HTML

Hypertext Markup Language (HTML) is the language of the World Wide Web. HTML is a markup language, mainly used to present the content of a website. The interpretation of HTML is the core function of every web browser. In the beginning of the World Wide Web, web pages consisted only of HTML. Nowadays HTML is often accompanied by [CSS](#) and [Javascript](#) (client-side).

HTML5

HTML5 is a major revision of HTML that had an impact on the way media was integrated in websites. It was introduced in 2014 and features, amongst other tools, new video and audio tags in order to handle media natively. As a consequence, plug-ins such as Java or Shockwave were made obsolete on websites that used these new features. Another important change was the inclusion of the DOM and the scripting API for [Javascript](#) within the [HTML](#) specifications. Through the integration of the Javascript API “webGL” browsers can render interactive 2D and 3D graphics natively.

HTTP

The Hypertext Transfer Protocol (HTTP) is an information transport protocol used when a user loads and interacts with a website. It is a request-response protocol. The web browser (client) requests information from the web server and the web server sends a response to the web browser, which could be [HTML](#) files or other resources like image, sound, or video files. HTTP is the highest level of transport protocol in the internet and it is used for web crawling.

HTTPS

HTTPS or Hypertext Transfer Protocol Secure is the encrypted version of [HTTP](#). It secures the communication between web browser (client) and web server, which could be otherwise intercepted and read. In the 2000s, HTTPS was mainly used for payment services such as for banks and online shopping. All other websites used HTTP. Only in the past few years has HTTPS become common for all websites, and [web browsers](#) have started to block HTTP-websites.

Java applets / Java plug-in / Java Runtime Environment

The Java plug-in is a [browser plug-in](#) that can render interactive animations and small design elements like rollover buttons. These animations were called applets. To run Java applets the Java Runtime Environment needed to be installed on the client computer. The Java Runtime Environment is an intermediate layer (a so-called process virtual machine) that makes the java

applets independent from the underlying operating system. According to Wikipedia,¹⁷ the Java Runtime Environment “contains a stand-alone Java virtual machine (HotSpot), the Java standard library (Java Class Library), a configuration tool, and—until its discontinuation in JDK 9—a browser plug-in.” Java was closed source until 2007. From 2010 until 2016 it was developed by Oracle. It was replaced by [Javascript](#).

[Javascript](#)

Javascript is one of the most important web programming languages. It is used to program interactive behaviour on a website and to create web animations. Web browsers can interpret Javascript. Javascript is mainly used for [client-side](#) programming on a web page (called a client-side [dynamic web page](#)), although it can also be used for [server-side](#) programming (server-side dynamic web page).

[Plug-in](#)

See [Browser plug-in](#)

[Server-side](#)

A website is hosted on a web server and accessed through a client computer. The client computer holds the web browser that is needed to render the website. The web server and client computer are connected through the internet. “Server-side” refers to the web server that holds the code for the website.

[Shockwave plug-in](#)

The Shockwave plug-in is a [browser plug-in](#) that can render Shockwave animations with the file extensions DCR, DIR, or DXR. While Shockwave was developed to create CD-ROMs, it was later used for the creation of many online video games. From 2005 until about 2017, Adobe Systems developed [Flash](#) and Shockwave. Flash files load more quickly, whereas Shockwave is more versatile. Shockwave and Flash are both closed source.

[Source code](#)

Source code is higher-level code that is human readable and that cannot be executed without translation into machine code. To execute source code, usually an interpreter or a compiler is necessary.

[Static web page](#)

A static web page directly loads its content from the web server without any server-side processing and without interactive behaviour on the client side. Such pages are usually just an [HTML](#) document and might include layout defined in [CSS](#).¹⁸

[URL](#)

The Uniform Resource Locator (URL) is the address of a website or web page, which points the web browser to the location of the web server holding the website. The address consists of generally meaningful text and is easier for users to remember than the IP address of a web server, which is a long number. In order to look up the IP address of the web server, the web browser contacts a [DNS-server](#) (usually the one provided by the Internet Service Provider of the user).

[Web browser](#)

¹⁷ https://en.wikipedia.org/wiki/Java_virtual_machine accessed 2020/10/21

¹⁸ Other definitions include client-side Javascript functionality in the definition of a static web page (s. https://en.wikipedia.org/wiki/Static_web_page, accessed 2020/09/02)

A web browser is a complex piece of software with a graphical user interface that connects the user to the web server containing the website and enables the rendering of the web page and the interaction between the user and web page. If you look at a web page without a web browser, for instance with the Linux command “curl,” the served web page will consist only of [source code](#)—such as [HTML](#), [CSS](#), or [Javascript](#) code—which would be interpreted by the web browser. A web browser can be installed on clients’ computers and mobile devices.

[World Wide Web \(www\)](#)

The World Wide Web is a network of websites that are identified with Uniform Resource Locators (URL). The users communicate with the web servers with [HTTP](#) and [HTTPS](#) and need [web browsers](#) to render the web pages.

Credits

Eoin O'Donohoe, author of this report, is a Digital Preservation Analyst at the Netherlands Institute for Sound and Vision (Nederlands Instituut voor Beeld en Geluid). He is currently working on areas of research that aim to lower the threshold for institutions looking to make a start on software archiving.

eodonohoe@beeldengeluid.nl

About this publication

This report was published by the Dutch Digital Heritage Network (NDE) in May 2021.
For further information, see: netwerkdigitaalerfgoed.nl

If you have any queries or comments about the contents of the report, please feel free to email us at:
info@netwerkdigitaalerfgoed.nl

