Web browser characterisation, emulation, and preservation

January 2021

Author: Claudia Roeck, Beeld en Geluid

dutch digital heritage network

dutch digital heritage network

Contents

Acknowledgements

We want to thank LIMA for their input to the research questions as well as for suggesting the case studies “Abstract Browsing” by Rafaël Rozendaal and www.tebatt.net. LIMA also helped analysing the case studies www.collapsus.com and www.wefeelfine.org.

About this publication

This report was published by the Dutch Digital Heritage Network (NDE) in December 2020 as part of the software preservation project. For further information, see: netwerkdigitaalerfgoed.nl

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

1. Introduction

This report is written for archivists and conservators who are preserving websites, either through server-side preservation or web archiving. It may also be interesting for artists or other individuals who have websites based on obsolete web technology that they would like to keep accessible. The report focuses on the impact of web browsers on the (online) websites they render and describes how web browsers can be preserved.

This report is part of the NDE Software Archivering project, for which several Dutch institutions provided case studies. One of these collaborating institutions is LIMA, an organisation for the collection, research, and preservation of media art in Amsterdam. LIMA recently set up "Arthost,"[1] a service for artists and institutional and private collectors to host, present, and monitor their net art websites. Arthost uses "browser emulations," which allow users to access obsolete websites via obsolete web browsers, to present net art that is a few years old using web browsers that were common at the time when the websites were created. The expression "browser emulation" is a common expression in digital art preservation, although it is not technically accurate: it is not in fact the browser that is being emulated, but the hardware that is running it. Specific web browser versions typically only work on corresponding computer hardware, so old hardware has to be emulated in order to run obsolete browsers. An emulation with an obsolete web browser is installed on a different machine than the web server and automatically started when a user accesses the obsolete website. To create these browser emulations, LIMA is collaborating with Rhizome,[2] a not-for-profit organisation in New York that specialises in the collection and presentation of net art.

One challenge facing this browser emulation project is that not every web browser renders a website in the same way. Certain browsers might even be inapt for certain websites. While web browser developers and other sources such as Wikipedia can provide information about the technical properties of web browsers, they provide only very limited descriptions of browser behaviour. LIMA realised that there was no objective way to decide which web browser should be used for a specific website and suggested that this team research web browsers and develop a way to conceptualise them in terms of the authenticity of websites and in regard to the browsers' provenance, version, and feature history.

1 https://www.li-ma.nl/lima/article/arthost accessed 14/07/2020

2 https://rhizome.org/

2. Problem statement and research proposal

It is the objective of this research to find criteria to differentiate between versions of web browsers and their plug-ins in terms of the look and feel of websites they display and to conceptualise browsers in terms of their preservation. Questions to be answered by this research are:

Research questions

1. How are browsers and browser environments structured and how does that influence their behaviour and properties?
2. How can someone decide which web browser version is best for a specific website in terms of creating an authentic representation of the website?
3. How are browsers and their extensions and plug-ins accessed and acquired? Should they be preserved as executables or source code?
4. How can browser environments be assembled and preserved to present websites?
5. What access problems can a browser emulation solve and what problems can it not solve?

In order to answer the **first and second** question, a small theoretical study is presented in sections 4 and 5. This study discusses:

- The short history of web browser, especially in regard to the development of its significant properties and the development of HTML (web2.0)
- The ecosystem and structure of a web browser: What are add-ons, plug-ins, and other terms. How many browser, extension, and plug-in versions are there?

In addition to the theoretical study, case studies should help to answer the **second** question by contributing practical knowledge. As one website will only bring forward a few browser properties and other properties won't be discovered, the combination of theory and practice is necessary.

The **third, fourth, and fifth** questions are addressed through case studies (see section 6). By setting up browser emulations, the problems of the acquisition and installation of browser components will manifest themselves. Potential problems with websites might emerge. In addition, practical knowledge about the creation of browser environments will be acquired.

The answers to the above-mentioned research questions will be discussed in sections 7 to 12.

3. Existing projects and research

LIMA has previously done research about web archiving (comparing different web archiving tools and their suitability for the preservation of netart) as part of the Arthost project, but this did not involve researching the browsers themselves. The results of this research are published on LIMA's website[3].

According to Dragan Espenschied[4], Rhizome is engaged in ongoing research on the significant properties of web browsers and how they affect the look and feel of websites. They are also investigating whether it is possible to deduce the browser software of websites from screenshots or screencasts. According to Espenschied, they are planning to publish their research at some point in 2020.

One of Rhizome's projects is Oldweb.today. On the Oldweb.today[5] platform the user can choose a web browser and access an archived website by entering a URL and choosing a year (see Figure 1 and Figure 2). Oldweb.today then searches for the website in a list of web archives, including the largest web archive, the Internet Archive. Oldweb.today is a great way to access archived websites in a more authentic way than with a modern web browser, and as no specialised software is necessary, it is highly accessible. Oldweb.today uses a selection of obsolete web browsers (fourteen out of several hundred possible options), but it does not explain how they chose these specific browser versions and what features (plug-ins and extensions) they provide.

[3] https://www.li-ma.nl/lima/news/arthost-end-report accessed 14/07/2020

[4] email from December 2019

[5] http://oldweb.today/

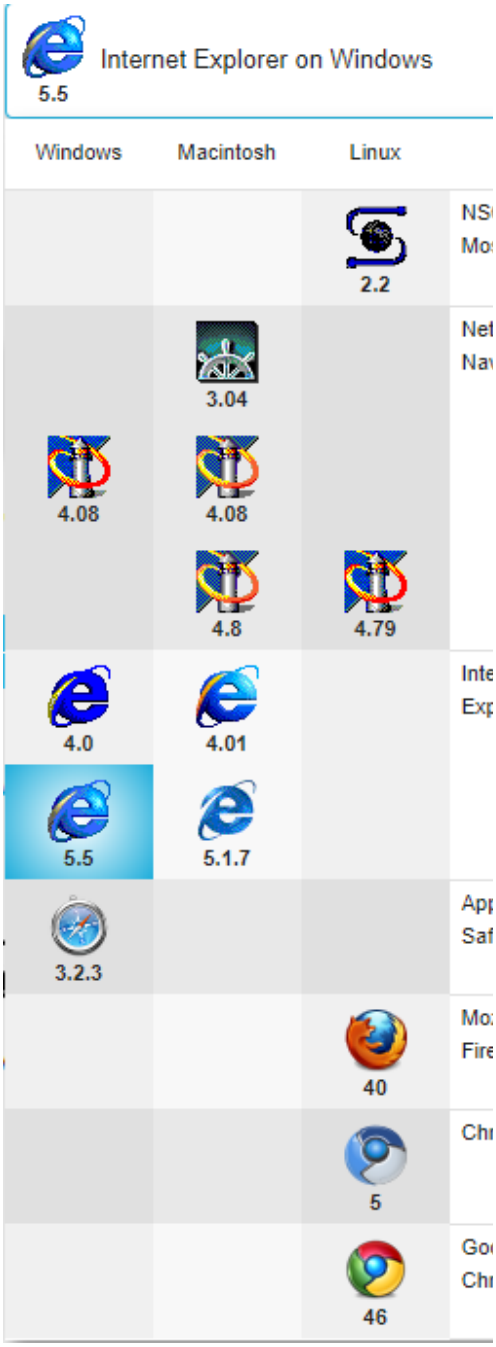


Figure 1 Selection of obsolete web browsers in oldweb.today (access 2020/07/21)

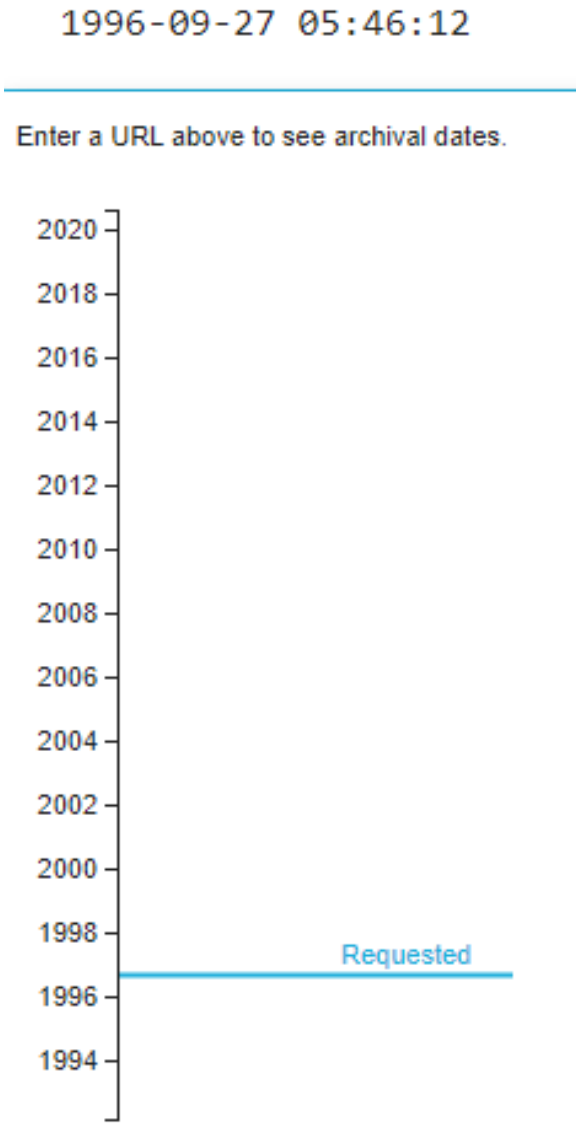


Figure 2 Selection of years to access the web archive of a specific year in oldweb.today (access 2020/07/21)

Rhizome's Conifer project[6] (formerly known as Webrecorder) is a web archiving tool with advanced features. Conifer enables users to archive websites, but in contrast to a standard web archiving tool, it captures what is requested by the browser and its plug-ins (Espenschied, Dragan; Kreymer, Ilya, 2016) and plays these streams back through the same browser and plug-ins. Conifer lets the user choose from a small selection of obsolete browsers for the recording and display of the archived websites. Here again, the same question arises as for the Oldweb.today: How did Rhizome choose this small selection of web browsers out of the several hundred browsers and their versions that exist?

[6] https://conifer.rhizome.org/ accessed 21/07/2020

To put it more generally, what are the criteria one should use when selecting an obsolete web browser to present a specific website?

Our project team found only a few papers about the behaviour and preservation of web browsers. Horsman researches how web browsers cache websites (Horsman 2018), with a focus on forensic investigations. Alcorn et al. (2014) provide a guideline to hack computers through the web browser, looking at browsers through the lens of a hacker or information security agent. Bang (2004) investigates automatic harvesting of browser plug-ins and concludes that this is not cost-effective. Xu and Miller's (2018) research comes closest to the research question of this report: they are looking for a method to quantify the differences among websites presented in different web browsers. However, the quantification does not give information about the nature of the differences (for instance, a missing icon, different animation, or different website layout). Neither does it provide information about the characteristics of web browsers. This report is trying to diminish the research gap on the characteristics of web browsers in regard to the authentic representation of websites, and on the access and preservation of obsolete browsers.

However, web browsers are complex software objects. Ferdman et al. compare a web browser with an ecosystem (Ferdman et al. 2017, p. 1). Alcorn et al. remark that "web browsers had one of the most dramatic and exciting evolutions in the information technology industry" (Alcorn et al. 2014, p. 12). The complexity and dynamic nature of web browsers make it a challenge to keep an overview of their versions, features, and requirements. We decided that learning more about the technical properties and the historical development of web browsers would help us to better analyse the differences among browsers and understand how they can be best described and preserved.

4. History of browsers

4.1. Brief history of browsers and the browser wars

Browsers opened the doors to web culture by making interactive websites with embedded media accessible to a wide public. A short overview of the history of browsers, the impact they had on the presentation of websites, and their relation to the web culture is provided here.
In 1991, Tim Berners-Lee provided the base for the internet by creating Hypertext Markup Language (HTML), which is the programming language of a website that provides it with the capability to link to other websites. Berners-Lee invented the HTTP protocol to enable an exchange of information between a user's computer and a web server, and he designed the first website[7]. In 1992, he introduced the Uniform Resource Locator (URL), a way to describe the location or "address" of a web resource.

The first web browser was a text-based browser called Lynx that was released in 1992. It was followed by the popular Mosaic browser in 1993, which had a graphical user interface and was able to display images within the web page window. Mosaic was later acquired by Microsoft and used for the design of the Internet Explorer. In 1994 the World Wide Web Consortium (W3C) was founded in order to standardize web technology and make recommendations for web browsers. This was also the year that the popular browser Netscape Navigator was produced, which became the de facto standard for Windows machines. The first browser war[8] between Netscape Navigator and Internet Explorer, beginning in the mid-1990s and ending with the demise of Netscape Navigator in the early 2000s (s. Figure 3). This competition was the driver for the development of many new advancements in browser technology. For instance, Netscape Navigator contributed significantly to the development of JavaScript, which is one of the most important server-side programming languages of the web because it enables the animation of page elements and the interactive behaviour of websites. Netscape also introduced the "frame" tag[9] in HTML to enable frames[10] on websites, which later became standard. (The "blink" tag,[11] which made the text on a website blink, was another popular HTML feature developed during this time that did not become a web standard.) Internet Explorer also developed a way to support nonstandard features of websites, such as vertical text and special page transitions[12]. As different browsers innovated rapidly, many websites used browser-specific features that could not be interpreted by all browsers of the time.

7 http://info.cern.ch/hypertext/WWW/TheProject.html accessed 2020/07/30. That was the first website in the www. It explained the what the www project was about.
8 https://en.wikipedia.org/wiki/Browser_wars accessed 2020/07/30
9 https://en.wikipedia.org/wiki/Netscape_Navigator accessed 2020/07/30
10 https://www.w3.org/TR/html401/present/frames.html accessed 2020/07/30
11 https://en.wikipedia.org/wiki/Blink_element accessed 2020/07/30
12 https://en.wikipedia.org/wiki/Internet_Explorer#Non-standard_extensions accessed 2020/07/30

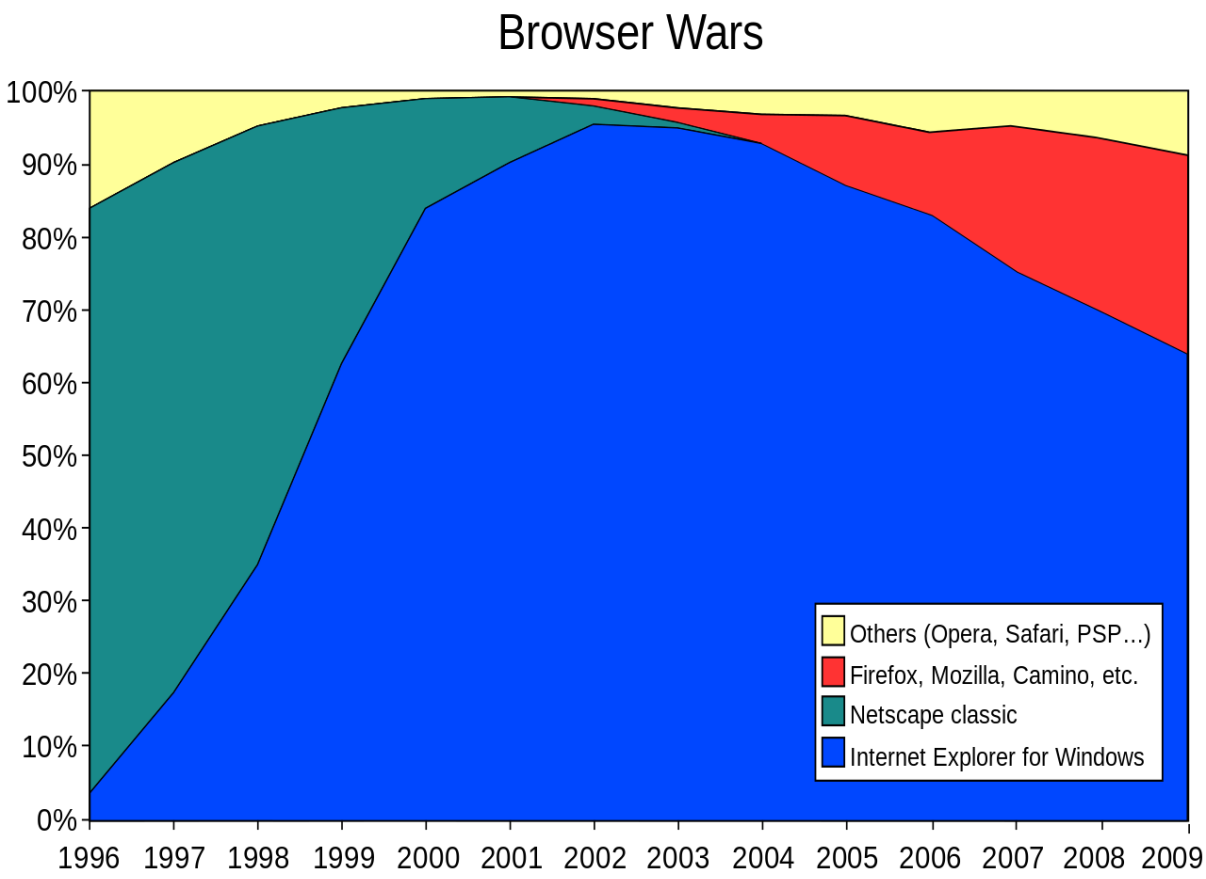


Figure 3 ***Usage share of browser during the first browser war. Source: https://en.wikipedia.org/wiki/Usage_share_of_web_browsers***

The second browser war[13] between 2004 and 2017 saw new browsers such as Google Chrome, Opera, and Firefox[14] emerge and the dominance of Internet Explorer decrease (s. Figure 4). During this time, some browser companies recognized the need for open web standards (such as the Document Object Model,[15] and ECMA for JavaScript) and collaborated in order to further develop them. In contrast to the first browser war, this period was dedicated to the expansion of and adherence to web standards—, therefore, the differences between web browsers of this era are less pronounced than during the first browser war.

13 https://en.wikipedia.org/wiki/Browser_wars accessed 2020/07/30

14 Mozilla (Firefox) rose from the ashes of Netscape Navigator whose code was released under an open source license in 1998

15 https://www.w3.org/TR/WD-DOM/introduction.html accessed 2020/07/30

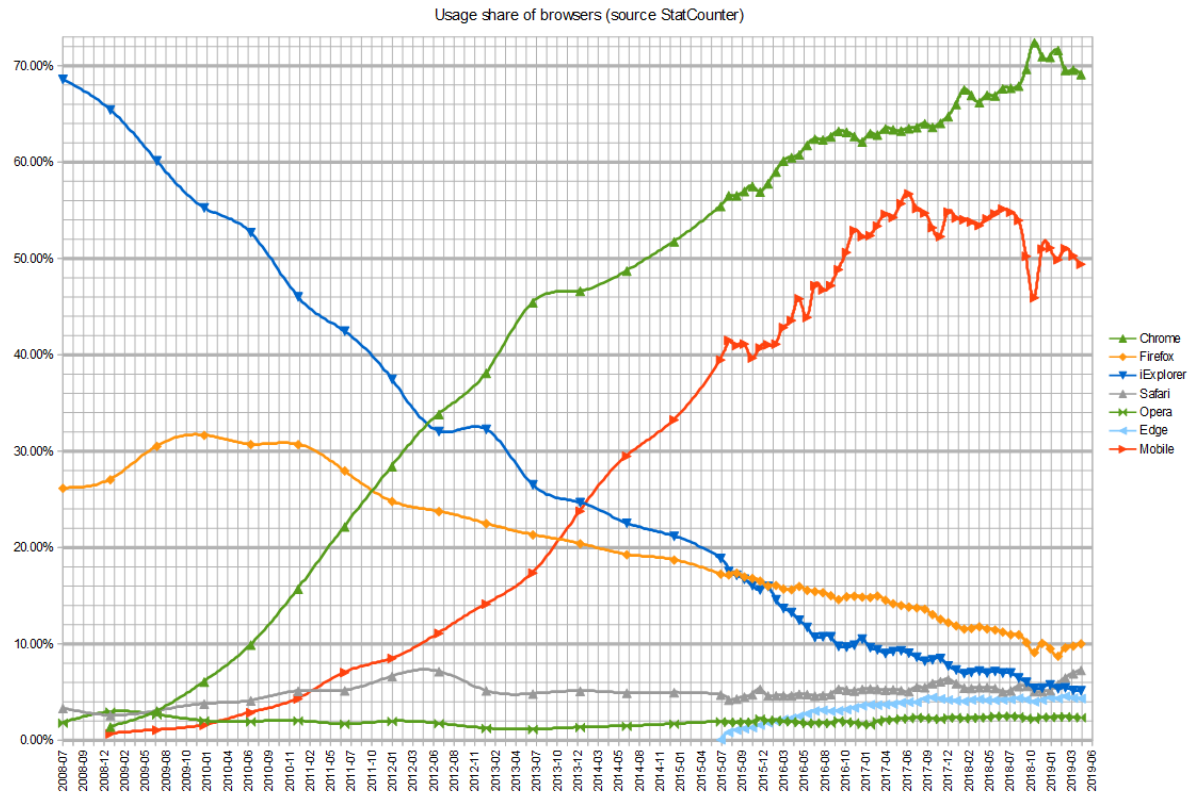


Figure 4 ***Usage share of browser according to StatCounter during the second browser war. Source: https://en.wikipedia.org/wiki/Usage_share_of_web_browsers***

The most recent notable developments in web browser technology are mobile web browsers, which, among other features, can automatically scale websites down to the screen dimension of the phone (if the websites are designed to do so). As mobile phones are built with more and more sensors that measure parameters such as the phone's position, proximity, acceleration, and exposure to light and sound, mobile browsers need a standardised way to access this hardware to support websites that use it[16]. For example, websites or apps for making video calls need direct access to the phone's camera and microphone. A standardised hardware access is not only relevant for the design of websites, but also for their preservation.

4.2. Brief history of online-video and its technology

In the past two decades, video streaming has become an integral feature of the internet. Websites with embedded videos began to emerge around 1995. Websites, which stream the video from the web server to the client, are examined in this report. For instance, www.collapsus.com, a website from 2010 and a case study explained later in this document, contains flash video which is streamed from a flash server. Other websites such as www.wishingtree.nl, also a case study in this report, store their flash video on the local web server, from where it is streamed to the client. Furthermore, in the 2000s, live video streaming became more and more popular and started to be a competitor of TV. As the uploading, downloading, and streaming of video is data intensive, the development and dissemination of the streaming technology is correlated with the development of internet bandwidth (s. Figure 5).

16 Further information: https://www.w3.org/2015/08/mobile-web-app-state/#Sensors_and_hardware_integration

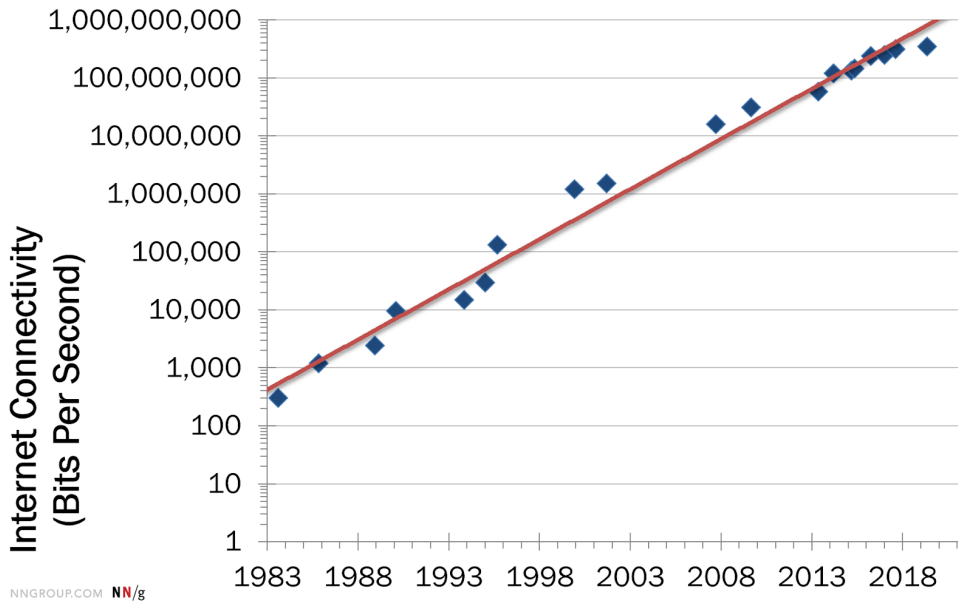


Figure 5 Development of internet bandwidth. Source: https://www.nngroup.com/articles/law-of-bandwidth/

Mosaic was the first web browser that could render images embedded in websites in 1993.[17] The early 1990s also saw the emergence of GIFs and the first webcams— one early example was from 1993, when computer scientists from Cambridge University set up a webcam that was focused on a coffee maker in order to avoid walking in vain to an empty pot[18]. In 1995, the first video player plug-ins were developed; these plug-ins, such as Realplayer, enabled users to play back audiovisual streams within web pages. Today, embedded audiovisual files cause problems for the playback with current, modern web browsers. Peer to peer file sharing services for video, games, and sound (for instance, Napster for mp3 files or eDonkey2000 for all sorts of files such as CD images, videos, games and software) emerged around 1999. They were followed in 2005 by YouTube, a platform where videos can be viewed without the need to download.

Interactive animations and graphics in websites were enabled by Java and Flash technology from 1995 until about 2018. These technologies were enthusiastically used by companies, artists, and game developers and are deprecated today. However, Flash was not an open standard and the licenses were controlled by Adobe (the situation was similar for Java) which is why W3C[19] strived for an open standard integrated in HTML. Websites based on Java and Flash can only be rendered by using old browser environments. With the advent of HTML 5 and the WebGL-API and the evolution of Javascript, browsers are no longer dependent on external web plug-ins.

17 https://crossbrowsertesting.com/blog/test-automation/history-of-web-browsers/ and https://en.wikipedia.org/wiki/Mosaic_(web_browser) accessed 2020/09/22

18 https://en.wikipedia.org/wiki/Trojan_Room_coffee_pot accessed 2020/09/22

19 https://www.w3.org/Consortium/

5. Technology of browsers

5.1. Introduction

In order to conceptualise browsers, it is important to have a basic understanding of their structure. To view a website, three “machines” are necessary: a web server where the website is hosted, a client—a web browser—that renders the website, and a DNS server that supplies the address of the requested website. The user only needs to install the web browser on his/her computer; the servers are part of the internet’s infrastructure. This is called the “client-server model” (s. figure 6).

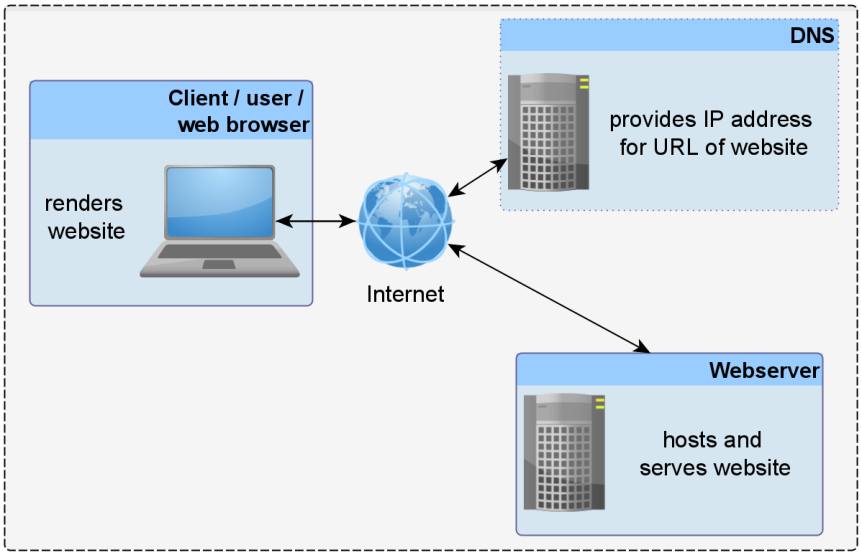


Figure 6 Client–server model for a website

The successful rendering of a website depends on the “mutual understanding” of the web server and the browser client. Within this model, the client and server communicate based on the request–response process of the HTTP protocol. The server delivers the website usually in HTML form, a markup language used to display the content of websites; The HTML document can contain a Cascading Style Sheet (CSS), used to lay out the websites (separation content from style) and Javascript, the most common scripting language used to make websites interactive. The Document Object Model (DOM) is a programming interface helping to structure a website in order to allow a script[20] to change part of the website without having to reload the entire website from the web server.[21] HTML, CSS, Javascript and the document object model are interpreted by the web browser. In contrast, scripts (computer programs) that are executed on the web server are called server-side scripts. Typical programming languages for server-side scripts are PHP, PERL, and Ruby. After the execution of a server-side script the web server provides the results to the browser as HTML code, and the browser renders the resulting website on the computer monitor.

20 A script embedded in the website such as a javascript-script. It enables reaction to user input, such as a user ticking a box on a website.
21 Further information in Alcorn, Wade; Frichot, Christian; Orru, Michel (2014): The Browser Hacker’s Handbook, p 7.

In order to ensure that websites are readable from any browser or hardware, the organisation W3C defines standards for HTML, CSS, the DOM, and other languages. These days, most web designers and browser developers follow these standards. This was not the case in the early days of browser development, which meant that certain website features did not work with certain browsers (see chapter 4.1).

Web browsers used to be much simpler in the early days of the internet: They could only display HTML-based websites and follow hyperlinks. “Now they have support for add-ons, plug-ins, cameras, microphones, and geolocation,”[22] they make use of the DOM, and are able to interpret the newest versions of HTML. In the following sections, the browser engine, browser APIs, plug-ins, and add-ons will be briefly explained.

5.2. Browser engine

A browser engine (s. Figure 7) —also called a layout engine—is the core of a web browser, rendering websites and providing the graphical experience for the user. According to Alcorn et al,[23] the most common graphical rendering engines, which the most popular browsers employ, are WebKit (which is used by Safari), Blink (which is used by Google Chrome and Opera), Trident (which is used by Microsoft), and Gecko (which is used by Firefox). The Wikipedia site “Timeline of Web Browsers” provides an overview of the evolution of most current web browsers, sorted by their browser engines. The browser engine needs to master different web languages (HTML, Javascript, etc.) and the Hypertext Transfer Protocol (HTTP) in order to render a website.

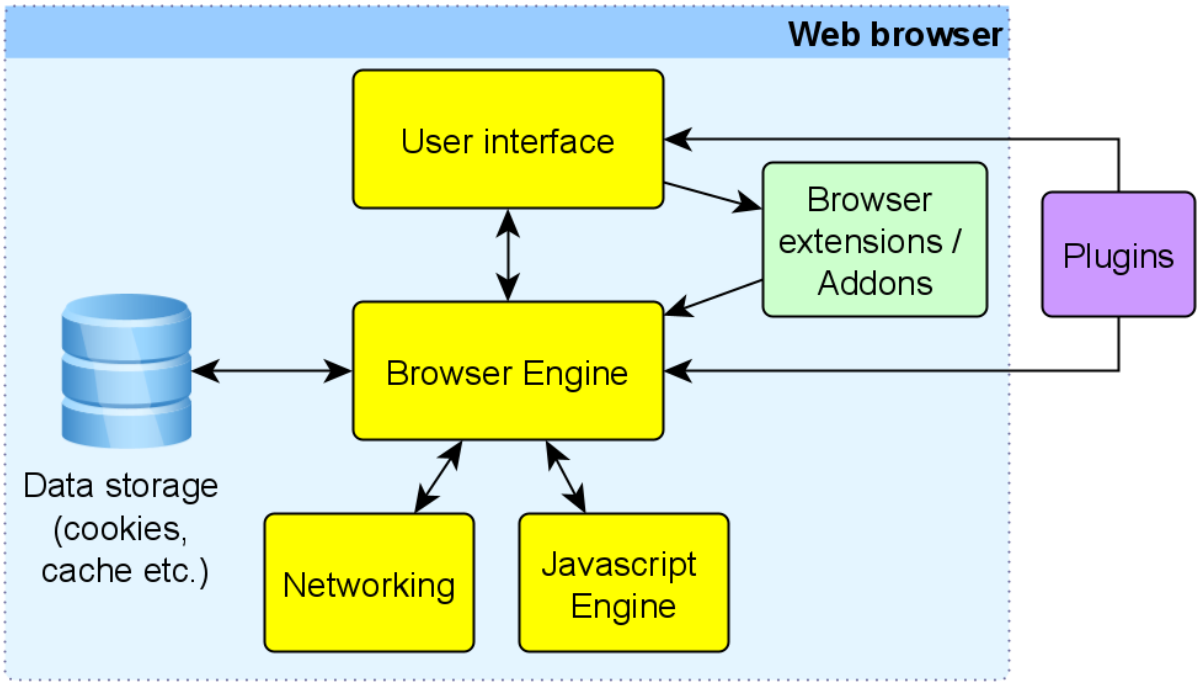


Figure 7 *Based on https://codeburst.io/how-browsers-work-6350a4234634*

22 Alcorn, Wade; Frichot, Christian; Orru, Michel (2014): The Browser Hacker’s Handbook, p. 12
23 Ibid., p. 7

However, these browser engines do not always abide by W3C standards. Because of that and other differences, browsers don't display website content in precisely the same way[24]. These rendering differences can be minor to major, depending on the website. For instance, the web artwork www.scrollbarcomposition.com (2000/2011) by Jan Robert Leegte uses frames (windows within a browser window) and scrollbars as design elements. Since each web browser has its own scrollbar design, the website looks different within each browser. Another example is "div. [property]", https://x20xx.com/div_property.html, (2013) by the Dutch internet art collective Jodi. The site is a Javascript animation of two images, which is rendered differently on different browsers (s. Figure 8 and Figure 9). The case study "Abstract Browsing" in chapter 6.8 provides another example of rendering differences between browsers.

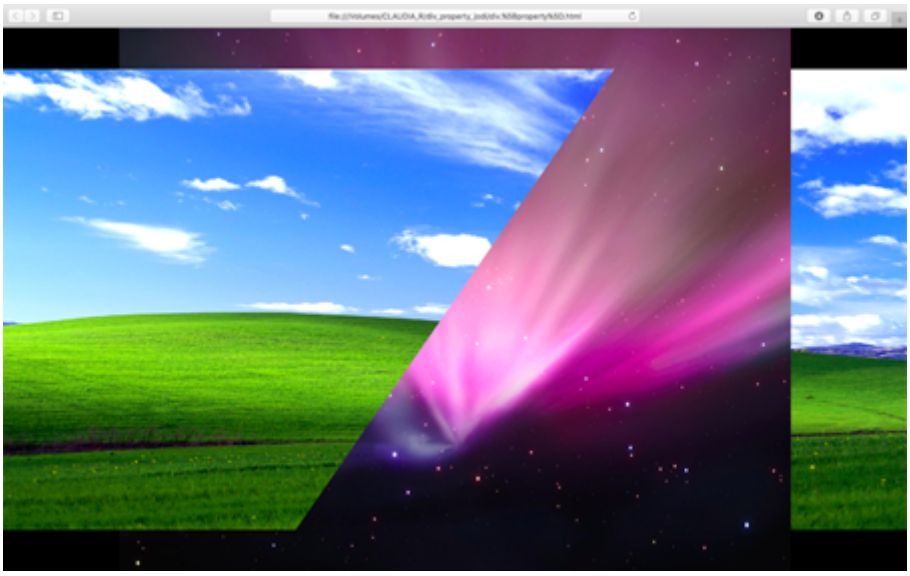


Figure 8 "div. [property]», Jodi (collection House of Electronic Arts Basel), viewed with Safari 10.10.02 (MacOS 10.11.6, about 2016).

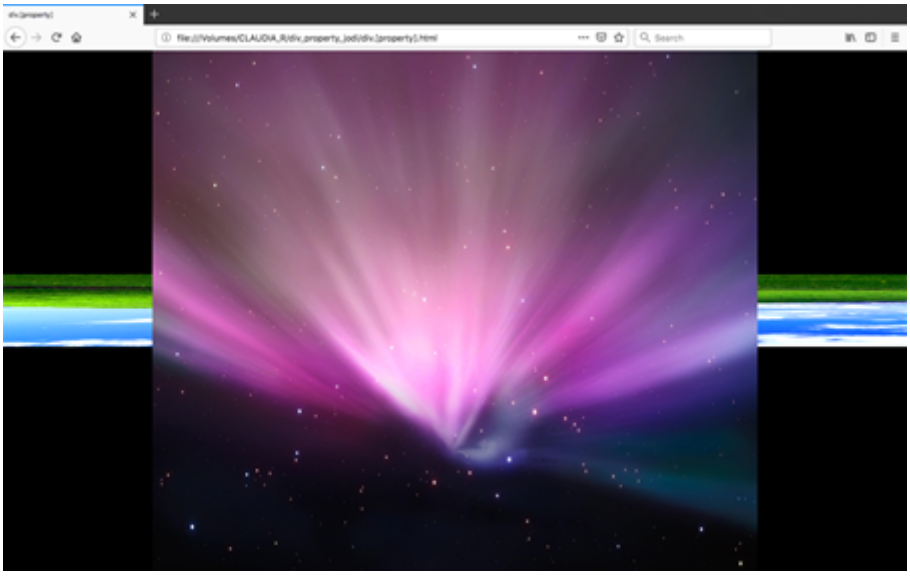


Figure 9 «div. [property]», Jodi (collection House of Electronic Arts Basel), viewed with Firefox 66 (Mac OS10.11.6, about 2016). The artists prefer the Firefox browser for this work.

24 https://crossbrowsertesting.com/blog/test-automation/history-of-web-browsers/ accessed 2020/08/11

5.3. Browser APIs

Application Programming Interfaces (commonly called APIs) are important for the functionality of the web browser. They enable web browsers to work with external programs by offering an interface for communication and integration between the web browser and program.
For example, the support of web audio and web animation in web browsers was solved through the integration of external plug-ins. For the integration of these plug-ins, specific browser APIs such as NPAPI for Firefox (discontinued), PPAPI for Chrome (discontinued from 2021), and ActiveX for Internet Explorer (discontinued) were made available. Due to the integration of web animation rendering in HTML5, these browser APIs became obsolete. Browser APIs also enable the integration of hardware such as webcams.

5.4. Browser versions

The most common web browsers—like Internet Explorer, Edge, Google Chrome, Safari, and Firefox—issue new versions with updates about every month. They are updated automatically by default (unless the user opts out of this in their browser settings). Many updates are security related, while some introduce new features, correct bugs, or change APIs. The user downloads a browser version as an executable for the corresponding operating system. There is no additional software necessary to run the executable.

Firefox, an open source browser, has had about 350 releases as of Spring 2020. These versions are documented at https://www.mozilla.org/en-US/firefox/releases/ within the categories “New,” “Fixed,” “Changed,” “Developer,” and “Unresolved.” It is compatible with Linux, Mac OS, Windows, and other operating systems. The old versions can be downloaded at an official Firefox archive: https://ftp.mozilla.org/pub/firefox/releases/.

In May 2020, Google Chrome is at version 85[25]. Chrome is developed for Linux, Windows, Android, and MacOS. In contrast to Firefox, there is no official Google Chrome browser version archive from Google, as Google does not want users to install outdated versions. Neither are old versions available in the Internet Archive. The closed source Google Chrome browser is based on the open source Chromium project whose versions are available in a project archive[26]. There are some private archives[27] hosting old Chrome browser versions, but there is no guarantee that they are free of viruses, sustainable, or authentic. It would be necessary to gather and preserve one’s own Google Chrome versions oneself or in conjunction with other institutions.

Internet Explorer is an obsolete, closed source web browser developed by Microsoft. Some of its Windows-compatible versions (IE7 to IE11) can still be downloaded on the Microsoft website[28]. Older versions are difficult to get hold of. For Internet Explorer versions 2 to 5, there was also a MacOS

25 Google Chrome version history: https://en.wikipedia.org/wiki/Google_Chrome_version_history

26 For Chromium (which is not the same as Chrome browser) Linux/Win/Mac: https://raw.githubusercontent.com/Bugazelle/chromium-all-old-stable-versions/master/chromium.stable.json (from version 37), accessed July 2020

27 For instance https://www.slimjet.com/chrome/google-chrome-old-version.php (from version 48) or https://superuser.com/questions/988243/where-to-download-the-last-chrome-version-with-java-npapi-support (Chrome version 44, works with NPAPI plug-ins), accessed July 2020

28 Download old versions of Internet Explorer here: https://www.microsoft.com/en-us/search/DownloadResults.aspx?FORM=DLC&ftapplicableproducts=%5e%22AllDownloads%22&sortby=+weight&q=Web+browsers%E2%80%8B accessed July 2020

compatible version, and for Internet Explorer 4 and 5 a Solaris version (UNIX). The current Safari browser (Safari 14) provides a simulation of Internet Explorer 7 and higher, as well as Edge,[29] in order to enable web programmers to check the rendering of a website in these browsers.

The plethora of browsers and browser versions makes it difficult to keep an overview of changes regarding their rendering properties and regarding their compatibility with plug-ins and add-ons.

5.5. Browser extensions / add-ons

Browser extensions or add-ons are part of the web browser (in contrast to plug-ins, which will be discussed in the next section) that extend the browser functionality, for instance by providing integration with specific websites or adding functions such as grammar checks or video download. In the 2000s, companies competed with each other to install their toolbars and widgets in the users' browsers. An add-on can also be a work of art. The latter is described as a use case in section 6.8. Add-ons are made with the same technology (Javascript, HTML, CSS) as a website. They are not necessarily portable across web browsers, as they need to use the web browser's specific API. For instance, the add-on "Abstract Browsing" (2014) by Rafael Rozendaal is made for Google Chrome and cannot be installed in Firefox.
Anybody can develop an add-on. For each browser, there are thousands of add-ons. Sometimes these add-ons are automatically installed by a software as part of a software bundle, while other add-ons are directly downloaded by the user. Add-ons can also compete with each other, sometimes uninstalling a competitor's add-on.
Old versions of add-ons are usually not available. Browser developers only provide the most current add-on version. More information about the availability of Firefox and Google Chrome add-ons can be found in Annexe 1.Annexe 1Annexe 1Annexe 1Annexe 1

5.6. Browser plug-ins

A plug-in is an executable that can be written in any programming language and runs independently from the web browser.[30] In the 1990s and 2000s, plug-ins were very popular, especially on gaming, music, and video-culture websites, and many artists created web-based artwork with animations and interactive or 3D graphics based on plug-ins. Plug-ins were developed in the 1990s because at that time the interactive parts of websites or embedded media could not be encoded in HTML or Javascript; HTML could only create static websites. Specialised companies such as Macromedia Director, Oracle, and Adobe created third-party plug-ins that could play back a website's embedded media.

Once the plug-in is installed, websites with certain features (embedded object tags or MIME-type headers) will trigger the browser to run the plug-in (Alcorn et al. 2014, p. 12). This allows files that would otherwise be run in an external program such as a video player to be rendered by the plug-in within the browser.

Plug-ins have to be updated frequently in order to close security gaps. According to an examination carried out by AV-TEST[31] in 2013, "Adobe's Reader and Flash and all versions of Java are together

29 In advanced settings of Safari, select "Developer mode." In the "Develop" menu of Safari select "User Agent." It opens a dropdown list of internet explorer versions.

30 https://en.wikipedia.org/wiki/Plug-in_(computing) accessed 2020/05/27

31 https://www.av-test.org/en/news/adobe-java-make-windows-insecure/ accessed 2020/05/27

responsible for a total of 66 percent of the vulnerabilities in Windows systems exploited by malware." For that reason, but also to make the web more accessible and less dependent on third-party software, the World Wide Web Consortium[32] developed HTML5 that is able to encode video. With the advent of HTML5 in 2014 and the further development of Javascript, browsers have stopped supporting plug-ins. This replacement of plug-ins with HTML5 and Javascript required a major programming effort, and it has had some unfortunate consequences: many websites that were based on plug-ins are now malfunctioning in current browsers. Browser emulation can be a viable way for such websites to be accessed.

Here are some examples of commonly-used plug-ins:

Plug-in	Description
Video and sound players, video download	• RealPlayer (RealNetworks): 1995– ?. In 2020, the RealPlayer can be downloaded as an independent application, but no longer as a browser plug-in or add-on. • VLC-web plug-in (VideoLAN): In 2020 the plug-in is no longer necessary. A video file format can be assigned to a specific video player, such as VLC, in the browser settings. The VLC player needs to be installed on the operating system level.
Video playback, games, interaction	• Java (Sun Microsystems, from 2010 on Oracle) 1995–2016. Discontinued. • Shockwave (former Macromedia Director, after that Adobe): discontinued between 2017 and 2019 • Flash (former Macromedia Director, after that Adobe): End of life December 2020 for all browsers[33] • Silverlight (Microsoft)[34]: Internet Explorer 11 supported until October 2021, Firefox discontinued since 2016, Chrome discontinued since 2015 • Quicktime (Apple): End of life since 2018
Text (pdf)	• Adobe Reader web plug-in (Adobe): In 2020 the plug-in is no longer necessary. A pdf can be assigned to Adobe Reader in the browser settings. The Adobe Reader needs to be installed on the operating system level.

It is very difficult to locate data on the use of browser plug-ins from the late 1990s until today. This history of plug-ins has almost disappeared. We could find only one graph showing plug-in use in 2011, the Figure 10 below:

32 Mission of the World Wide Web Consortium: https://www.w3.org/Consortium/mission

33 https://blog.mozilla.org/futurereleases/2017/07/25/firefox-roadmap-flash-end-life/ accessed 2020/06/03

34 https://www.microsoft.com/getsilverlight/Get-Started/Install/Default accessed 2020/06/03

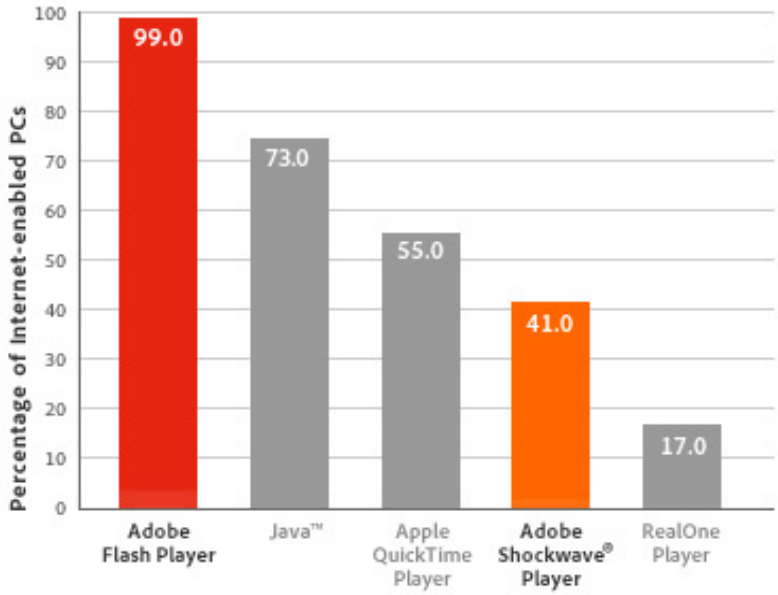


Figure 10 Statistics of plug-ins used in 2011 (source: Adobe[35]. Millward Brown survey 2011). It includes plug-ins of Adobe and its competitors.

Today, plug-ins are made available by the private companies that developed them. Now that plug-ins are obsolete, there is a risk that these companies will take down their archives. More information about the availability of and differences among the Java, Shockwave, and Flash plug-ins can be found in Annexe 1. Two of these popular plug-ins are also explored in the case studies in section 6. There it will become evident how important they are for accessing certain websites.

5.7. HTML5: rendering of video, audio, and 3D animations

Since the introduction of HTML5 in 2014, HTML has supported video and audio tags, which enable the rendering of video and audio without external plug-ins. HTML5 also introduced webGL,[36] a Javascript API that supports hardware-accelerated 3D rendering without plug-ins.

5.8. Transition from HTTP to HTTPS

HTTP is an information transport protocol used when a browser loads and interacts with a website. With HTTPS, the information exchanged between the client and the web server is encrypted. However, nowadays the great majority of websites are HTTPS secured (encrypted), and older versions of browsers do not support HTTPS. This is relevant when setting up a browser emulation. The websites of browsers like Firefox, Chrome, or of plug-in providers are HTTPS secured, so they cannot be accessed with old browser versions. For instance, in Windows XP, it is not possible to access HTTPS websites with the standard Internet Explorer 6.0, even if the internet can be

35 http://www.adobe.com/products/shockwaveplayer/shockwaveplayerstatistics.html accessed on the wayback machine of the Internet Archive (date: Feb 19, 2013).

36 https://en.wikipedia.org/wiki/WebGL

accessed.[37] This means that a browser version has to be downloaded from a fairly new web browser and then transferred to the obsolete operating system.
On the other hand, old websites, such as the case studies discussed in section 6, are usually not encrypted. If they are opened with a current browser, this browser will ask for a security certificate, and the user will have to confirm that it is ok to open the website without this certificate. This browser behaviour is already a policy in Google Chrome and it will become increasingly difficult to open non-encrypted websites with current browsers. In other words, it will become more and more difficult to view HTTP websites outside of browser emulations. Dragan Espenschied's tweet underlines that issue (s. Figure 11).

Figure 11 Tweet of Dragan Espenschied 19 Nov 2020 about the introduction of HTTPS-Only mode by Firefox

5.9. Caching

A web cache is a temporary storage for website elements. It increases a website's performance by storing data locally in a cache so that the browser does not have to request the data from the server. One important precondition for this is that the website had been opened with the same browser before. Caching behaviour is relevant for the testing of websites, when it might be useful to delete the browser cache first in order to prevent it from loading the website from local storage. The cache can be cleared in the browser settings.

5.10. Browser configuration / policy

Browsers can usually be configured in much more detail than just in the settings. A sophisticated browser configuration can be useful to regulate browser emulation access at archives or in exhibitions. More details are described in Annexe 1.

37 https://en.wikipedia.o>rg/wiki/Transport_Layer_Security lists which browser versions support which protocol (SSL protocol version, TLS protocol version, certificate support, vulnerabilities).

5.11. Chapter summary

This chapter roughly summarised the technical development of web browsers and the effect this has had on the representation of websites. It also briefly explained the complex ecosystem of browsers and their components. In the following chapter, the effect of browser development on the representation of websites and on the preservation of web browsers will be illustrated and discussed by means of case studies.

6. Case studies

We selected several websites to use as case studies, testing them on different web browsers in order to learn how the significant properties of a website are supported by different web browsers and how the web browser technologies described above affect the browser emulations. These case studies also informed the description of the browser technology above regarding their relevance for preservation.

6.1. Selection of case studies

Originally, two websites were selected as case studies: http://www.collapsus.com/, and www.wefeelfine.org. They were selected as representatives of Dutch web culture and for their dependency on different plug-ins. Although we knew that only two websites could not expose all of the features of a web browser, we hoped these sites would be useful examples that would help us develop a method to identify browser properties and behaviours. However, as we worked, our initial website selection had to be modified.

The website www.collapsus.com, created by the Dutch animation production company Submarine in 2010, uses Flash video and Javascript to create an interactive experience. This website turned out to have problems that could not be solved with an emulation. Due to these problems, another website was chosen: www.wishingtree.nl. Wishingtree.nl (2002–2012) also uses a flash plug-in and can (still) be rendered by current web browsers. This has the advantage that its significant properties can be assessed without emulation.

In order to test a use case of an obsolete website that cannot be viewed correctly in current web browsers, www.wefeelfine.org/ from 2006 was chosen as a case study. It needs a Java plug-in to be rendered. After several tests this website also turned out to have a problem that could not be solved with emulation, so another Java-based website, www.tebatt.net, was introduced as a case study.

These websites (www.collapsus.com/, and www.wefeelfine.org) with “problems,” or “damages” as a conservator would call it, were kept as case studies as they illustrate typical challenges faced when working with obsolete websites.

The fifth case study is a browser extension called “Abstract Browsing” by Dutch artist Rafael Rozendaal (2014). This site is an example of how artists can use browsers in unexpected ways.

6.2. Online website versus archived website

As members of the NDE network also host web archives, it would have been useful not only to test online websites (described in section 4.1) but also archived websites. Archived websites are websites that have been crawled with a web crawler at one or several distinct times. Web crawlers such as

Heritrix intercept the traffic between the web server and the client and store it in a specific format.[38] They do not capture the scripts running on the web server. In other words, the crawler makes an essentially static website out of a dynamic website. Such archived websites need a specific software to render them. The web browser alone is not able to render a web archive.

The most common web crawlers crawl independently from web browsers. Browser-specific behaviour thus cannot be captured and rendered. However, the web archiving tool that was designed by Rhizome is different. This tool, Conifer,[39] records the website traffic through the web browser and is able to capture and play back websites with user interaction and plug-in-streams such as flash videos. This enables a more authentic playback of the archived website. Conifer runs on a current operating system and integrates browser emulations in the form of docker containers. However, at the time of testing, the EaaS framework and Conifer were not integrated in order to combine web crawling with custom-made browser emulations. This is the reason why archived websites were not tested for this research. Therefore, the browser emulations set up for the tests outlined below were only tested with online websites.

6.3. Web browser testing

The websites were tested in different browser environments in order to bring forward the properties of web browsers and their development. We expected that the websites would look different when viewed with different web browsers or different browser versions, due to the different browser engines (for more on this, see section 4.2 above). We primarily tested different versions of Firefox, as its versions are easily available, in contrast to Google Chrome or Internet Explorer, although we also tested two versions of Internet Explorer in order to compare them to Firefox.[40]

The websites we tested were all created, maintained, or restored between 2010[41] and 2012. Thus, the years between 2010 and 2019 were chosen as a test period for browser and plug-in versions.

As wishingtree.nl needs a Flash plug-in, and wefeelfine.com a Java plug-in, the relevant Flash and Java versions were downloaded. The choice of operating system was rather random: Windows XP (SP3) was a preinstalled environment in the Emulation as a Service (EaaS) platform. Thus, the following environments were set up:

Year	OS	Firefox	Flash	Java
2010	Windows XP	Firefox 3.0.5 Internet Explorer 6	Flash player 10.1.102.64	Java 6.18
2013	Windows XP	Firefox 25 Internet Explorer 6	Flash player 10.3.183.90	Java 7.11
2016	Windows XP	Firefox 44 Internet Explorer 6	Flash player 11.6.602.171	Java 8.72

38 The usual file format of a crawl is a WARC-file.

39 Espenschied, Dragan; Kreymer, Ilya (2016): Workshop: Symmetrical Web Archiving With Webrecorder. iPres2016. https://zenodo.org/record/1255965#.Xv86uOdS8uU

40 Internet Explorer 11 is part of the operating system Windows 7 and Internet Explorer 6 of Windows XP. As the Internet Explorer versions are quite neatly tied to the Windows operating system and used to be upgraded directly, it was too time consuming to install other IE versions. Advanced operating system knowledge is necessary for that and/or the right Windows service pack.

41 The Java-based www.wefeelfine.org website was still actively maintained in 2010.

2019	Windows 7	Firefox 66 Internet Explorer 11	Flash player 32.0.0.192	most current Java version (1.8.0.251)

The EaaS emulation framework was used to set up the web browser environments. Within that framework, Windows XP was emulated with the emulator qemu. Windows 7 was installed on the virtualisation platform Virtualbox.

To compare the behaviour of a website shown in different browsers, we applied criteria such as movement and patterns in animations, the sound and position of website objects, reaction to key or mouse input, and functionality in general. Screenshots and screencasts were made for comparison. These screencasts and live emulations were visually compared next to each other to determine differences between browsers, but these observations were not quantified.

6.4. Results: www.collapsus.com

The flash-based website Collapsus.com (created in 2010) was a revolutionary interactive documentary created in order to attract a young audience for television in the Netherlands (s. Figure 12). Submarine and VPRO, a Dutch TV channel, co-produced it. It was made with top-notch web technology and was nominated for a Digital Emmy award. Two documentary videos[42] of the website were made, which give an impression of how the website should look.

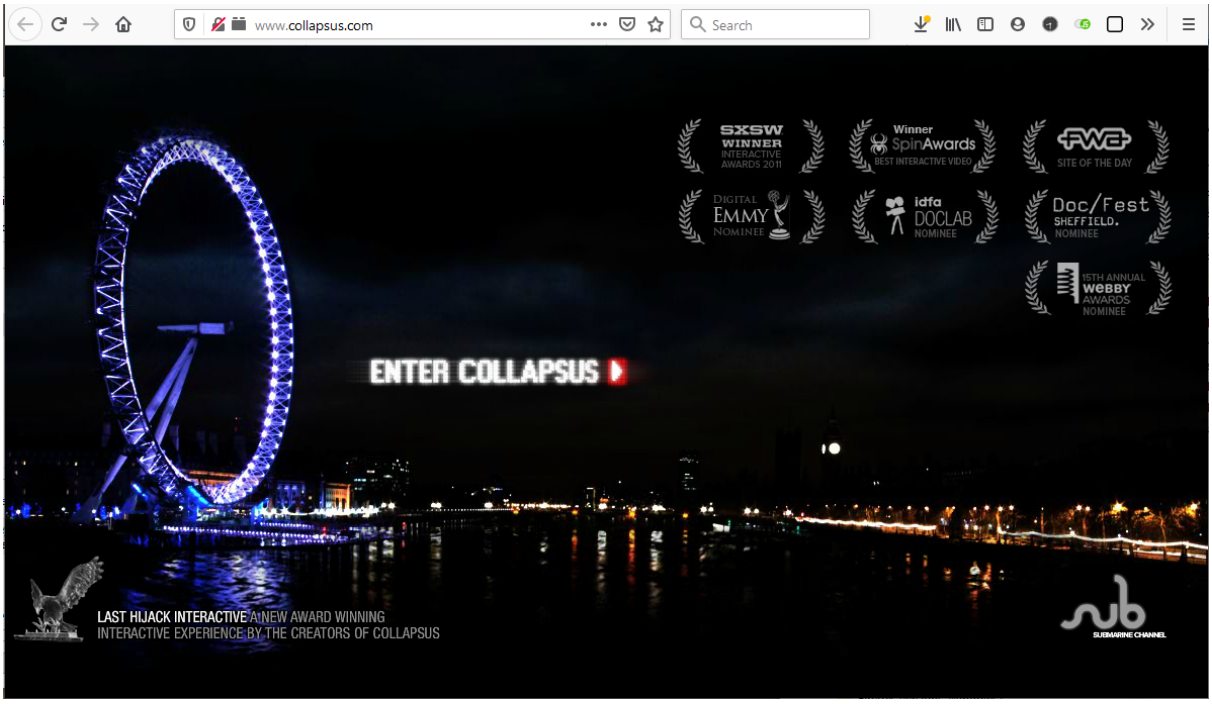


Figure 12 Collapsus.com in a current browser (Firefox 77)

42 https://vimeo.com/15396143: Tommy Pallotta, the creator talks through the interactive documentary
https://submarinechannel.com/profiles/profile-creator-producer-collapsus/ about the creators of collapus

After some testing in different browser environments, it turned out that it was not possible to run the website correctly: When clicking on “ENTER COLLAPSUS”, the Flash movie did not play back (s. Figure 13).

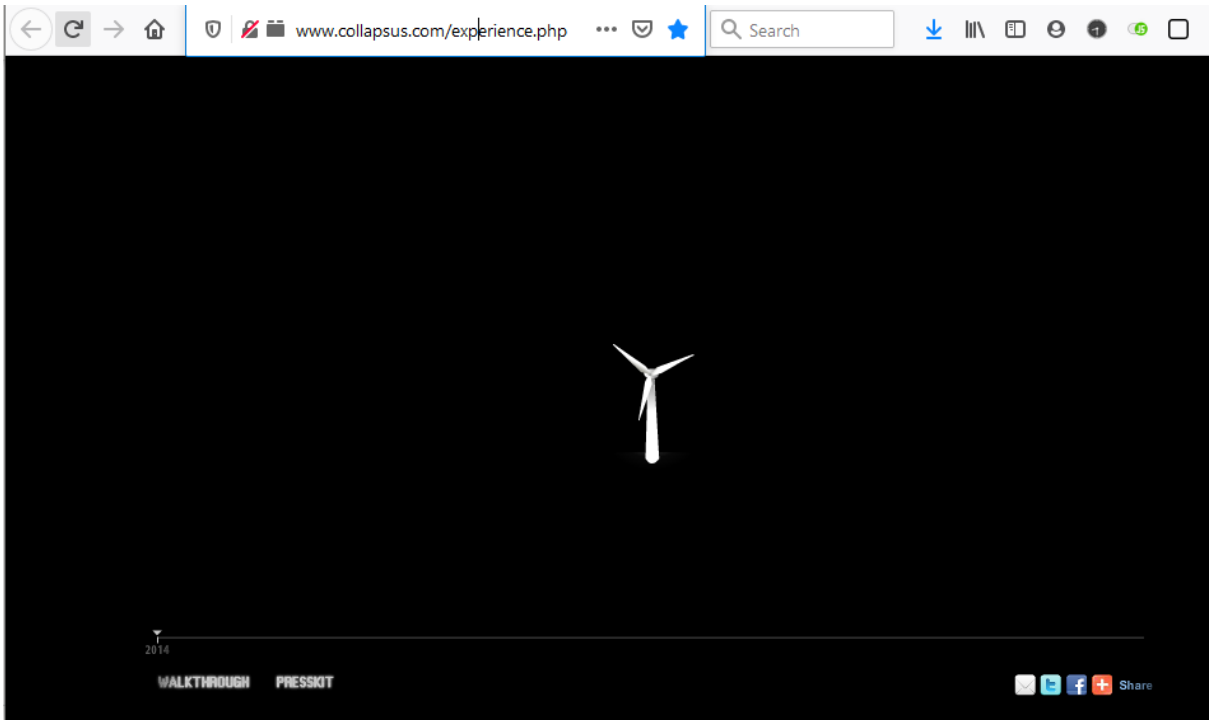


Figure 13 www.collapsus.com/exerpience.php: Flash movie not playing

After investigating the website itself, we found that it has an external dependency: the Flash video is streamed from a Flash server. The best way to see that was by using the Chrome developer tools, which revealed that www.collapsus.com/experience.php was requesting http://streams.collapsus.com/fcs/ident2. This website is probably a Flash streaming server that does not exist anymore (s. Figure 14). Strangely, this is visible in the console of the Chrome developer tools, but not visible in the Console of the Firefox developer tools.

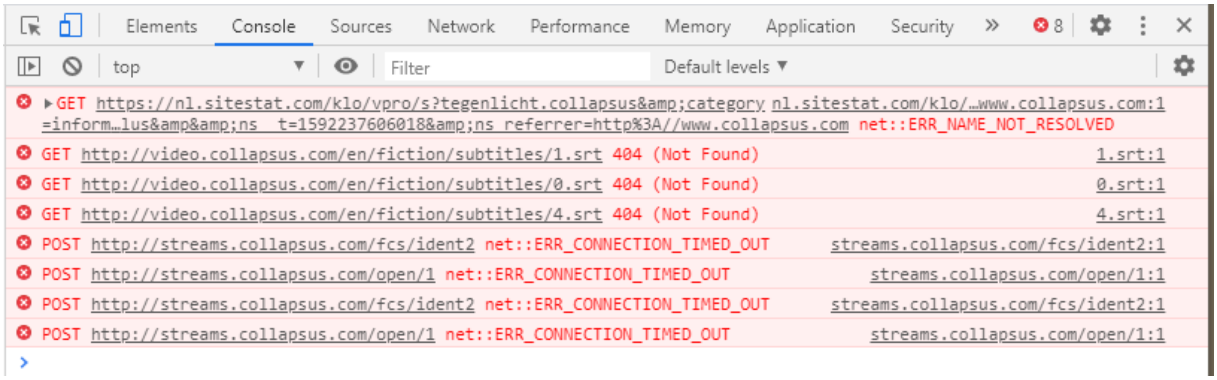


Figure 14 Chrome developer tools (Console)

This broken external dependency lead to the conclusion to drop this case study, as its significant properties cannot be perceived, no matter what web browser is chosen to render the website. The problem can only be solved server-side, in collaboration with the operator of the website. This case study shows that obsolete websites can have underlying problems that only become apparent when

working with them in greater depth. Unfortunately, it is a typical example of lost digital heritage if the Flash video is not recovered and made accessible in the future.

6.5. Results: www.wishingtree.nl

The collapus.org case study was followed by the www.wishingtree.nl case study. The artists Elsa Stansfield and Madelon Hooykaas created www.wishingtree.nl in 2001–2002. Wishing Tree is a spiritual work. The leaves of a beautiful tree embody the wishes of the people who interact with the website. The users can add new wishes to the tree. When they click on a leaf, the wish appears as a text above the tree and a spherical sound can be heard. The website was restored using Flash technology in 2012 with the help of Jaap van der Kreeft, in order to present it at an exhibition.[43] The website can still be viewed in current browsers with the Flash plug-in enabled (s. Figure 15), but Flash will be discontinued by the end of 2020. Thus, starting in 2021 users will no longer be able to experience wishingtree.nl unless it is translated to contemporary technologies, or emulation is employed.

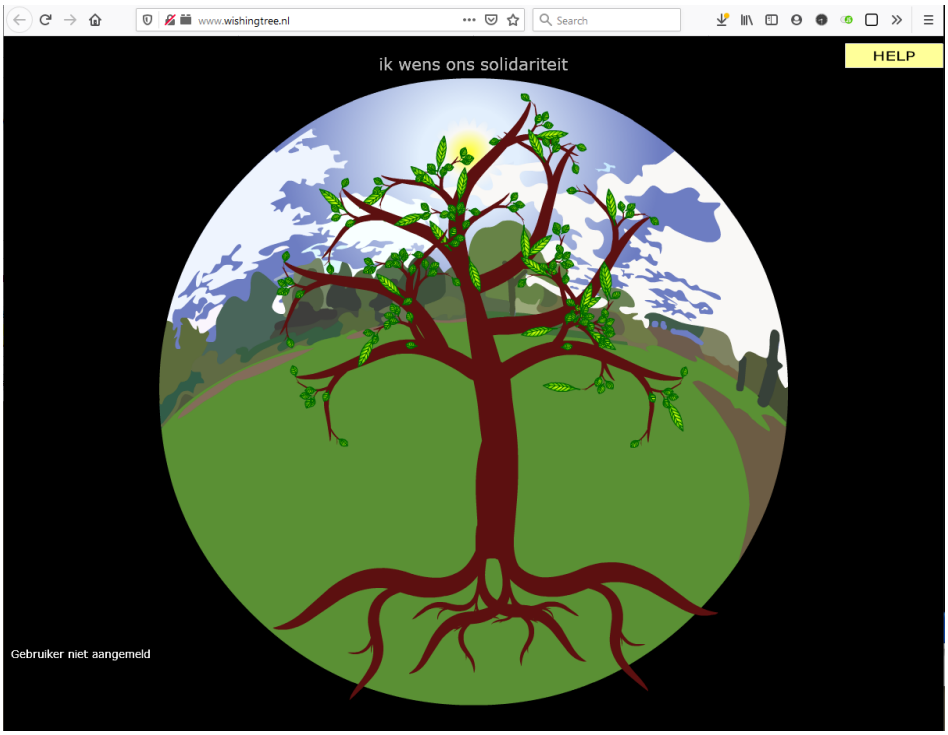


Figure 15 www.wishingtree.nl in Firefox 77, Shockwave Flash 32.0 and Windows 10.

43 Museum voor Religieuze Kunst, Uden (NL), 16 juni - 11 november 2012

Figure 16 www.wishingtree.nl in Firefox 44, Shockwave Flash 11.6 and Windows XP.

Figure 17 www.wishingtree.nl in Internet Explorer 6, Shockwave Flash 10.1.10 and Windows XP.

The browser emulation test results are listed in Annexe 2. During these tests it became clear that the environments could be set up without major problems due to the availability of Flash and Firefox. The most challenging part of these tests was preventing browsers and plug-ins we were using from being automatically updated.

Wishingtree.nl was rendered in all the browser environments quite consistently (s. Figure 15, Figure 16, and Figure 17). The interaction with the mouse was a bit slower in the emulation environment than

in the native environment,[44] but that may vary depending on the resources (e.g., RAM or computing power) assigned to the emulation. The sound was played back well in all of the environments. Even the text font of the wish written above the tree was very similar in all the browser versions and the two different browser types (Firefox and Internet Explorer). The position of the web elements was the same in all the environments.

6.6. Results: www.wefeelfine.org

The website Wefeelfine.org was conceived of and designed by the researchers Jonathan Harris and Sep Kamvar in 2006 to explore how human feelings are expressed on the internet. During a few years (until about 2010), they captured a large amount of data about feelings and stored it in a database. The results were presented in an interactive way in a Java applet. On the website itself, the designers informed the user that it is “best viewed in Firefox, Safari, or Internet Explorer” (s. Figure 18). The website needs to be rendered in a browser with a Java plug-in, but these Java plug-ins are obsolete today. The only way to present this website (more or less) correctly is in a browser emulation. Of course, the broken links cannot be restored with a browser emulation.

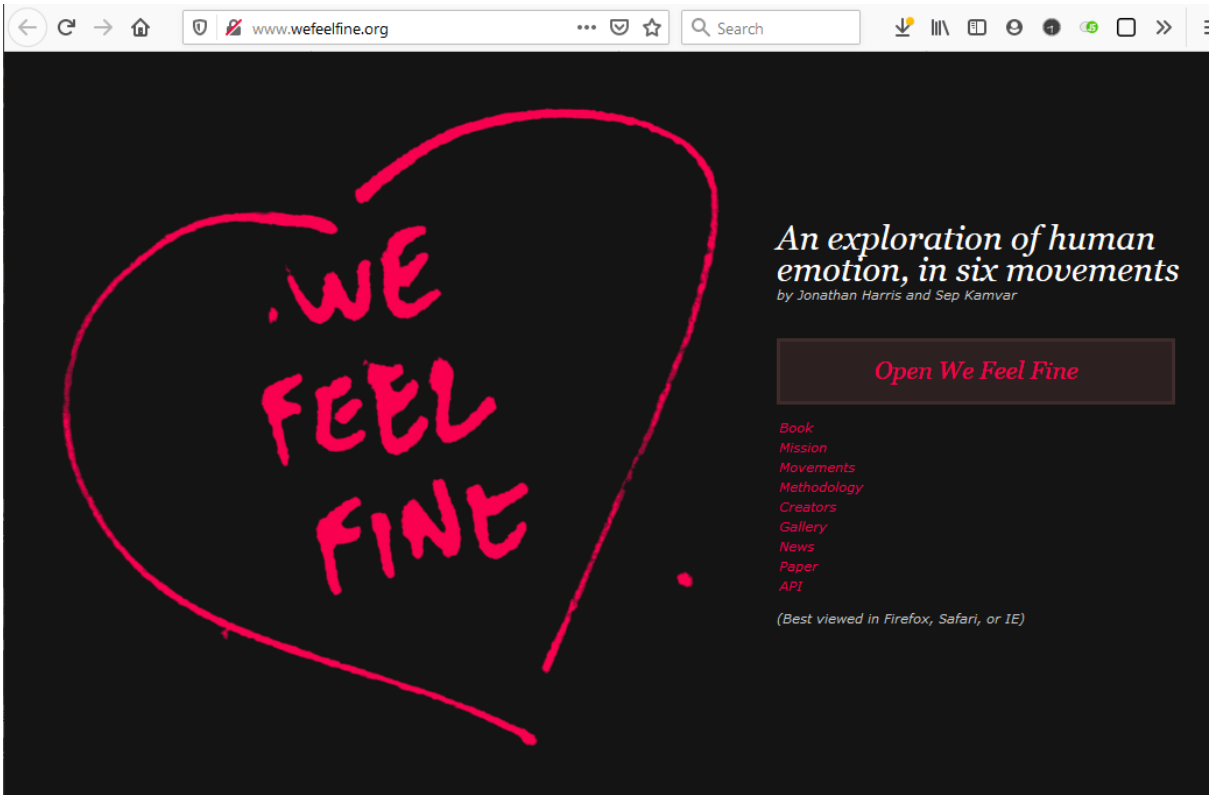


Figure 18 Welcome page of www.wefeelfine.org in Firefox 77 and Windows 10.

To perform all the tests listed in chapter 6.3 was a challenge. The Java plug-in cannot be simply downloaded. Instead, the Java Runtime Environment (JRE) needs to be installed (s. Annexe 1), which automatically installs the plug-ins in Internet Explorer and Firefox. When we attempted this, we experienced problems with the Oracle Java archive (the download did not work for several days). This is mentioned to highlight the dependency from this company archive. After many tests (documented in the Annexe 3), it became evident that there was a problem with the Java animation on the website

44 Operating system directly installed on hardware, no emulation layer

itself. One way we tested this was to download the Java file[45] and open it with the Java Runtime Environment. As it could not be opened with any Java version, it became clear that the Java file itself must be corrupt. An inspection with the Chrome developer tools showed that there were more rendering problems. Much like with www.collapsus.com, the problems need to be solved server-side first, before a browser emulation can render the website.

6.7. Results: www.tebatt.net

Given these difficulties, a different website with Java animations was tested: www.tebatt.net, a website made by the artist Trevor Batten presenting his computer-animated projects. Trevor Batten studied in the UK and came to the Netherlands in 1972 to research, teach, and practice electronic sound and computer-generated art.[46] He worked mostly with Amiga computers to produce his animations before he turned to Java in 2002. “Diagonal Combinations” also called “Grammars” (2004) is one of many computer-generated animations he made in Java (s. Figure 19).

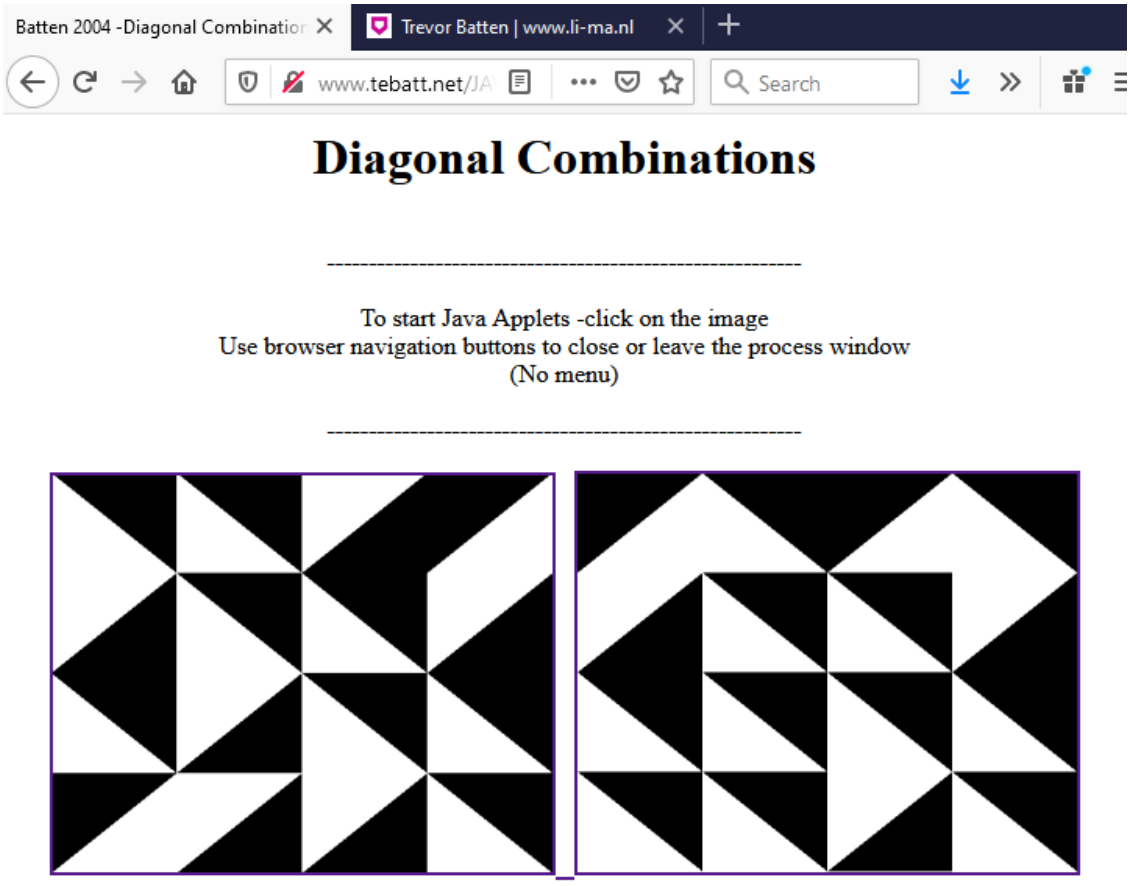


Figure 19 Java-Animation “Grammars” from Trevor Batten’s website[47] (start screen, not animated). Displayed in Firefox 78.

The test results for www.tebatt.net are listed in Annexe 4. The animation “Grammars” could be rendered with the oldest tested Firefox versions 3.0.5 (JRE 6.18, s. Figure 20) and 25 (JRE 7.11) in

45 The file-name can be found in the source code of the website. The file extension is .jar. By adding the filename with the jar extension to the end of the URL, the browser will suggest downloading the file.

46 Trevor Batten’s cv until 2004: http://www.tebatt.net/person.html

47 http://www.tebatt.net/JAVAGALLERY/SKETCHES/ARTICULATION/GRAMMARS/Gram/GramInfo.html

Windows XP. Newer Firefox versions were not able to play back the Java animation. The combination of Internet Explorer 6 and JRE 6.18 could play back the animation (s. Figure 21) in Windows XP. (The differences in Figure 20 and Figure 21 are due to the randomisation of the pattern.) The other tested Java versions did not open with Internet Explorer 6.[48]

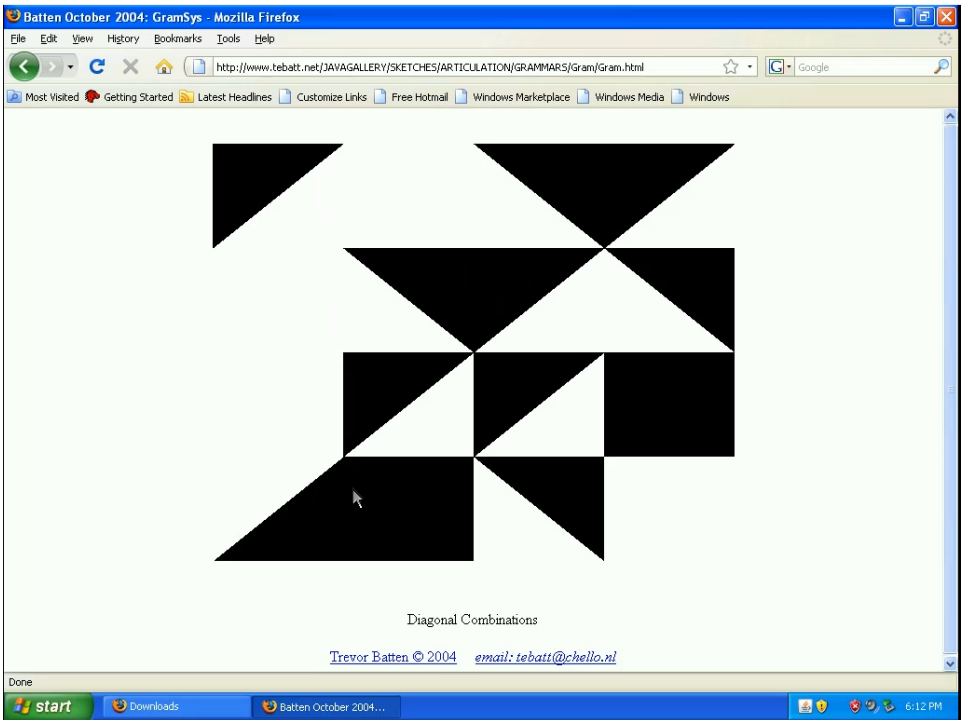


Figure 20 Screenshot Java animation "Grammars" in WinXP, Firefox 3.0.5, Java 1.6.0.18

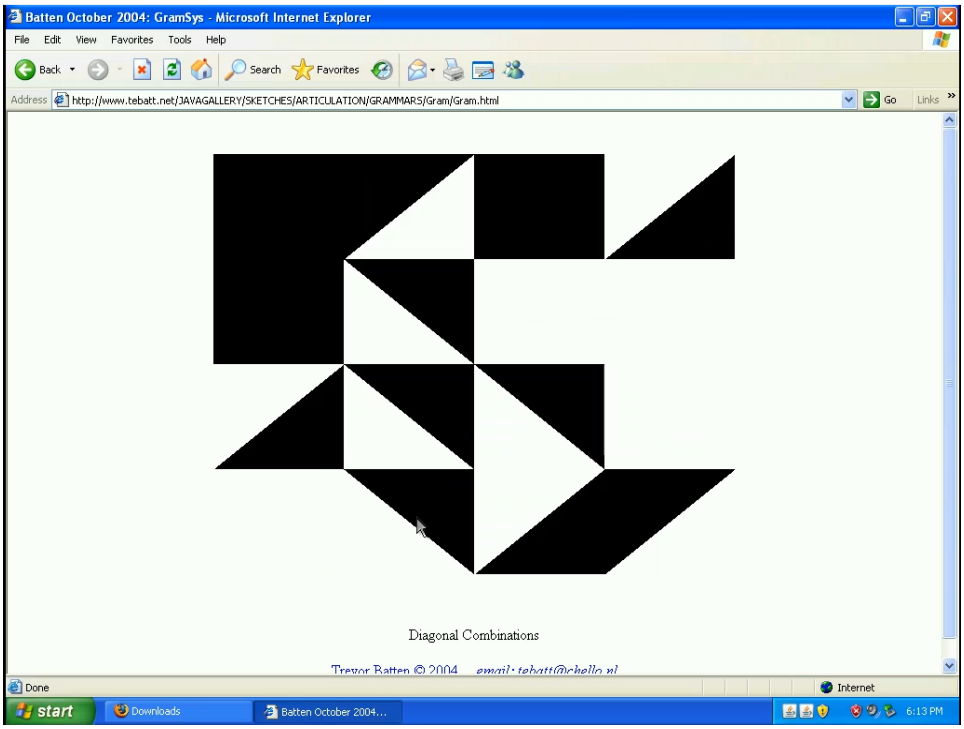


Figure 21 Java animation "Grammars" in WinXP, Internet Explorer 6, Java 1.6.0.18

48 IE blocked them because they were not up to date.

The downloaded Java file of the “Grammar” animation (gram.jar) could be rendered with all the Java versions except for one (s. Figure 22).

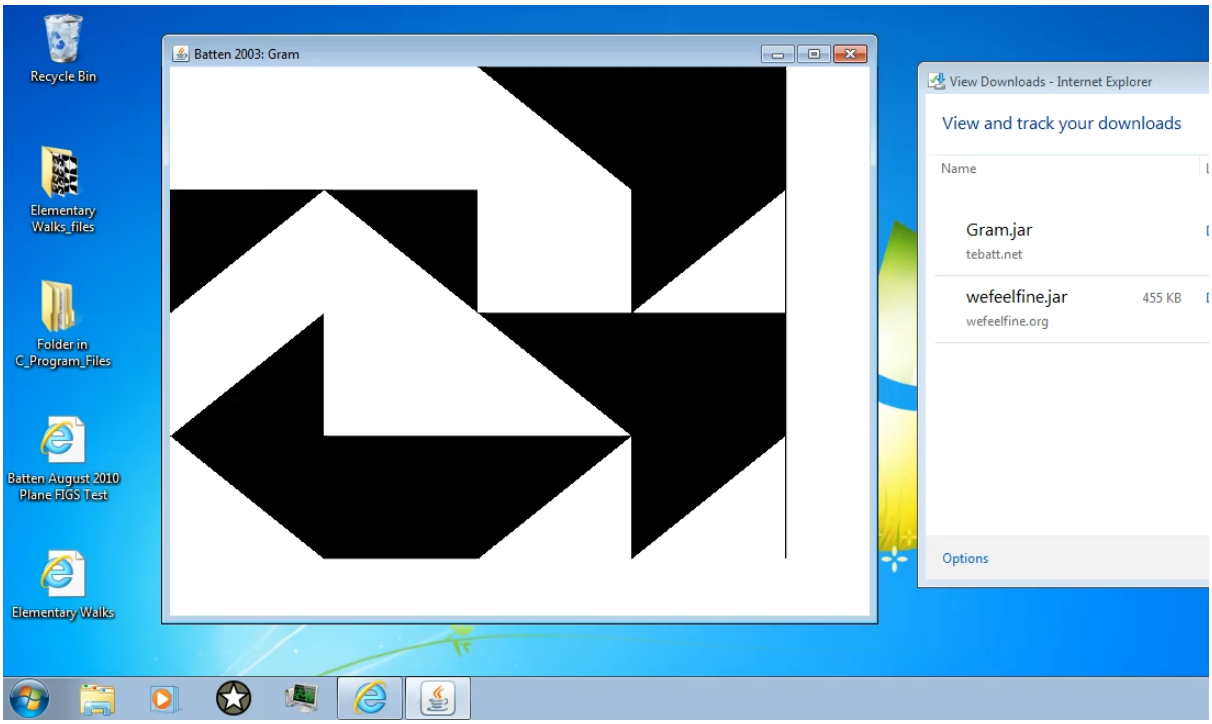


Figure 22 Downloaded Java file of the “Grammar” animation (gram.jar) opened in Windows 7 with Java Runtime Environment 1.8.0.261. The animation could not be played back within a web browser in this configuration.

The browser tried to block outdated Java versions. However, if Java is updated to newer versions, the discrepancy between the old browser version and the newer Java version will become bigger until they are no longer compatible. From the tests above, it seems that older browser versions work better with Java, as they do not have such sophisticated blocking mechanisms. Hence, although in theory there are many combinations possible, in practice not many work. There were no obvious rendering differences observed between the versions of Java. However, detailed comparisons were not made.

6.8. Case study “Abstract Browsing“ by Rafael Rozendaal (2014)

“Abstract Browsing” is an artwork created by the artist Rafael Rozendaal in 2014.[49] “Abstract Browsing” consists of a button on the top right corner of the browser frame. If a user clicks this button while any website is open, it creates a layer of colour patterns based on the visual structure of the opened website and overpaints it, so that the underlying website content is not readable anymore (s. example in Figure 23). Rafael Rozendaal describes it on his website: “It shows you the skeleton of the web. It’s like seeing an X-ray of a building, showing the structural elements.”[50]

49 Description and download link: http://www.abstractbrowsing.net/
50 https://www.newrafael.com/notes-on-abstract-browsing/

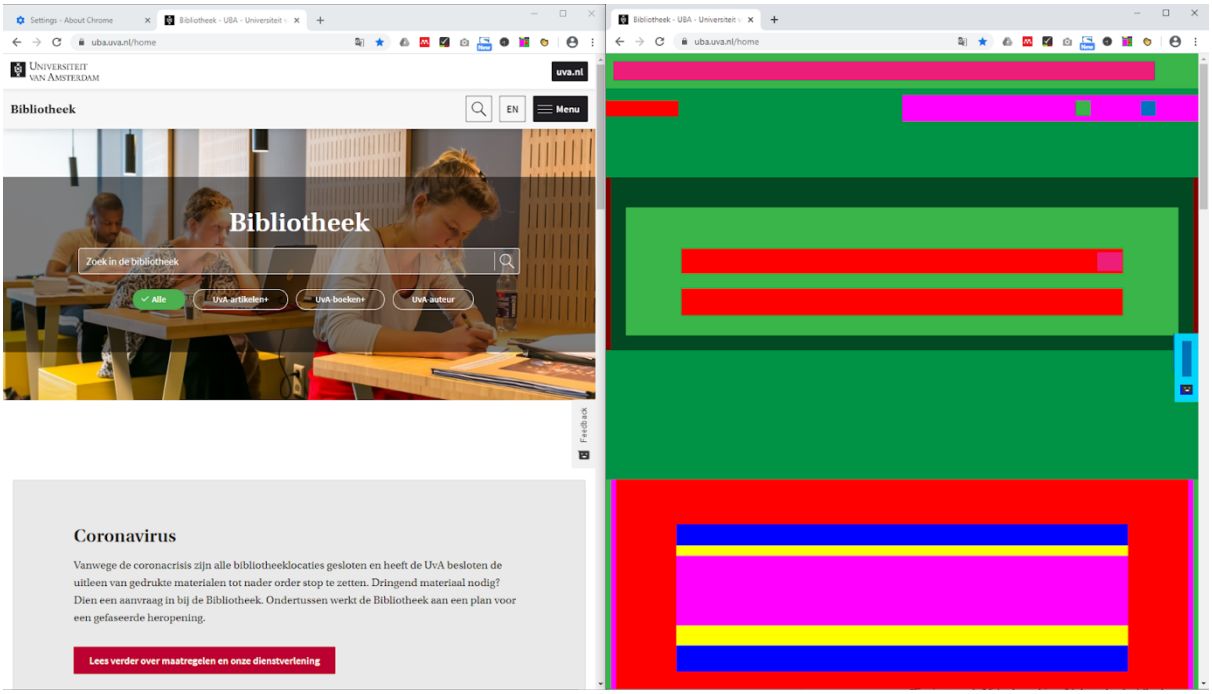


Figure 23 Screenshot of a randomly chosen website on the left and of the applied "Abstract browsing" extension on the right (2020/05/13)

Technically Rozendaal achieves this effect by creating a browser extension or add-on, which is currently available only for the Chrome browser. Rafael Rozendaal has already updated the browser extension several times in order to fix bugs, most recently in 2017 (s. Annexe 5, Rafael Rozendaal's email and s. Figure 24). According to him, this extension works well with new Chrome browser versions without further adaptations.

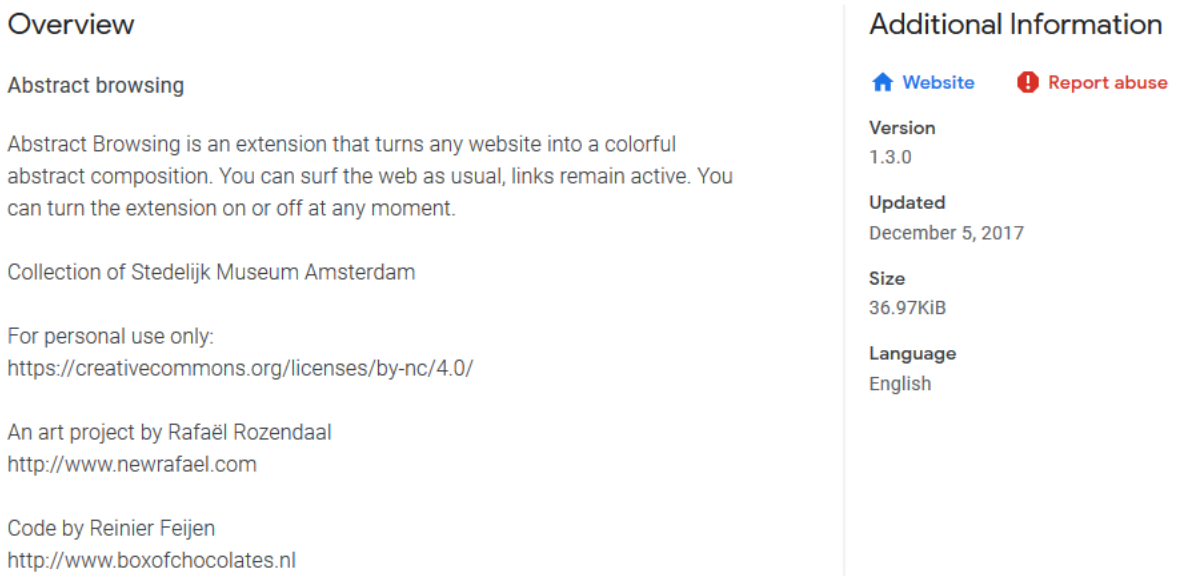


Figure 24 Screenshot Chrome extension "Abstract Browsing" 2020/05/19

The tests[51] we executed with Chrome version 33 (released in 2014, the year when "Abstract Browsing" was released), Chrome version 83 (released in 2020, the most current version) and

51 The chrome browsers were installed in Windows 7.

Chrome version 56 (released in 2017) showed that “Abstract Browsing” version 1.3 could be installed and run in all the browsers without problems.

Rafael Rozendaal developed the add-on only for Google Chrome, which is why no other web browsers were tested. As described in chapter 5.4, there is no official Google Chrome archive. Versions below 48 are difficult if not impossible to source. This is why it is necessary for institutions interested in preserving websites to gather and preserve their own Google Chrome versions. Old “Abstract Browsing” versions could not be downloaded from the Google Chrome web store[52], which is why the tests were only run with the latest add-on version. (The artist himself kept the old versions as zip files, but naturally, they are not publicly accessible.) The add-on needs to be locally installed. The extension is packed in a folder on the local computer which has structured subfolders containing images, Javascript code, and metadata (written in JSON), which the browser interprets on the fly.

“Abstract Browsing” 1.3 ran on all the tested Chrome versions. Whether these versions generate the same look and behaviour is not easy to evaluate, as Rozendaal used a random generator to create the colour compositions. Every few seconds the colour composition changes automatically, so the sequence of these compositions is never the same and the number of possible combinations is huge. The screenshots below (Figure 25 to Figure 28) and in Annexe 6 show how the add-on works.

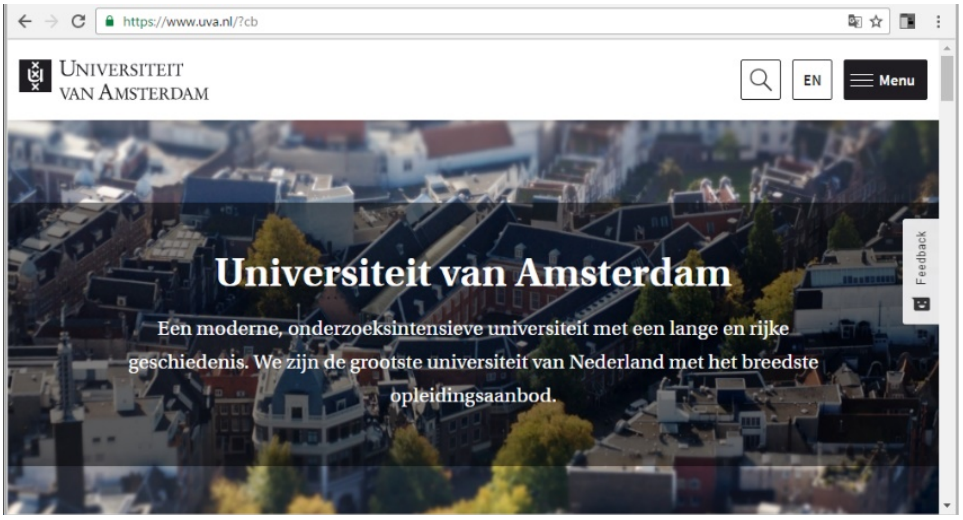


Figure 25 Chrome Version 56. The feedback button on the right window is there.

52 https://chrome.google.com/webstore/category/extensions?hl=en accessed May 2020

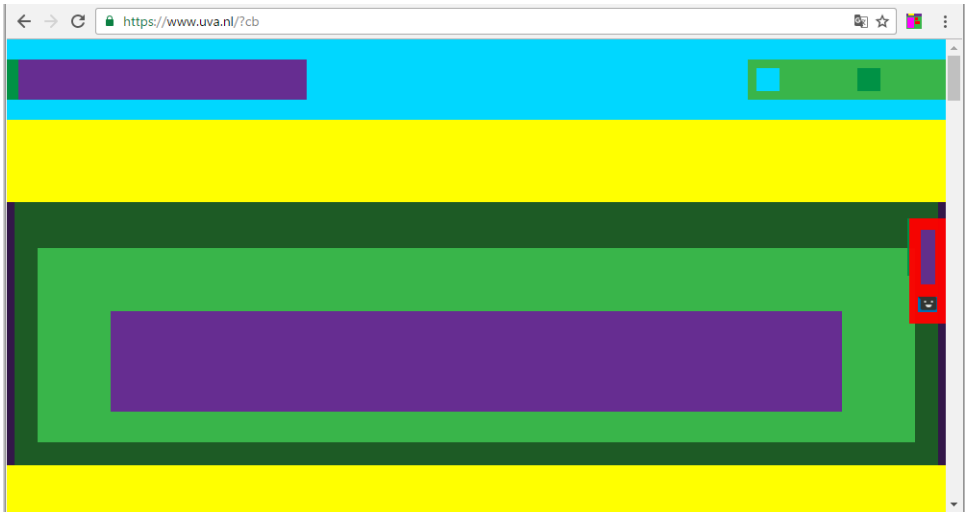


Figure 26 Chrome Version 56. The feedback button on the right window is there.

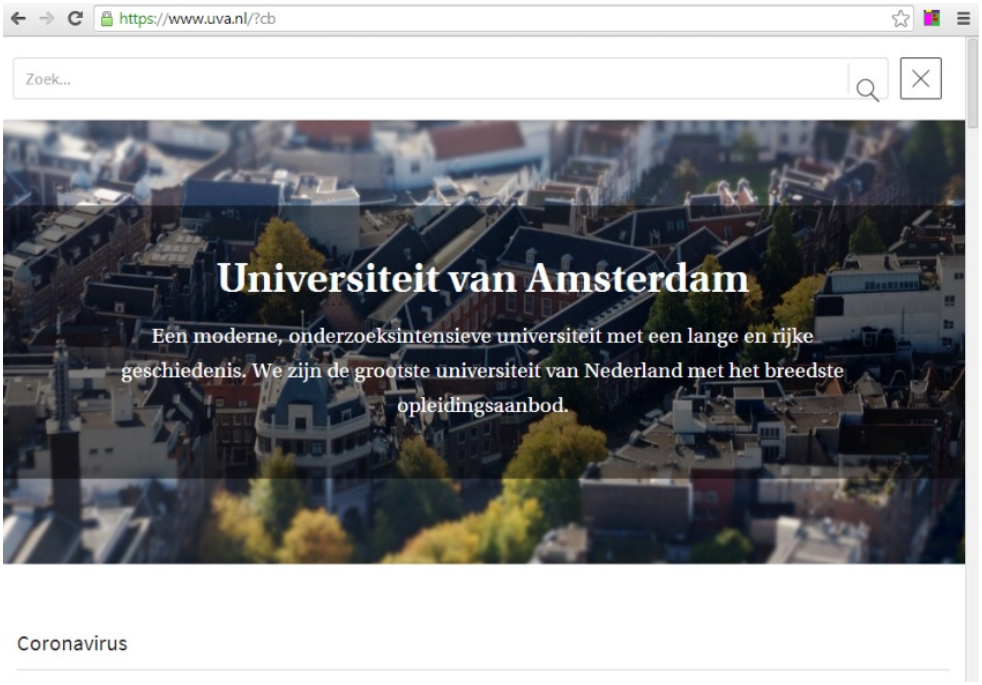


Figure 27 Chrome Version 33. The feedback button on the right window side is missing.

Figure 28 Chrome Version 33. The feedback button on the right window side is missing.

The screen recordings of the three different browser versions suggest that there are no big differences among the three versions. The main difference was found regarding a small tag on the side of the window. Chrome version 33 was not able to render it, hence, it was also not visible in the “Abstract Browsing” mode.

6.9. Conclusion

All the case studies date from 2004 and later—the second half of the first browser war and the beginning of the second browser war described in chapter 4.1. These are the years when web browsers began to conform more closely to the W3C standards. Because of this, the rendering differences between web browsers decreased during these years. This is one of the reasons why the wishingtree.nl and the tebatt.net case study showed so little rendering differences between different browsers and their versions.

If you want to investigate specific browser properties more closely and find websites that do show significant differences in display between browsers, it is important to have deep, practical knowledge of technical and cultural internet history. This kind of knowledge is mostly held by web designers, gamers, and artists who designed websites at that time. They might also know which browser environments were in use at a certain time within certain user groups. Hence, institutions preserving and or researching websites with a focus on website design and functionality (not just on collecting pure information), should collaborate with such experts in order to identify and build the right browser environments to view these websites.

7. Significant properties of web browsers

We will now address the four research questions raised in section 2. A separate chapter will be devoted to each research question. This section will answer the first research question and describe the significant properties of web browsers.

Section 5 of this report described the inner workings and the ecosystem of web browsers. Browsers and their components can be chosen according to their technical properties: browser engine, web API for plug-ins, which HTML versions it can interpret, which encryptions it can handle, which computer architecture it is built for. This information is all publicly available. However, a systematic technical description of each browser type and version does not exist and it is difficult to get an overview. The following properties are relevant for the display of websites:

Technical properties of web browsers, version dependent:

- HTML versions interpreted
- Internet Protocols interpreted
- Javascript versions supported
- Browser API for plug-ins
- Hardware APIs and their versions
- Often-used plug-ins and the formats they can render
- Sound rendering (In the early days this was plug-in dependent): plug-in name, or HTML version[53].
- Video rendering (This was plug-in dependent. Nowadays it is integrated in HTML5): plug-in name or HTML version.[54]
- 3D and virtual reality rendering (This was plug-in dependent. Nowadays it is integrated in HTML5): plug-in name or webGL version.

Another way to choose a web browser is according to how it presents a website—in other words, according to its significant properties. The following **browser behaviours** were deduced from sections 4 and 5:

53 The website https://www.w3schools.com/tags/tag_audio.asp informs about which browser versions support the html audio tag.
54 The website https://www.w3schools.com/tags/tag_video.asp informs about which browser versions support the html video tag.

Behaviours of web browsers, version dependent:

- Positioning of website elements
- Scaling of website elements within browser window
- Look, sound, and movement of web animation
- Reaction to key or mouse input
- Fonts used
- Look of browser and browser elements (scroll bar, mouse pointer, pop-up windows)
- Browser specific add-ons or toolbars (that cannot be found in other browsers)
- Browser specific (non-standard) HTML commands
- Bugs in the browser engine (because bugs can be exploited by web designers and artists)
- Colours. Certain old browser versions are not colour managed and cannot interpret colour profiles correctly.[55]

These browser attributes are important in order to decide whether they are suitable for a specific website. However, this kind of information is in most cases impossible to find. These behavioural properties can mainly b e found through testing and experimentation. Because a website's behaviour is a combination of website code, the browser software, client hardware, web server configuration, and internet performance, it will be challenging to rule out all these influences and describe pure browser properties.

55 https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/3.5/ICC_color_correction_in_Firefox and https://cameratico.com/color-management/firefox/ and https://cameratico.com/guides/web-browser-color-management-guide/ accessed September 2020

8. Selection of a web-browser for a specific website

This chapter discusses the selection criteria for a web browser in order to achieve the most authentic representation of a website (2nd research question of section 2).
In the early days of the internet, certain websites, such as those that were web-based artworks or games, were developed for specific web browsers. In particular, before Web2.0, there were specific, non-standard features (for instance, particular HTML commands, see chapter 4.1) that could only be interpreted by some browsers and not by others. Even after this period, when most of the internet has been standardised, certain websites are rendered better in certain browsers. Similarly, there are some add-ons that were only made for particular web browsers and that do not exist for other ones, as in the case of “Abstract Browsing” described in chapter 6.8.

As discussed in section 7, lists of the significant properties of web browsers are not easily available. To best select a browser environment for a specific website, we recommend consulting former users or user groups. Another method is to choose browser and plug-in versions that are contemporary with the years the website was most active. The availability of such components and their compatibility with the web browser is another factor to consider.

In many cases, the type of browser is not relevant, as many websites were designed to be viewed from any browser. In that case the availability of the browser might be the main criterion for selection.

Security risks are not usually a concern for the browser client itself as an emulation contains the attacks, and it can be reset if necessary. The browser emulation, though, can be a risk for a web server that is accessed through a leaky browser. Hence, the website and its server might have to be protected in other ways (e.g., through a firewall/proxy server, running the website in a container, or making regular snapshots).

9. Preservation of web browsers and their components

This section answers the 3rd research question of section 2 and describes how web browsers can be preserved. First, web browsers and their components need to be available before they can be preserved. This availability is not guaranteed in the long term, as browsers are generally hosted by single company archives or private collectors (s. sections 5.4, 5.5, 5.6 and Annexe 1).

As a preventive measure **the preferred browser environment and its single components should be acquired together with a web archive or a website.**

The components of browser emulations consist of **executables** or zip files (in case of add-ons) and are more or less stable. The source code of a browser and its components is usually not necessary to build a browser emulation. Therefore, it is not necessary for emulation and access purposes to map the development of a browser component from the first to the last version. This is rather an objective of source code archives with versioning control.

10. Creation and preservation of web browsers environments

10.1. Challenges when creating web browser environments

This section discusses how web browser environments can be created and preserved, answering the 4th research question of section 2. The case studies in section 6 showed that it can take a big effort to assemble the right browser environment including plug-ins and add-ons. Due to the complexity and ephemerality of web browsers and their components, the number of possible browser environments is much smaller in practice than in theory. Many components such as outdated add-ons, plug-ins, or browsers are difficult or not possible to find. To make them work together can be a challenge, too, as there is often a lack of information about how these components have to be installed. For instance, the metadata of a plug-in might just mention that it is made for Windows XP, but not which service pack, which build, which browser version or which other settings are needed.

Furthermore, an environment usually evolves slowly, as automated processes force users to install updates regularly. These user environments are built up from incremental steps. The process an archivist uses is different: he/she selects the (supposedly) right version of each component and installs them in the right order. For the reasons mentioned above, the compatibility of these versions is not always guaranteed.

The tests also showed that it makes quite a difference whether a browser environment will be used offline (for instance, for a web archive) or online: when using it online, the forced updating behaviour of operating systems, browsers, plug-ins, and add-ons has to be countered by disabling the automatic updating in each component. This can be complicated and is not always possible. The same goes for the blocking behaviour of web browsers: if they recognise that the installed components (mostly plug-ins) are not up-to-date, they might block them, and it might not be possible to unblock them except by updating the component. In that case, the component can usually not be updated to the next version, but only to the most current version.

Many obsolete operating systems such as Windows XP still maintain licensing systems. If you want to use them, licenses for obsolete operating systems have to be organised or purchased, which can be time consuming, as they are not sold officially anymore. For instance, on the Microsoft website it is not possible to purchase old operating systems or software unless you are a “Microsoft business partner,” and even then only the more recent versions are available.[56]

56 The website https://partner.microsoft.com/en-US/ explains how to become a Microsoft business partner. Currently (in 2020), Microsoft business partners can download old Windows operating systems—but only until Windows XP. Older operating systems are no longer made available.

For websites, the underlying client operating system is usually not important. However, it can be relevant for certain Internet Explorer (Windows) or Safari (MacOS) extensions that are not available in other browsers. The underlying hardware can be generalized for web browsers installed on computers, which makes browsers suitable for emulation.

For those who would like to set up a browser emulation themselves, instructions on how to do so using the EaaS platform are given in Annexe 7. The building of emulation environments is also described in the "Emulation as a Service for heritage institutions. Test Report[57]" by Eoin O'Donohoe from October 2020.

10.2. Description of web browser environments and their components

This section discusses how to describe browsers and their environments. In order to access a website with an obsolete web browser, a whole software stack needs to be available and described. Most likely, the browser emulation will be built from single components. However, it is also possible to use a disk image of a client instead. The **following lists of the information needed to describe web browsers at different levels in relation to emulation** are based on the conceptual framework of the EaaS platform[58]:

- **Component** level (for web browsers and related components such as browser plug-ins, browser add-ons, operating systems, Java Runtime Environment, mostly in form of executables): technical and descriptive metadata that characterize these components such as version, release date, build, browser engine, hardware architecture compatibility, operating system (version), installation instructions and requirements (s. also section 6).
- **Disk image** level (for browser environments and operating systems created from disk images): technical information about the disk image, provenance (What hardware / media type was the image made from? Where does it come from?), hardware requirements, content of the disk image (software stack).
- **Browser environment** level (for environments that were built by the conservator/archivist): A technical level that comprises all the components (software stack) and their relations to each other; a description of typical use, features, and behaviour of this browser environment (What kind of websites / period / former user groups was it used for?); an indication of a reference website; Was it assembled from components or does it stem from a disk image?; What was the reason to create this environment (for instance an exhibition, an access request, access of a specific website); Is this environment set up for online or offline access?; Is a proxy server involved? Specific browser and network settings; file format of browser emulation.
- **Emulator** level: What emulator and what emulator version is used for the browser emulation?; Command line and/or settings for emulation (computer architecture, use of libraries and drivers).

57 https://zenodo.org/record/4281471#.X8fPArN7IGw accessed November 2020

58 https://eaasi.gitlab.io/eaasi_user_handbook/guide/import-resource.html EaaSI is an implementation of EaaS at Yale University.

These levels are interconnected and can be represented in the metadata standard PREMIS[59] or other metadata standards and their content be based on or refer to Wikipedia / Wikidata. NDE will publish a separate report about metadata of emulated environments which will be relevant for browser emulations.

In contrast to an executable of a browser or a plug-in, browser environments are variable. They consist of several components which can be replaced or whose settings and metadata are changed. Such changes can be saved as snapshots that depend on the base environment or previous snapshots. Similar to changes (“commits”) in a versioning control system such snapshots have to be described in a meaningful and consistent way. How this can be mapped in PREMIS is to be investigated (probably snapshot as the output of an “event” executed by the “agent” emulator).

59 http://www.loc.gov/standards/premis/v3/premis-3-0-figures.pdf This pdf by the Library of Congress provides a good guideline of how to describe software and hardware environments and emulations with PREMIS

11. Problems a browser emulation can solve

This section addresses the issues of which problems a browser emulation can solve, and which it cannot (5th research question of section 2). For more detail on the latter question, see the case studies www.collapsus.com and www.wefeelfine.org in section 6, which exposed a number of problems a browser emulation could not solve. As we saw, there are many reasons why a website may not be able to be rendered properly. If a website is not running in the prepared environment, it has to be investigated to see if the problem is the website itself, the browser environment, or perhaps a simple setting on the web browser, such as allowing pop-up windows or activating plug-ins.

A **browser emulation** can resolve access problems due to obsolete plug-ins and browser features. It cannot fix problems server-side, such as outdated script language (for instance old versions of PHP) and database versions, or provide access to external data that has been dislocated, deleted, or otherwise changed. The web development tools in the browser as well as Linux commands like “weget” and “curl” can help to detect errors like missing resources.

12. Other outcomes of this research: Sharing browser emulations and their components

An important precondition for browser emulation is that the software stack is available. After a few years, part of the software may have already disappeared or not be publicly available. The accessible software archives are maintained by private companies or other private organisations and their existence is fragile.

There is a huge need for **software archives (executables) with a better distribution of risks,** for instance through multilateral funding and support, in order to **guarantee their long-term survival**. As this might not happen soon, organisations should store software and emulator collections on their own or in collaboration with other organisations.

To facilitate the use of browser emulations, **sharing functioning environments** would be the most efficient approach. The EaaS platform supports that approach. As is well known, software licensing terms have to be clarified for this. A separate report from NDC about that topic is underway.

Besides that, it would be useful to accumulate a **collection or database of reference browser environments** that were typical for a certain period, a certain type of web technology, and/or certain user groups. These reference environments or their description can support the creation of browser environments for emulation. This database should contain information about the user group who used this kind of environment, the types of websites that it was typically used for, the period when it was used, its distribution (niche or widespread), its components and versions, and typical features and behaviours. For digital video, there is a website that gathers typical artefacts: http://www.avartifactatlas.com/. This could serve as an example for setting up something similar for web browsers and their environments.

Linked metadata and the use of metadata standards would support the sharing of environments and browser components. In particular, references to Wikipedia and Wikidata could facilitate the exchange of browsers and their environments.

13. Outlook

Web browsers are still in a steep development curve, gaining the ability to render more and more immersive content. How different web browsers present 3D and virtual reality and the impact of WebGL on different browsers, as well as how to preserve such animations, are interesting questions left to be answered. How websites using webcams work in browser emulations has not been tested within this research and remains to be investigated.

Mobile devices have become ubiquitous for web browsing. The investigation of web browsers installed on mobile devices, their access to sensors, and websites with access to webcams could be a continuation of this study. Emulations of mobile devices might enable certain preservation steps.

The use of browser emulations in order to access web archives will become more and more important in the future in order to provide an authentic experience of captured websites. However, this **integration of custom-made browser emulations and access of web archives** is not a standard. The Internet Archive does not provide access to sites using different web browsers. Oldweb.today, as described in section 3, is an exception and a good example of how web archives can be viewed with different web browsers. During the course of this study, the EaaS framework underwent important developments with new features. The networking of emulation environments was made possible, as was the ability to create small, closed networks (like a simulation of a part of the internet) with a faux DNS server. The EaaS team also integrated access to the Internet Archive. In the future, it would be interesting to investigate whether and how archivists can create browser-specific access to web archives with these new EaaS features in order to improve the authentic playback of web archives.

14. Bibliography

Alcorn, Wade; Frichot, Christian; Orru, Michel (2014): The Browser Hacker's Handbook: Wiley.

Bang S (2004) Archiving Web Browser Plug-ins: Working draft for the International Internet Preservation Consortium (IIPC). https://digital.library.unt.edu/ark:/67531/metadc1457745/m2/1/high_res_d/archivingwebbrowserplugins_0_1_166.pdf

Espenschied, Dragan; Kreymer, Ilya (2016): Workshop: Symmetrical Web Archiving With Webrecorder. In ipres (Ed.): iPRES 2016 - 13th International Conference on Digital Preservation. iPRES 2016. Bern, Switzerland, 3-6 October 2016.

Ferdman, Sela; Minkov, Einat; Bekkerman, Ron; Gefen, David (2017): Quantifying the web browser ecosystem. In *PLOS one*. Available online at https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0179281.

Horsman, Graeme (2018): I didn't see that! An examination of internet browser cache behaviour following website visits. In *Digital Investigation* 25 (June), pp. 105–113. Available online at https://www.sciencedirect.com/science/article/pii/S1742287617301524?via%3Dihub.

Xu Z, Miller J (2018) Cross-Browser Differences Detection Based on an Empirical Metric for Web Page Visual Similarity. ACM Transactions on Internet Technology 18 (3):1–23

Annexes

Annexe 1: Browser components and configuration

Firefox add-ons

Firefox does not maintain an archive of legacy Firefox add-ons. However, someone unaffiliated with Firefox created an archive of old Firefox add-ons in a GitHub archive[60]. The readme of this archive states: "*This catalog contains 93,598 versions of 19,450 Firefox add-ons created by 14,274 developers over the past 15 years using XUL/XPCOM technology before Mozilla decided to ruin the classic extensions ecosystem and go exclusively to WebExtensions.*" According to the readme, these add-ons were tested with Firefox ESR 45-52 and Firefox 45-58b.

The Wayback Machine gives access to legacy Firefox add-ons from Firefox version 57 on: https://web.archive.org/web/20171004160834/ . As the search for add-ons in the archived Firefox interface does not work, it is very difficult to find a specific add-on. In addition, the add-ons are automatically updated when installed (if an update exists). We did not test whether the updating can be prevented by adjusting the Firefox settings, but we did determine that downloading the add-ons without installing them is not possible. The location of the installed add-ons on the local computer can be checked by entering the URL "about:support" in the URL-box of the Firefox browser and then by going to the field "Profile Folder."

Google Chrome add-ons

According to Cimpanu,[61] 188,620 extensions were available on the Chrome web store in 2019. A small number of these extensions have been downloaded several million times, while many have been downloaded just once. Examples of popular Google Chrome extensions are "Google Photos," used to access photos in the cloud, and "Text," a text editor.

Google Chrome extensions can be developed from within the Chrome Browser in developer mode. According to the Google Chrome website,[62] "extensions are downloaded by the Chrome browser upon install, and are subsequently run off of the local disk in order to speed up performance. However, if a new version of the extension is pushed online, it will be automatically downloaded in the background to any users who have the extension installed."

There is no official Google Chrome add-on archive. Only current extensions can be downloaded from the official Google Chrome web store.[63]

60 Github archive of Firefox add-ons for Firefox version 45 to 58: https://github.com/JustOff/ca-archive/

61 https://www.zdnet.com/article/half-of-all-google-chrome-extensions-have-fewer-than-16-installs/ By Catalin Cimpanu, August 3 2019, accessed 2020/05/20

62 About Chrome extensions: https://developer.chrome.com/extensions/faq#faq-gen-01

63 Download Chrome extensions: https://chrome.google.com/webstore/category/extensions?hl=en

Java applets (plug-in)

Java applets were used for interactive graphics, animations, and 3D-graphics before CSS and Javascript had these capacities. Web designers and artists used them not only to create animations, but also for "trivial effects such as rollover navigation buttons."[64] According to Wikipedia,[65] until 2011, Java applets ran 3D graphics much faster than Javascript, as they had access to 3D hardware acceleration.

Java applet graphics were embedded in web pages, but they could also be integrated so that they opened in a separate browser window. The embedding of the applet in the web page depends on which HTML-version is used and the ability of the web browser to understand it (some commands seem to be browser-specific, such as the "embed" command that could only be interpreted by Mozilla browsers[66]). HTML web pages were able to pass parameters to Java applets. Java applets were based on the web API NPAPI and have been obsolete since 2016[67].

Old Java Runtime Environments (JRE) can be downloaded from Oracle's website[68] (Oracle acquired Sun Microsystems in 2010). The Java version history can be found on Wikipedia[69]. When installing the Java Runtime Environment (JRE), the plug-in is installed automatically in Windows[70]. For MacOS it is important to know that only one JRE version can be installed at a time. If an older one has to be installed, the newer one needs to be uninstalled first.[71] For Linux, the plug-in is also installed with the JRE, but some additional steps have to be taken.[72]

According to Dragan Espenschied, the conservator of Rhizome, the standard environments the artists were developing their works for were Netscape browsers up to version 4.8 and later Windows XP (SP1, which includes Internet Explorer 6 and the Microsoft Java VM). "[By] the time Windows 7 arrived, Java applets were really mostly used in intranets where company system administrators could distribute a Java browser plug-in to all users").[73] That confirms that certain user communities use plug-ins in specific ways and in specific environments.

64 https://en.wikipedia.org/wiki/Java_applet accessed 2020/06/03

65 https://en.wikipedia.org/wiki/Java_applet accessed 2020/06/03

66 https://en.wikipedia.org/wiki/Java_applet accessed 2020/06/03: "*The applet can be displayed on the web page by making use of the deprecated* `applet` *HTML element, or the recommended* `object` *element. The* `embed` *element can be used with Mozilla family browsers* (`embed` *was deprecated in HTML 4 but is included in HTML 5). This specifies the applet's source and location. Both* `object` *and* `embed` *tags can also download and install Java virtual machine (if required) or at least lead to the plug-in page.*"

67 https://java.com/en/download/help/enable_browser.xml accessed 2020/06/03: Java plug-ins work only until Chrome version 41, Firefox version 51, Safari version 11.

68 Java JRE archive: https://www.oracle.com/java/technologies/oracle-java-archive-downloads.html

69 https://en.wikipedia.org/wiki/Java_version_history

70 https://docs.oracle.com/javase/9/install/installation-jdk-and-jre-microsoft-windows-platforms.htm#JSJIG-GUID-B2D8D2D9-D902-4CA8-8282-4FA4391E1B17 : "*Java Plug-in technology, included as part of the JRE, establishes a connection between popular browsers and the Java platform. This connection enables applets on websites to be run within a browser on the desktop. The Java Plug-in is automatically enabled for supported web browsers during installation of the JRE. No user intervention is necessary*"

71 https://docs.oracle.com/javase/9/install/installation-jdk-and-jre-macos.htm#JSJIG-GUID-5F4A0659-BFC5-4CB9-9920-D2DEABF29894 "*When you install the JRE, you can install only one JRE on your system at a time. The system will not install a JRE that has an earlier version than the current version.*"

72 Manual Installation and Registration of the Java Plug-in on Linux: https://docs.oracle.com/javase/9/install/manual-installation-and-registration-java-plug-linux.htm#JSJIG-GUID-252AC1DA-02BB-4B27-9CD9-C2CBA5D871BA

73 Email, June 12; 2020 to crock@beeldengeluid.nl

Flash and Shockwave plug-ins

Shockwave and Flash both provide interactive multimedia content such as video games or interactive graphics. Typically, they were embedded in a web page. Both were developed with MacroMind Director.

MacroMind developed Shockwave from the late 1980s until 1993. It was mainly used for interactive CD-ROMs and kiosk presentations. During this time, there was no Shockwave player. Therefore, the only way to publish interactive content was by generating executable applications. In 1993, Macromedia acquired MacroMind and released the first Shockwave player in 1995.[74] Shockwave's script language is called "Lingo."

Flash started as SmartSketch and was developed by FutureWave Software around 1993. SmartSketch created vector-based animations. Per Wikipedia, "In 1995, FutureWave modified SmartSketch by adding frame-by-frame animation features and released this new product as FutureSplash."[75] In 1996, Macromedia acquired FutureSplash and released it as Macromedia Flash 1.0. Flash's script language is called ActionScript.

Adobe Systems acquired Macromedia and hence both Flash and Shockwave in 2005. Adobe Systems continued the development of Flash and Shockwave until about 2017. At this time, they began to phase out the Shockwave and Flash players gradually, depending on the web browser and operating system. For instance, the Shockwave player for MacOS was discontinued in 2017. Shockwave was finally discontinued for all operating systems in 2019 and Flash in 2020.

The names of the Shockwave and Flash player plug-ins are somewhat confusing. The Shockwave player's full name is "Shockwave Director player," though it is colloquially known as just "Shockwave." The Flash player's full name is "Shockwave Flash player," but it is colloquially known as just "Flash."

The Shockwave and Flash file format extensions also need an explanation. The SWF file format extension used to be short for "Shockwave Flash," which was the Flash format. As this abbreviation was confused with the Shockwave format, it was renamed "Small Web Format." The file format extensions used for Shockwave are DIR, DCR, or DXR.

The version history of the Shockwave player is documented on the Adobe website.[76] However, there is no official, accessible Shockwave player archive from Adobe. Adobe Shockwave players versions 7 to 12 for Windows OS can be downloaded at the Internet Archive: https://archive.org/details/ShockwaveInstallers. Unlike Flash, the Shockwave browser plug-in is not available for Linux. The MacOS Shockwave players are more difficult to get. There are various private sites where specific versions of Shockwave player can be downloaded.[77]

In contrast to Shockwave, there is an official Flash player archive from Adobe[78] for versions 2 to 32. It contains the versions for the operating systems Linux, Windows, and MacOS and for different browser APIs including NPAPI, ActiveX, and PPAPI.

74 https://en.wikipedia.org/wiki/Adobe_Shockwave accessed 08/06/2020
75 https://en.wikipedia.org/wiki/Adobe_Flash accessed 08/06/2020
76 https://helpx.adobe.com/shockwave/kb/shockwave-player-version-history.html
77 For instance https://mac.filehorse.com/download-shockwave/download/ (Versions 11.6 to 12.2) for MacOS
78 https://helpx.adobe.com/flash-player/kb/archived-flash-player-versions.html accessed September 2020

ActiveX: Internet Explorer API for browser extensions or plug-ins

ActiveX was developed by Microsoft in 1996 and is a framework used to render content that is embedded in a web page. Not only Internet Explorer, but also Windows Media Player and Microsoft Office used ActiveX for certain features. ActiveX still works with Internet Explorer 11, which is why Internet Explorer 11 is still installed in Windows 10, in order to maintain compatibility with legacy web technologies. However, the development of Internet Explorer was stopped in 2015 and it has been replaced by Microsoft Edge, which does not use ActiveX. ActiveX only runs on X86-architecture.

There was also an ActiveX plug-in for Firefox for users who had a Firefox browser installed on a Windows machine.

Because ActiveX constituted a security risk for the browser client using it, the user could deactivate it in the Internet Explorer settings. This was called ActiveX Filtering.[79] When ActiveX Filtering was on, videos and games could not be rendered correctly.

Users did not usually install ActiveX directly. Instead, the installation would be triggered after a user installed a plug-in (for instance, a Shockwave version or Flash plug-in) made for ActiveX , which would prompt Internet Explorer to ask to install or update ActiveX.[80] This behaviour can become a problem when creating browser emulations.

Browser Configuration / Policy

Browser policies are meant for companies who want to prevent the installation of add-ons, or prevent access to certain websites, or provide the same bookmark list for all the employees. The use of browser policies for Firefox is documented on https://github.com/mozilla/policy-templates. Google Chrome policies are implemented in operating systems (for instance, Windows Registry). The website “chrome://policy,” if entered in the URL box of Chrome, lists Chrome’s preconfigured policies.

Browsers can usually be configured in much more detail than just in the settings. A sophisticated browser configuration can be useful to regulate browser emulation access at archives or exhibitions. These features are a bit hard to find, so brief descriptions of how to access them in both Firefox and Chrome are outlined in the following paragraphs.

In order to check the **Firefox** configurations and be able to configure Firefox in detail, use the URL box of the Firefox browser and enter: “about:about”. This website lists all the possible “about:” sites. “about:support” provides a good oversight of the local browser environment with a local path for the storage of the Firefox user profile (containing the add-ons). Particularly useful are: “about:config,” which lists all the hidden (implicit) configurations of Firefox. To manipulate these configurations, you need professional knowledge. “about:plug-ins” and “about:addons” list the installed Firefox plug-ins and add-ons.

Google Chrome has a similar function. Entering “chrome://chrome-urls/” or “chrome://about” in the URL box of the Chrome browser yields a list of “chrome:” websites. The website “chrome://system/”

79 https://support.microsoft.com/en-gb/help/17469/windows-internet-explorer-use-activex-controls accessed June 2020

80 https://en.wikipedia.org/wiki/ActiveX : Starting with Internet Explorer 3.0 (1996), Microsoft added support to host ActiveX controls within HTML content. If the browser encountered a page specifying an ActiveX control via an OBJECT tag (the OBJECT tag was added to the HTML 3.2 specification by Charlie Kindel, the Microsoft representative to the W3C at the time[8]) it would automatically download and install the control with little or no user intervention.

lists the browser version, operating system, and add-on versions. The site “chrome://version” lists the versions of the installed plug-ins, JavaScript, and path of local profile. The “chrome://flags” probably corresponds most to the “about:config” site of Firefox. It lists the configurations of Chrome that are not usually accessible in the settings.

Annexe 2: Results browser emulation for www.wishingtree.nl

Year	Installation within emulation[81]	Results Installation	Results Behaviour wishingtree.nl
2010	Windows XP, Firefox 3.0.5 Flash player 10.1.102.64.	Firefox blocks the Flash plug-in. It cannot be activated.	The tree animation cannot be rendered. Wishes cannot be entered.
	Windows XP, IE 6.0.2900 Flash player 10.1.102.64.	Flash plug-in installed in IE	Video animation works well.
2013	Windows XP, Firefox 25 / IE 6.0.2900 Flash player 10.30.r183.90	Flash plug-in installed in Firefox and IE	No difference between Firefox 25 and IE6 can be seen.
2016	Windows XP, Firefox 44 Flash player 11.6.r602	Flash plug-in installed in Firefox and IE	No difference between Firefox 44 and IE6 can be seen
	Windows XP, IE 6.0.2900 Flash player 32.0.0.363	It was not possible to install Flash player 11.6.r602 in IE6. The most current Flash player version 32 was automatically installed.	No difference between Firefox 44 and IE6 can be seen
2019 / 2020	Windows 7 Firefox 66 IE11 Flash player 32.0.0.363	The newest Flash player version 32.0.0.363 was added.	No difference between Firefox 66 and IE11 can be seen
2020	Not emulated (directly installed on laptop), Windows 10, Firefox 77.01, Internet Explorer 11.0.195	The newest Flash player version 32.0.0.363 was added.	No difference between Firefox 77 and IE11 can be seen

No differences were visible and audible between the Firefox versions.

81 The Windows XP emulation was done with the qemu emulator for x86 computers on the EaaS platform, the Windows 7 emulation with virtualbox.

Annexe 3: Results browser emulation for www.wefeelfine.org

Year	Installation within emulation[82]	Results Installation	Results Behaviour wefeelfine.org
2010	Windows XP, Firefox 3.0.5 / IE 6 and Java 6.18 (JRE)	The Java plug-in is installed in IE and Firefox	Firefox and IE both exposing same error message: “The application failed to run.” The downloaded wefeelfine.jar file cannot be played back.
2013	Windows XP, Firefox 25 / IE6 Java 7.11 (JRE)	The Java plug-in is installed in IE and Firefox	Firefox and IE both exposing same error message: “Runtime Exception.” The downloaded wefeelfine.jar file cannot be played back.
2016	Windows XP, Firefox 44 / IE6 Java 8.72 (JRE)	Java 8.72 could not be installed. (errors)	IE6 and Firefox could not render the website The downloaded wefeelfine.jar file cannot be played back.
	Windows XP, Firefox 44 / IE6 IE6, Java 7.11 (JRE)	The plug-in was automatically installed in Firefox.	Firefox 44 does not render the website, although the plug-in is installed. IE6 could not render the website either.
2019	Windows 7 Firefox 66 / IE 11.0.9600 Java 1.8.0.251 (JRE)	Most current Java version installed (JRE). The Java plug-in 11.251.2 is installed and enabled in Internet Explorer 11.	Java is not supported by Firefox from Version 52. Internet Explorer did not work with the most current Java version, although it should.[83] The Java animation could not be rendered. The downloaded wefeelfine.jar file could not be played back.

82 The Windows XP emulation was done with the qemu emulator for x86 computers on the EaaS platform, the Windows 7 emulation with virtualbox.

83 When opening wefeelfine.org in IE11, IE pretends the Java plug-in is not installed. In a second installation attempt it said the plug-in was blocked. Although in the Internet options of IE11, scripting was enabled and the website was added to the list of trusted websites, it was not possible to unblock the Java plug-in. Checking in the developer tools of IE (DOMexplorer),it gives the error “An error has ocurredJSPlugin.3005”. According to https://stackoverflow.com/questions/28301393/an-error-has-ocurredjsplug-in-3005 it depends how IE11 was installed (From which version it was upgraded, etc.). So, the error seems to be a bug.

Annexe 4: Results browser emulation for www.tebatt.net

Year	Installation within emulation[84]	Results Installation	Results Behaviour http://www.tebatt.net/JAVAGALLERY/SKETCHES/ARTICULATION/GRAMMARS/Gram/Gram.html
2010	Windows XP, Firefox 3.0.5 / IE 6 and Java 6.18 (JRE)	The Java plug-in is installed in IE and Firefox	Firefox and IE both rendering the animation. Also works when downloading and playing back the gram.jar file with Java.
2013	Windows XP, Firefox 25 / IE6 Java 7.11 (JRE)	The Java plug-in is installed in IE and Firefox	Firefox is rendering the animation, IE not. Works also when the gram.jar file is downloaded and played back with Java.
2016	Windows XP, Firefox 44 / IE6, Java 8.72 (JRE)	Java 8.72 could not be installed. (errors)	IE6 and Firefox could not render the animation
	Windows XP, Firefox 44 / IE6, Java 7.11 (JRE)	The plug-in was automatically installed in Firefox	Works when the gram.jar file is downloaded. Animation is not rendered directly from the IE or Firefox browser.
2019	Windows 7 IE 11.0.9600 Java 1.8.0.251 (JRE)	Most current Java version installed (JRE). The Java plug-in 11.251.2 is installed and enabled in Internet Explorer 11.	Works when the gram.jar file is downloaded. Does not work directly from the Internet Explorer.

84 The Windows XP emulation was done with the qemu emulator for x86 computers on the EaaS platform, the Windows 7 emulation with virtualbox.

Annexe 5: Rafel Rozendaals email

From: **Rafaël Rozendaal <newrafael@gmail.com>**
Sent: Wednesday, 29 January 2020 16:16
To: crock@beeldengeluid.nl
Cc: Gaby Wijers <gabywijers@li-ma.nl>
Subject: Re: Abstract Browsing (2014)

Hi Claudia,

my answers are below:

Rafaël Rozendaal

> On Jan 29, 2020, at 06:59, <crock@beeldengeluid.nl> <crock@beeldengeluid.nl> wrote:
>
> Dear Rafael
>
> I'm working at the Institute of Sound and Vision in the Netherlands. We are looking into web browser emulation and how we can access and preserve obsolete web browsers in collaboration with LIMA (Gaby Wijers copied in). These obsolete web browsers in return can be used to view technically obsolete websites. We will use your artwork "Abstract Browsing" which is technically a browser extension as a case study.
>
> I am interested in the evolution of the "Abstract Browsing" browser extension.
> • Was it always a chrome extension or did you also program it for other browsers?

It was always a Chrome extension, I use Chrome as my main browser, that's why. Also, my developer works on Windows, so Chrome was a cross platform choice.

> • How often did you have to update the extensions and for which reasons? (Did you make the changes for technical reasons or were there other reasons)?

In the first two years we updated it a few times to fix bugs. Since then it has not been updated, it works fine most of the time. There are some websites that don't work perfectly (some images are still visible) but that's OK, most of the time it works great.

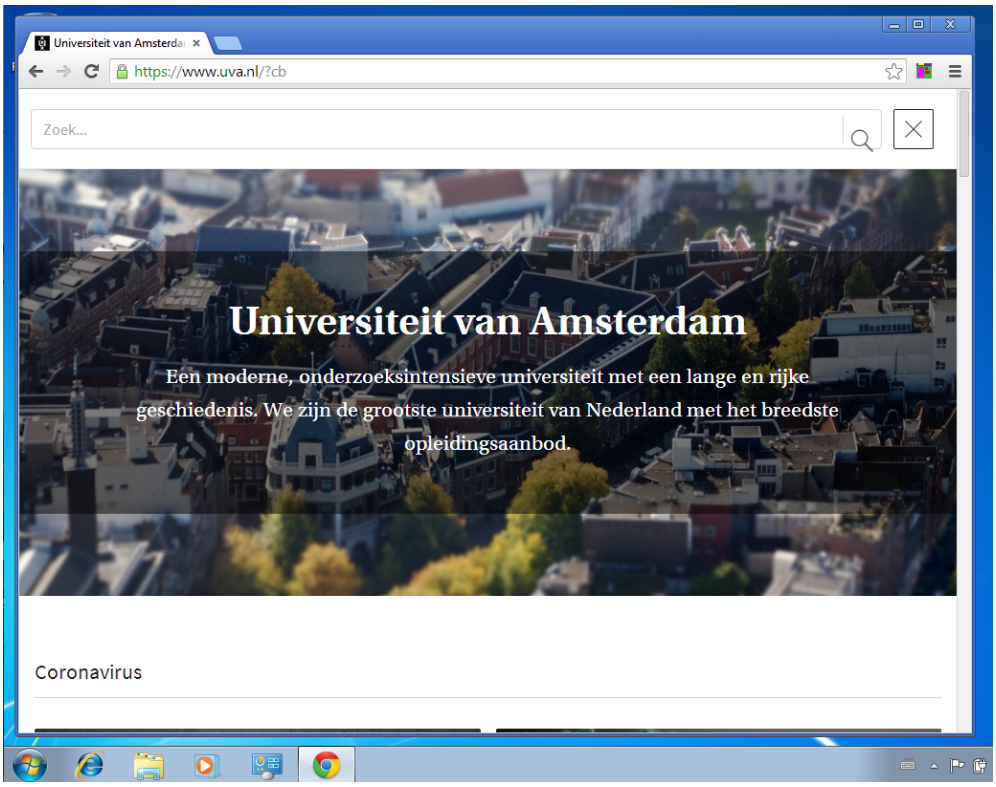
> • Do you preserve the old versions, and if yes, how?

Yes, we save the old versions as .zip files in our shared Dropbox

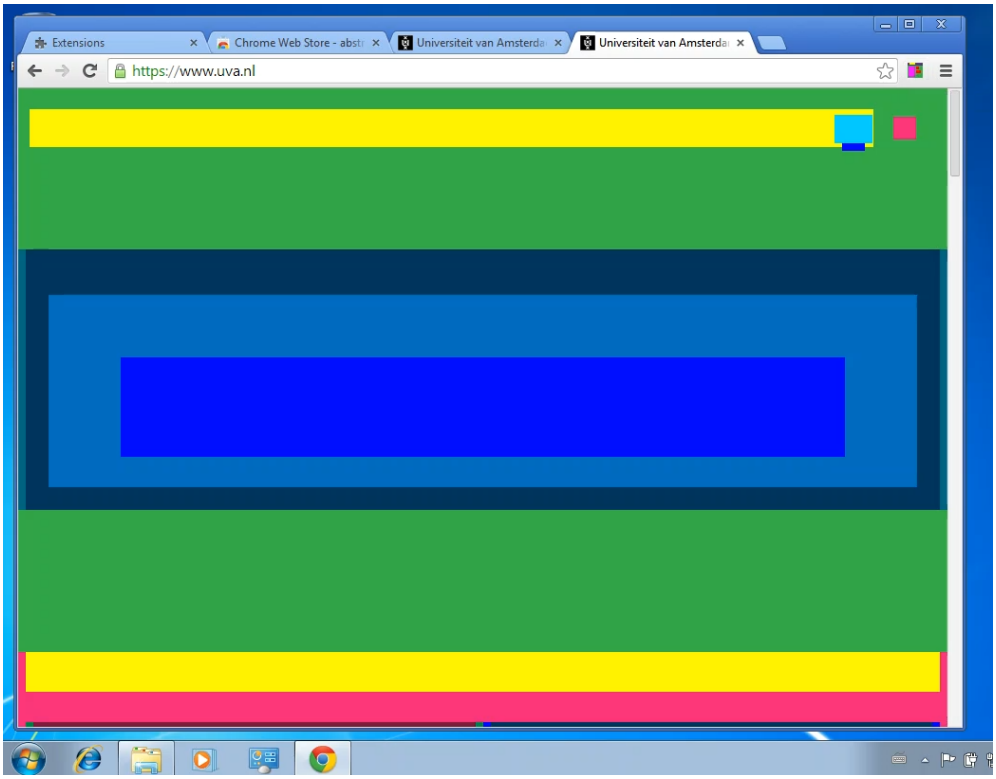
> • Are you planning to continue updating the browser extension?

Yes
>
> If you had time to answer us these questions, it would be amazing.
>
> Best wishes,
> Claudia Roeck
> +31 655 489 552

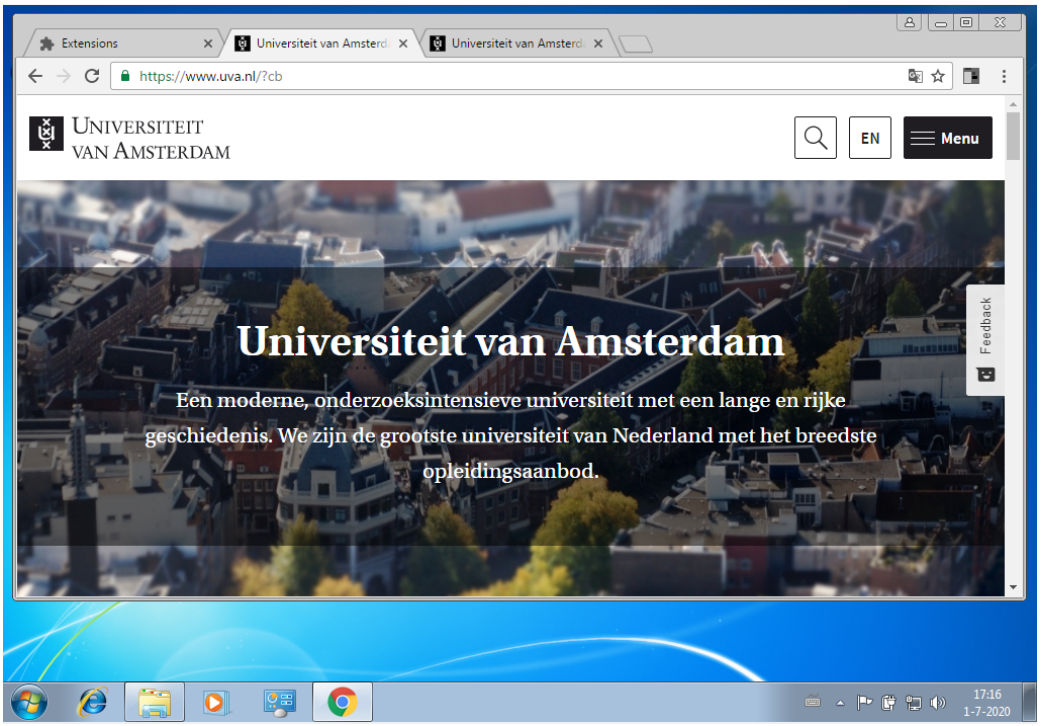
Annexe 6: Screenshots tests Google Chrome and "Abstract Browsing"



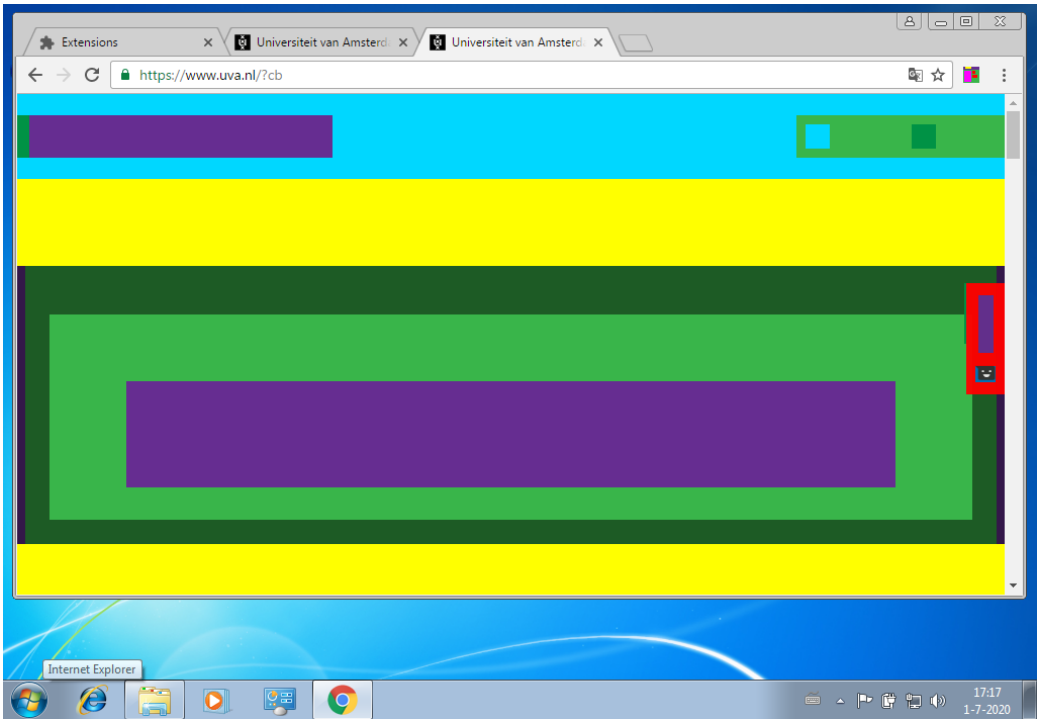
Chrome Version 33. www.uva.nl. The feedback button on the right window side is missing.



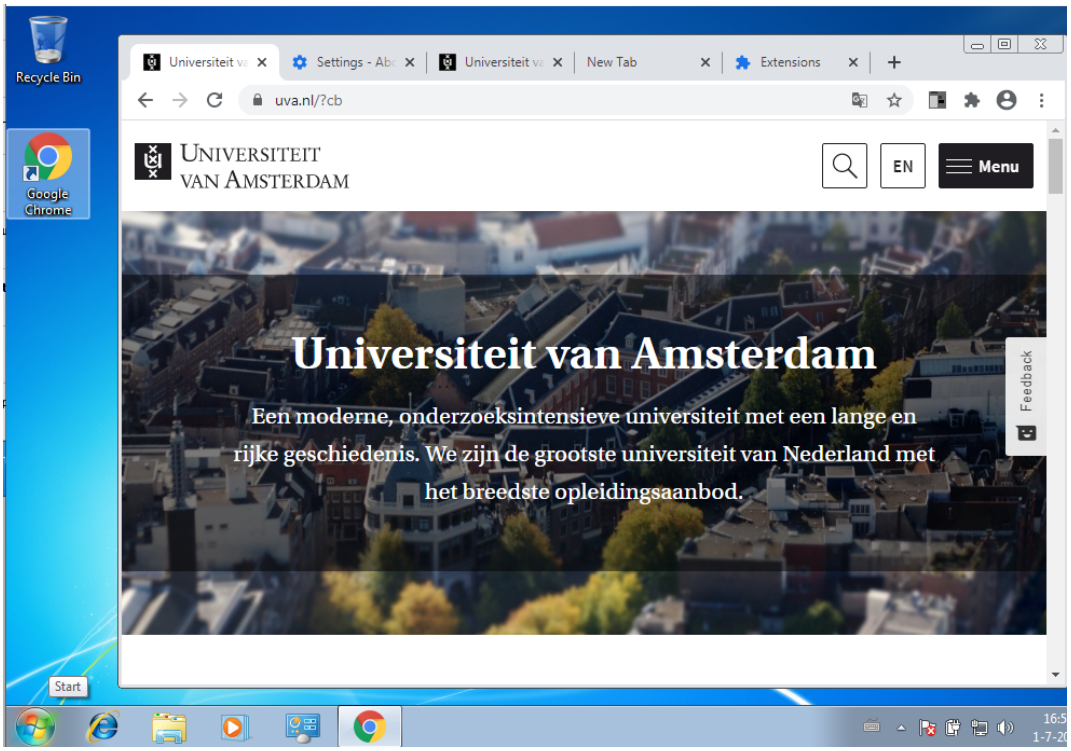
Chrome Version 33. www.uva.nl. The feedback button on the right window side is missing.



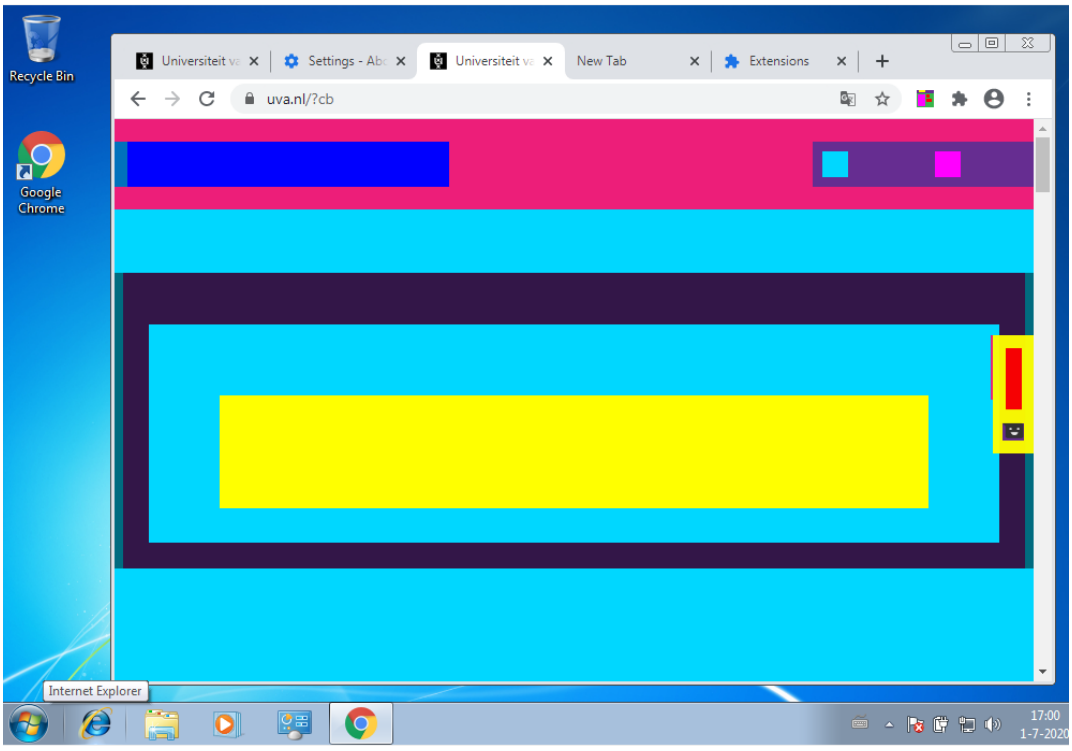
Chrome Version 56. www.uva.nl. The feedback button on the right window side is rendered.



Chrome Version 56. www.uva.nl. The feedback button on the right window side is rendered.



Chrome Version 83. www.uva.nl . The feedback button on the right window side is rendered.



Chrome Version 83. www.uva.nl. The feedback button on the right window side is rendered.

Annexe 7: Browser Emulation instructions for the EaaS platform

Necessary components:

- operating system (OS)
- web browser software compatible with OS
- plug-in software compatible with OS and browser
- add-on software compatible with browser and OS

Steps for building a browser emulation on the EaaS-platform[85], developed by the bwFLA team at the University of Freiburg (D) and provided by OpenSLX:

1. Install an operating system[86] (consider the system's licensing when choosing). Alternatively, share an operating system (base environment) by selecting one from the "Environments/Virtual machines" tab. If it is an obsolete operating system, make sure it does not update automatically when connected to the internet—check the system update settings and deactivate automated updating. Shut down the operating system from within the guest system and then save the environment as a new environment.
2. Make sure the internet access of this base environment is disabled: go to the "Environments/Virtual machines" tab and select the base environment, click on "Details," then "Networking," and uncheck the boxes labeled "Enable Networking" and "Enable Internet Access." Then save the environment (scroll down to do that).
3. Install browser version and plug-ins: in your host system download the desired browser and plug-in versions from the Internet or from an archive and create an iso-File out of the browser or plug-in software that needs to be installed. In EaaS, import this iso as an object. Click on the "Details" button of the object, and go to "Rendering Environments," click "Add Environment" and select the operating system you prepared in step one. Then click "Run Environment." While or immediately after installing browser and plug-ins, make sure to disable automatic updates! Depending on the website that will be presented, pop-up windows need to be enabled in the browser settings. After having installed everything within the environment, shut it down from within the environment and save any changes by clicking on the EaaS sidebar on the left. This step can be done iteratively: first for the web browser and then for the plug-in. The object environment can then be saved as a revision.
4. Adapt the screen resolution within the environment. For instance, in Windows XP, right click on "Properties" when hovering over the screen, then select "Settings" and adjust the display resolution. The emulation will automatically adapt the size of the emulator window. The emulator window will stay visible (full screen not possible). Shut down the operating system from within the guest system. Save the environment.
5. On the EaaS platform, enable access to the internet for this freshly created browser environment: go to the "Environments/Object Environments" tab and select the environment, click on "details," click on "networking," and check the boxes "enable networking" and "enable Internet access." Then save the environment (scroll down to do that).

85 Build from March 2020.

86 Check the EaasI-guidelines https://eaasi.gitlab.io/eaasi_user_handbook/ . EaaSI is an implementation of EaaS at Yale University.

These steps could easily be adapted for other emulation platforms or hypervisors. Emulation platforms offer more potential than a hypervisor, as they provide several emulators for different hardwares. Another advantage of EaaS is the exchange of object and base environments within the EaaS community. The installation process for browsers or plug-ins can be cut short by building on existing environments. However, the organisations using such platforms need to be aware of the fact that the emulators hosted by these platforms are necessary to access their browser environments. Finally, the newest version of EaaS (2020) provides the ability to embed an emulation window in a website in order to facilitate access.

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[87]) is an implementation of the EaaS platform at Yale University. The university and OpenSLX made a user documentation[88] 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

87 https://www.softwarepreservationnetwork.org/eaasi-gitlab/ (accessed october 2020)
88 https://eaasi.gitlab.io/eaasi_user_handbook/ (accessed october 2020)

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 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,[89] 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.[90]

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

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

90 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)

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

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.

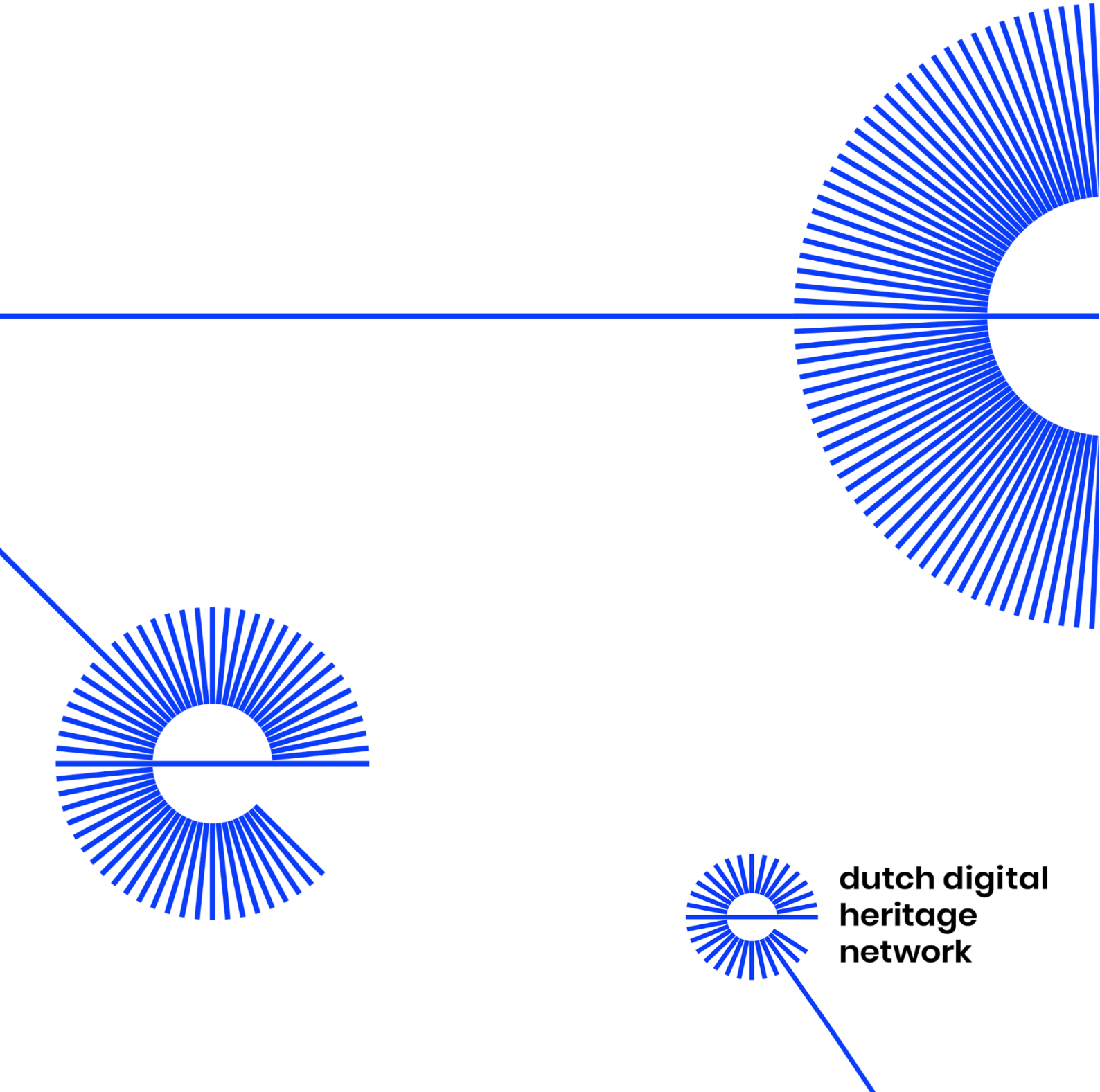
World Wide Web (www)

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.

About this publication

This article was published by the Dutch Digital Heritage Network (NDE) in January 2021.
For further information, see: www.netwerkdigitaalerfgoed.nl/en/ .

If you have any queries or comments about the contents of this article, please feel free to email them to: info@netwerkdigitaalerfgoed.nl.



dutch digital heritage network