FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

The Chaser

Chrome Extension For History Visualization

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Chris Boesch
Matrikelnummer 1025952

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Dipl.-Ing. Dr.techn. Ivan Viola
Mitwirkung: Dr.techn. M.Sc. Manuela Waldner

Wien, 13. August 2014

Chris Boesch	Ivan Viola

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

The Chaser

Chrome Extension For History Visualization

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Chris Boesch
Registration Number 1025952

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Dipl.-Ing. Dr.techn. Ivan Viola
Assistance: Dr.techn. M.Sc. Manuela Waldner

Vienna, 13th August, 2014

Chris Boesch | Ivan Viola

Erklärung zur Verfassung der Arbeit

Chris Boesch
Brandmayergasse 36/11 1050 Vienna

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. August 2014

Chris Boesch

Abstract

Revisitation of previously requested URLs happens frequently and the most common list-view-based visualization of the user's browsing history provided by nearly every internet browser cannot give a compact general view. For this reason we designed and implemented an extension for Chrome called The Chaser, by which an alternative visualization of the content is possible. The currently available add-ons / extensions have other aims to illustrate the history. Some of them are about to show the differences in quantity of called pages. Others give an overview of page paths and the user's tracks from site to site. Our extension concentrates on helping finding a visited page and giving users a better overview of their called URLs. The user should get the ability to control the time-line with mouse gestures and/or keyboard input. After discarding a 3D prototype we came to the conclusion of designing a simple, self-explanatory time-based illustration with two dimensions. The x-axis represents the time with different levels of detail and the y-axis the visited hosts. After performing an evaluation with six probands where The Chaser's visualization and its efficiency was compared to the standard list-view, all of them would prefer our extension against the standard history view.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Studies show that over eighty percent of the requested URLs have been visited previously [CM01]. Web histories can help users re-finding sites but they use the standard history list rarely according to [TG97]. This suggests that the visualizations with a list-view in the current internet browsers are not appealing to the end-users. On this account our new history illustration called The Chaser represents the visited elements and the temporal distribution of URL visits in a single compact time-line-based visualization as it could support revisitation. All data is illustrated in two dimensions with the time progress displayed on the x-axis and the hosts on the y-axis. There are several examples exploring new history representations with the third dimension or other visual elements like for example WebComets [CTKE13], BrowseLine [HG09] and many more. In Chapter 2 six visualizations are presented with different goals than revisitation or giving an overview. A very good example is WebPath - A Three Dimensional Web History [FS98] where the user also has the ability to search for a specific visited URL including the relationships between all site calls. As an example, Figure 2.5 shows the navigation and the structure provided by WebPath. It is not possible to get a general view of all site calls from one year and that is where The Chaser can score since all calls can be visualized in a time-line-based illustration. The user should get a good overview on his history and re-finding of visited sites should be easier than with the standard view. Time-dependent data can be presented with a linear approach as users need an easy-to-use system for accessing large data sets [FC00]. Proper specifications of all design features, properties and which technologies are used can be found in Section 3.2 and Chapter 4. To get a better insight in other history mechanism, details can be found in Section 2.2, where other papers and visualizations are described. As a measurement of efficiency, an evaluation with six people indicates the benefits of the Chrome extension in comparison with the standard history view provided by Chrome (see Chapter 5). In the last Chapter 6, some future ideas will be presented and an overview will be given.

CHAPTER 2

Related Work

Nowadays many extensions and add-ons of internet browsers are present on the market and all of them have slightly different aims and techniques to visualize the user's history with its site calls. In the following sections two main categories are presented which enclose several papers and publications. The first group tries to illustrate the history in a strict linear way in two dimensions and on the other hand the second one visualizes elements by trying to use a third dimension in different variations.

2.1 History Visualization with a time-line approach

The biggest and most obvious benefit of a two dimensional illustration is, that nearly every user has an idea how to read a chart with this design which is discussed here [CM02]. To use one axis for time is not a new invention and can be found in several papers. Some methods on how to display big data sets is described in the paper [MS03]. There is a chart mentioned, which was made about 150 years ago by Minard, who lived from 1781 to 1870 (see Figure 2.1). It is one of the best illustrations from that century in case of containing more than two attributes. Classic charts in two dimensions display two attributes by having two axes. The following paragraphs describe three different methods with a similar approach to illustrate time-dependent data. The table 2.1 gives an overview how x-,y-axis and time-line-marks are used to map the history items and additionally The Chaser is included to show differences.

Eyebrowse [MVK10] is the first addon presented and has three main goals but only one of them is mandatory for this work. It is about giving users a better understanding on their online behavior. The extension is realized with a Firefox plugin and a website. As Figure **??** shows, Eyebrowse gives a general overview of URLs that have been clicked. The extension tracks the time a user spends on a site and is therefore able to display an overview of days with exact online periods. The x-axis is used to show all site calls during one day and the days are listed in y-direction. The difference to our solution is

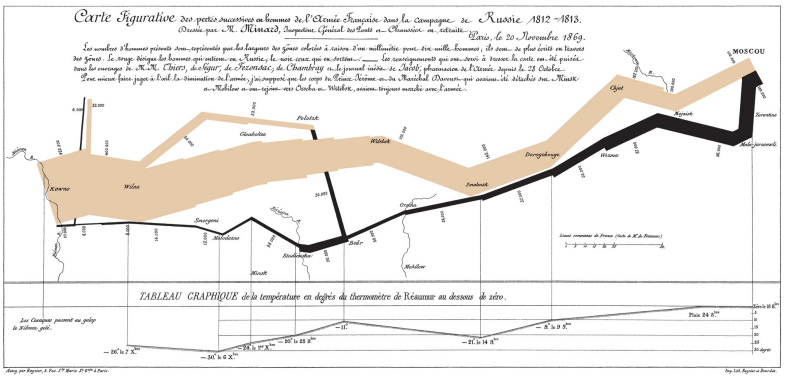

Figure 2.1: Minard's visualization of the Fate of Napoleon's Army.

Name	**x-axis**	**y-axis**	**marks**
EyeBrowse	one day (time)	last 20 days	URL calls with domain grouped color-codes
BrowseLine	sequence of calls	hours of day	domain-grouped site calls
WebComets	zoom-able time-line	used tabs	URL calls with several attributes (e.g.: duration)
The Chaser	zoom-able time-line	domains	URL calls

Table 2.1: Overview of time-line-based visualization and their mapping settings.

that The Chaser uses the y-axis for the grouped site calls. To display the URL attributes, Eyebrowse illustrates them with different colors. As it can be seen same URLs have the identical appearance and as a consequence frequencies can be read off easily. For re-finding a visited page this approach is not beneficial as it is hard to detect a certain color-coded element.

Helping users to re-find a website is the task Browseline [HG09] wants to accomplish. The app has a two-dimensional time line which can be seen in Figure 2.2. The y-axis represents the Macro time, which encompass data values from one hour. On the other hand the x-axis splits the hours from the vertical measurements to show orders of URL calls. Therefore the exact time a site is called is not visualized but can be reached by clicking on a domain stack. The blue color of those represents how many sub-pages with the same domain are included. The darker the color the more often sub sites of a certain domain have been called. In comparison to our approach their extension illustrates grouped sub-domains with color codes whilst we decided to illustrate all sub-calls along the x-direction chronologically with exact positioning according to their time stamps.

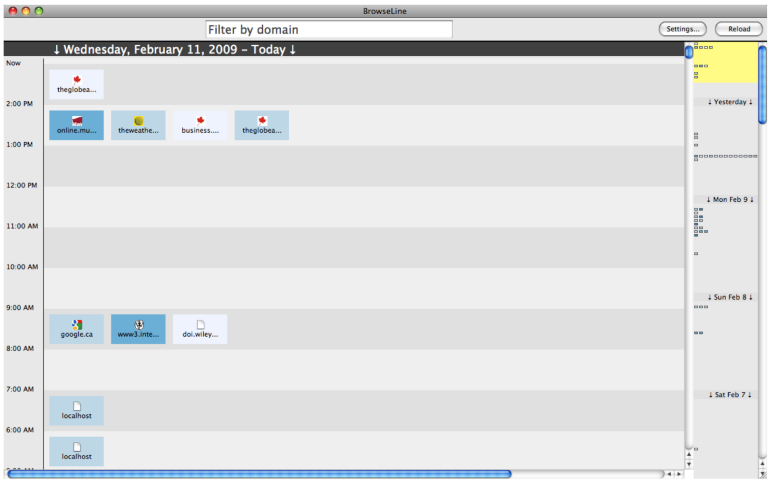

Figure 2.2: Screenshot of BrowseLine.

WebComets [CTKE13] includes parallel browsing. It stands for using multiple tabs in the web browser and therefore history entries are sorted in a standard view by their time without differentiating their origin. With this extension each tab has sort of own history and with the use of different graphical elements, many attributes can be included in the visualization (see Figure 2.3). Each line parallel to the x-axis represents a tab with its tail, which indicates the active time for a web page. The size of the glyphs illustrates the amount of sub-domains included. The pie chart itself visualizes with its degrees a ratio of the number of site calls of the current Web page compared to all domain-calls. To enable so much information about history items, the extension has its own logging because all browser integrated functions are not sufficient for the needs. Due to the complex illustration described above a user is not able to interpret all graphical elements without any instructions. Our goal was to offer a simplified interface that users can deal with without delving into a manual.

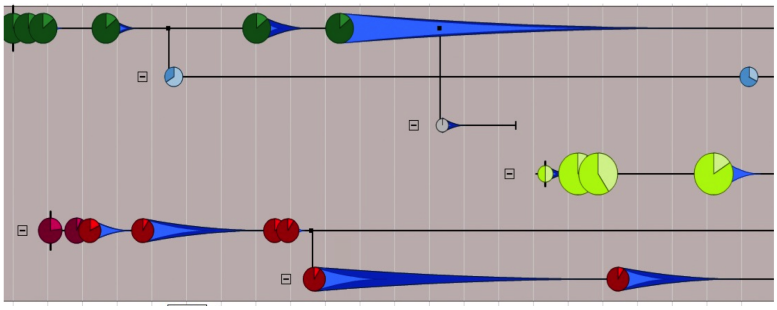

Figure 2.3: WebComets: Parallel browsing history.

2.2 History Visualization with different approaches

By trying to display as much information as possible, the approach to use the third dimension is implemented by some further extensions. The first extension described concentrates on connections between site calls without focusing on any time-line. The second one visualizes history elements in a volume to show more attributes. The last one tracks user behavior during a specified task on a website. This system is not designed for end customers but for web design teams to run usability tests on their implemented sites. With Trails [YI11] relations between site calls are displayed by lines. In Figure 2.4 the so-called 'Individual View' is shown and each ring represents a URL. At the bottom a day can be set and appropriate items will appear. The focus is not concentrated on an order of site calls but on how they are linked and therefore users can reconstruct their online behavior.

Figure 2.4: Trails: Individual view with relations.

The second application to be introduced is 3D based and called WebPath [FS98]. It can be used alongside normal browsers since it is an independent computer software to track and visualize a users web behavior. By visiting a web site the application builds a cube with information taken from HTML and displays it in a volume. Figure 2.5 shows how a user navigated from one site to another. He can reconstruct where he has been.

The last framework we want to describe is called WebQuilt [L01] and uses a proxy logger to capture all requests from the client browser to a web server. The system helps web designers to analyze how users interact with their web pages. The collected data will be visualized including all links and paths. Each website is represented by its screen-shot and Figure 2.6 shows how connections of twelve usage traces are displayed. This task has to be done by twelve users and the optimal path is shown with thick blue arrows. The other color-coded lines indicates how long the users have spent on a site and the thickness of those represents more heavily traversed paths.

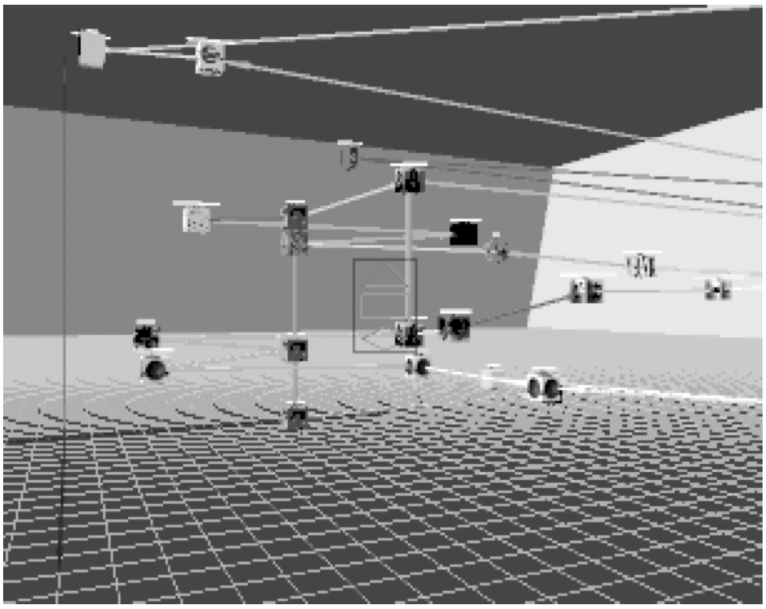

Figure 2.5: WebPath: Navigation from site to site.

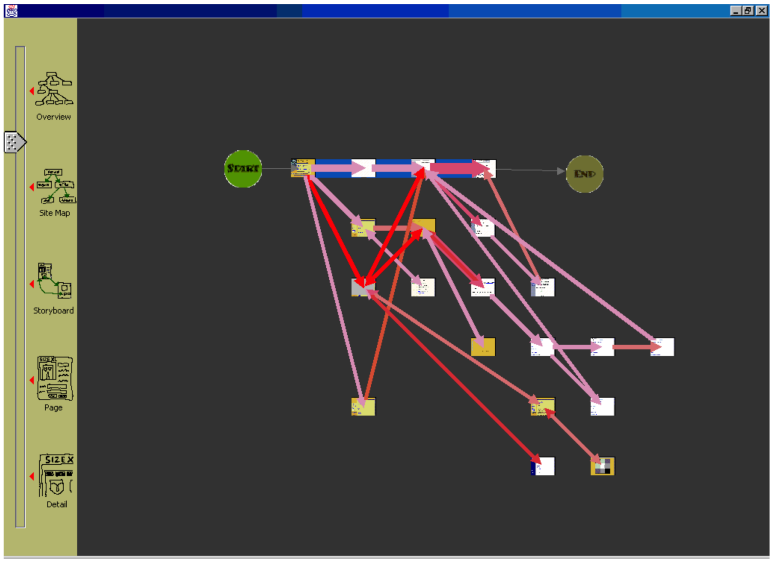

Figure 2.6: WebQuilt: Visualization of linked sites.

CHAPTER 3

The Chaser

The Chaser is the name of the Chrome extension we implemented. It visualizes the browsing history logged by Chrome itself and gives users an overview which pages have been visited. After discarding many design decisions the final interface can be seen in Figure 3.1. At the beginning an approach using a third dimension was considered. It was accompanied by complex handling and no advantage over a two dimensional illustration. The third dimension would be used as a time axis in z-direction. The hosts are sorted in circles which result in a tube. As the user goes back in time by following a domain trail, his/her view goes deeper and deeper in the tunnel. This type of illustration brings a great user experience as 3D can impress a lot but in general 2D is a more effective illustration and navigation technique [CM02]. The main disadvantage is a non or very low profit of the visualization because the circular mapping is not useful as it is not linear and therefore the number of domains around the circle is limited due to shortage of space. In Figure 3.2 a draft illustrates the 3D approach and the final appearance in comparison. An easier re-finding of sites is not guaranteed with the use of the third dimension as visual distortions can occur.

Finally we decided to focus on a chart with two dimensions. The extension uses all calls logged by chrome.history API [his] and filters after a user request which calls are displayed. There are three different levels of detail available including a year-, a month- and a day-view. Therefore a user has the possibility to start looking at a whole year and then focus on a desired month followed by a day. Proper descriptions about the navigation can be found in Section 3.3. Section 3.2 gives an overview on all design decisions and why the final product has its appearance.

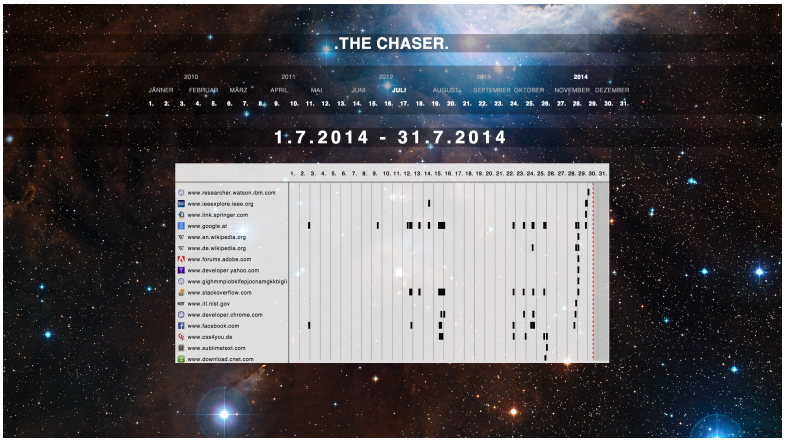

Figure 3.1: The Chaser: Final interface.

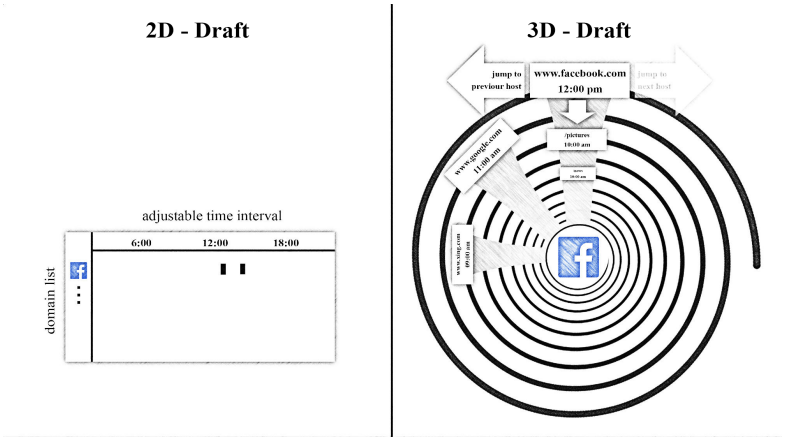

Figure 3.2: Comparison between 2D and 3D draft

3.1 History Data

To describe why and how the chart is illustrated with its design we have to look behind the curtain to understand what happens in the back-end. All site-calls are visualized by rectangles with exactly the same width. That is because chrome.history API [his] offers an attribute called 'visitTime' without further information about how long a user has been active on this site. How a list with those attributes is generated and how the mapping works in detail the Section 4.2 gives further information. The items which represent the URL-calls are sorted by their last occurrence in the current time interval from top to bottom. That means that the first line with its favicon and host name on the left side contains the last URL a user has opened. All site-calls are grouped by

their domain to reduce the length of the list and enable users to get a summary of the quantity of calls inside host categories. For comparison Figure 3.3 shows the list view from Chrome and how duplicated hosts can overload the view and make it even harder to locate a URL. Nowadays users click through hundreds of pictures in a few minutes especially on Facebook. This results in extraordinary long lists and the standard view of Chrome's history is overcrowded. The Chaser tries to compensate this by its grouping algorithm described above and in Chapter 4.

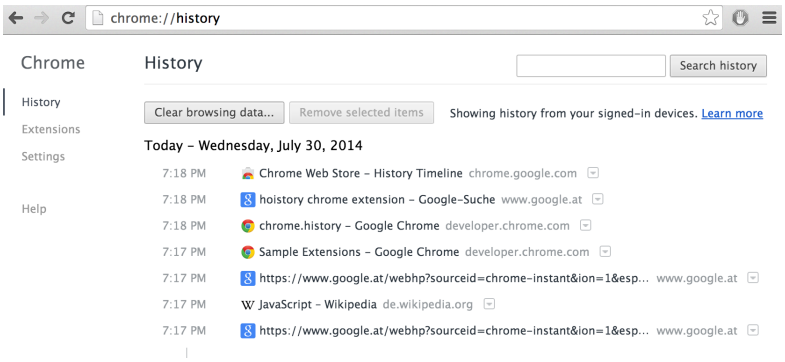

Figure 3.3: Chrome standard history-view.

3.2 Visualization

It is important to give users the ability to navigate simply and predictably. The entire interface consists of one site which is displayed after opening the standard Chrome history view as it is overwritten by this extension. The goal was that no instructions are necessary when using the history. The extension has an overview and detail approach since a whole year can be requested by a user but the zoom level is also possible to be extended to one day. The appropriate history items are illustrated in a chart positioned in the center of the page. At the top of the page an overview and detail navigation is displayed to interactively change the level of detail (see Figure 3.4). That means that a user can choose any displayed year and if s/he wants to refine the view, the zoom level can be adjusted until the day-view is reached. This navigation illustrates the last five years but with the addition to go back in time further by using the arrow keys. In our opinion year-views can be really interesting as other extensions like for example BrowseLine [HG09] do not offer something similar. With The Chaser users can detect in which months they have been more active than in others and which domains have been used most. The displayed time interval in the middle of the page is in addition to the highlighted elements at the top and is illustrated in Figure 3.5.

Figure 3.4: The Chaser: Navigation and date-overview.

Figure 3.5: The Chaser: Information with visible time interval.

By viewing the current year, month or day a red vertical line crosses the chart. It symbolizes the point of time the extension has been opened or refreshed. The area shaded in grey represents the future and therefore no elements are visible there. At the top of the chart a legend with measurement unit is displayed and depending on the actual view one of the following variations is used for labeling. Figure 3.6, 3.7 and 3.8 display all possible indexes.

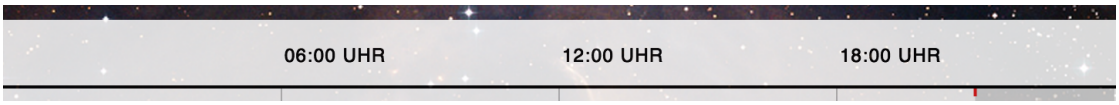

Figure 3.6: The Chaser: Label for day-view.

Figure 3.7: The Chaser: Label for month-view.

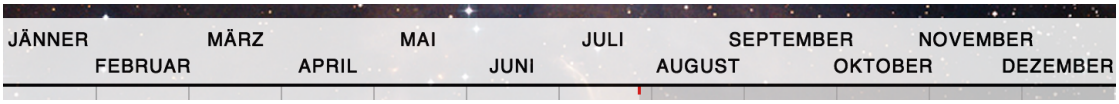

Figure 3.8: The Chaser: Label for year-view.

3.3 Interactive Exploration / Navigation

To support the navigation within The Chaser the cursor style is adjusted to show functionality of clickable text. If text elements are only for descriptive purposes, the cursor does not change its appearance but as an item works as well as a link, a pointer is shown. Another functionality implemented by us is a mouse-over function. By hovering over rectangles inside the chart the appropriate links will be displayed beside them. After clicking on the rectangle a new tab with this URL will be opened. Users have also the alternative to click directly on an favicon or a host group on the left side to open this host in a new tab. In a previous design draft a preview was included. By moving over rectangles a small screen-shot is shown to give users an idea about the content of this site. The main reason for not implementing this functionality is, that Chrome only saves screen-shots of several URLs. An extension described earlier called WebPath [FS98] tries to include background color and images of the pages in its visualization.

Users have the choice to navigate only with their mouse through the extension or use the keyboard for some of the functions. It is not possible to handle everything with keys and the following list shows the range of functions available. The arrow key to the left displays the year, the month or the day before depending on the current view. It is implemented with a fluent animation with the aid of jQuery [jqu], a JavaScript library. The code can be seen here: 4.3.3. The right-key does exactly the same in the opposite direction. With the use of the up-key, the zoom level is scaled down but the reverse way does not work as it is not clear which month or day should be illustrated. The last key supported by The Chaser is the n-key. By clicking it, the current view jumps to the present point in time. The rest of functionality users have to resort to the mouse navigation.

CHAPTER 4

Implementation

The following chapter describes which technologies are used to implement this Chrome extension. It is a combination of different programming languages to cover all needs. An overview can be seen in Figure 4.1. Some of them are working in the background, whereas others are used for the front-end to display everything accurately. All parts are united in a json file called manifest. It contains information about the plug-in as well. A proper description can be found in Section 4.1. Further discussion of the individual parts will be done in the next sections.

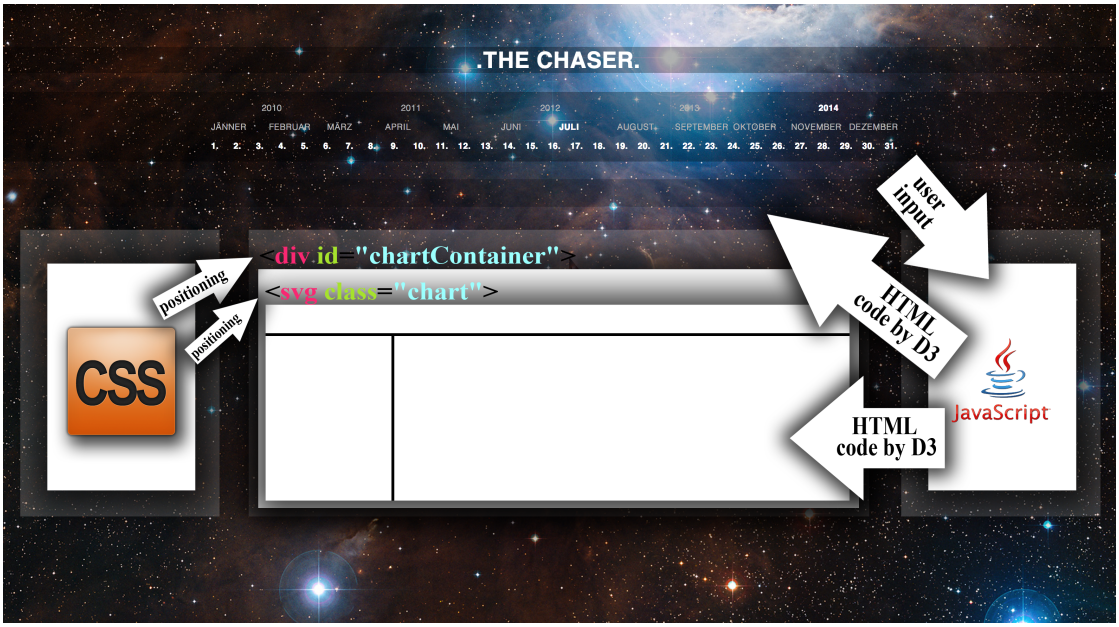

Figure 4.1: The Chaser: Overview of used technologies.

4.1 Setup

At the beginning of implementing a Chrome Extension, the `manifest.json` 4.1 has to be written. The entire file is listed below. Some fields are required and others are just optional. The name and the version is mandatory and as The Chaser wants to get access to the history items of Chrome, correct permissions have to be set. Besides history and favicon the fileSystem is responsible for saving the records persistent on the clients computers during the logging for the evaluation. It is followed by the initialization of the icons for the plug-in with different pixel dimensions. After this the extension overwrites the standard history page of the browser with its own html file. The final important field contains necessary scripts which are described here: 4.3. `Manifest.json` 4.1 is for all intents and purposes like a skeleton of the whole extension to ensure correct accesses to all files.

```
{
  "name": "THE.CHASER.-.History.Visualization",
  "version": "1.0",
  "description": "Reads.your.history.",
  "content_security_policy":
        "script-src'self''unsafe-eval';object-src'self'",
  "permissions": [
    "history",
    "chrome://favicon/",
    "fileSystem"
  ],
  "icons": {
    "48": "icon48.png",
    "128": "icon128.png",
    "265": "icon265.png"
  },
  "chrome_url_overrides" : {
    "history": "historyVis.html"
  },
  "background": {
    "persistent": false,
    "scripts": ["d3.min.js","historyVis.js",
        "jquery.min.js", "jquery.easing.1.3.js"]
  },
  "manifest_version": 2
}
```

Listing 4.1: manifest.json

4.2 Data Mapping

The function `buildUrlList()` and `completeList()` shown in listing 4.2 gathers all the history entries saved by Chrome itself. Chrome's history API provides a `chrome.history.search()`-method to query a sorted list including `HistoryItems`. This method has some parameters to be mentioned. The first `text` attribute can search the history list matching this string. To get all entries, an empty value is assigned. `maxResults` offers the possibility to determine a specific size of the returned list and to get all items without an interval limit `startTime` is set to '0'. The last parameter to be set is a callback function including an array of `HistoryItems` which have several properties of one URL. To get all visit-times of this URL, the function `chrome.history.getVisits()` returns all `VisitItems` encapsulating one visit to a URL. Every site-call is saved in a list called `historyList`. `visitTime` is a property contained by those elements and represents when this visit occurred in milliseconds since the epoch which starts at 01/01/1601 00:00:00. To map all items accurately the difference between this visit-time and the present time is calculated and due to the fact that the maximum zoom level is one day, time-stamps are converted to minutes. According to this, they represent minutes from present to the point of time a page was loaded. This is realized in the function called `calculateTimeStamp()` 4.2. The exact procedure to map this elements is described later. The function `buildUrlList()` 4.2 is called once a page is loaded as all entities are saved in an array which is sorted by time-stamps at the end of the function `completeList()`. The `historyList` is used for later queries and requests. The advantage over working along with this assembled array is that several custom-calculated values have to be computed and set only one time.

```
1   function buildUrlList(divName, timeInterval) {
2
3     var historyListTMP = new Array();
4     var url2 = "";
5
6     chrome.history.search({'text': '',maxResults: 1000000,
7       'startTime': 0}, function(historyItems) {
8
9       historyList.length = 0;
10      historyListTMP.length = 0;
11      url2 = historyItems[0].url;
12
13      for (var i = 0; i < historyItems.length; ++i) {
14
15        ...
16
17        historyListTMP.push({
18          'url':historyItems[i].url,
19          'title': historyItems[i].title,
20          'host': getHost(historyItems[i].url),
21          'groupNumber': getGroupNumber()
22        })
```

```
23
24            if(i == historyItems.length-1) {
25              completeList(historyListTMP);
26            }
27          }
28        });
29      }
30
31      function completeList(historyListTMP) {
32
33        var url2 = "";
34        var title2 = "";
35        var host2 = "";
36        var groupNumber2 = "";
37
38        for (var i = 0; i < historyListTMP.length; ++i) {
39
40          url2 = historyListTMP[i].url;
41          title2 = historyListTMP[i].title;
42          host2 = historyListTMP[i].host;
43          groupNumber2 = historyListTMP[i].groupNumber;
44
45          chrome.history.getVisits({"url":url2}, function(visitItems) {
46
47            for (var j = 0; j < visitItems.length; ++j) {
48
49              historyList.push({
50
51                'timeStamp': calculateTimeStamp(visitItems[j].
52                  visitTime),
53                'url':url2,
54                'title': title2,
55                'host': host2,
56                'hostID': getURLid(url2),
57                'groupNumber': groupNumber2,
58                'lastVisitTimeReal': ( visitItems[j].visitTime)
59              });
60            }
61          })
62        }
63        var listSorted = historyList.slice(0);
64        listSorted.sort(function(a,b) {
65          return a.timeStamp - b.timeStamp;
66        });
67
68        historyList = listSorted;
69      }
```

Listing 4.2: Function buildUrlList().

```
function calculateTimeStamp(oldTimeStamp) {
    var t = (now - oldTimeStamp) / 1000 / 60;
    return t;
}
```

Listing 4.3: Function calculateTimeStamp().

After a user requested a time interval to be displayed an appropriate list named `listToDraw` will be filled-up with `calcListToDraw()` 4.2 (see Figure 4.1). A variable called `timeInterval` has to be set before. It represents the requested range of view in minutes. If a user calls for a specific month to be displayed, the suitable `timeInterval` has to be calculated to guarantee that all entries added to `listToDraw` are correct. The time stamp of all entries has to lie inside the specified interval which is checked with the first `if`-statement. Next a list is filled with appropriate attributes including all items from `buildUrlList()` 4.2 and in addition a `groupNumber`. This value is calculated with the function `setGroupNumber()` 4.2. All URLs with the same host are combined to one group and have the same `groupNumber` which starts at zero. The importance of this step and why it is necessary is explained later.

Figure 4.1 gives a better understanding of the used technologies and how the JavaScript file receives a request triggered by a user input. To complete the mapping the computed `listToDraw` 4.2 has to be integrated dynamically in the SVG-element included by the HTML file. The chart container which will contain all history items is pre-positioned with absolute pixel values in the HTML by using CSS. It is important and necessary that the width has exactly 720 pixels to guarantee an accurate positioning of the history entities. To draw all elements inside this container a JavaScript library called D3 [BOH11] is used which is described in Section 4.3.4. To calculate positions of the rectangles, the extension uses the requested interval and all time stamps. To display one day, with 24 hours, as an example, there are 1440 minutes to be shown within 720 pixels. For this reason one pixel illustrates two minutes. Other display modi, the month or the year view, have different minute to pixel conversations. As it is mentioned above, groups contain different URLs but with the same domain. They are represented on the y-axis and all elements of them are drawn in the same row with their computed x-values. To determine the correct x-positions time stamps are used and the implementation of this method can be seen in Listing 4.3.4. Due to the fact that URLs with different endings but same hosts are treated as one group, the user has the ability to get an overview on how often different hosts from one domain have been visited.

```
function calcListToDraw(completeList) {
  listToDraw.length = 0;
  for (var i = 0, ie = historyList.length; i < ie; ++i) {

    if(completeList[i].timeStamp + nowPosition <= (timeSteps
      + timeIntervall) && completeList[i].timeStamp +
        nowPosition >= timeIntervall) {
      setGroupNumber(completeList[i].host, listToDraw);
      listToDraw.push({
```

```
        'timeStamp': ((completeList[i].timeStamp)) ,
        'url':completeList[i].url,
        'title': completeList[i].title,
        'host': completeList[i].host,
        'groupNumber': getGroupNumber(),
        'lastVisitTimeReal': completeList[i].lastVisitTimeReal});
    }
  }
}
```

Listing 4.4: Function calcListToDraw().

```
function setGroupNumber(host, hisList) {
  for (var i = 0, ie = hisList.length; i < ie; ++i) {
    if(hisList[i].host == host) {
      groupNumberTmp = hisList[i].groupNumber;
      return;
    }
  }
groupNumberTmp = groupCounter;
  groupCounter++;
}

function getGroupNumber() {
  return groupNumberTmp;
}
```

Listing 4.5: Group number calculation.

4.3 Used Technologies

The following sections give a short introduction of all technologies which are used. Some of them are front end parts and others handle all the data in the background.

4.3.1 HTML

HTML is responsibly for the general structure of the extension. In combination with CSS, 4.3.2, an exact positioning and sizing of DIV- and SVG-containers is possible as it can be seen in Figure 4.1. History related elements and denotations will be loaded and integrated dynamically during run-time. To get a better overview the main part of the HTML-code can be found here: 4.3.1. With the assistance of D3 [BOH11], SVG-contents can be changed dynamically from the JavaScript file. With this approach, the general user interface can be designed and configured in the HTML-file without taking care of the mapping of the data. Neither rectangles nor favicons or other history related data is pre-positioned within the HTML-code at first.

```
...
<div id="container">

  <div id="timelineContainer">
    <svg class="timeline"></svg>
  </div>

  <div id="chartContainer">
    <svg class="chart">
      <svg class="nowLine"></svg>
      <svg class="grid"></svg>
    </svg>
  </div>

  <div id="faviconContainer">
    <svg class="favicon"></svg>
  </div>

</div>
...
```

Listing 4.6: historyVis.html

4.3.2 CSS

CSS stands for Cascading Style Sheets and is a markup language to change the formatting and the design of an HTML site. In this Chrome Extension it is used to give all containers exact measurements and is responsible for all font and color settings. Without CSS The Chaser is not usable in case of all elements predefined in the HTML-file are not positioned correctly.

4.3.3 jQuery

jQuery [jqu] is a JavaScript library offering general JavaScript/Ajax functionality, as it is necessary to animate whole DIVs. All transitions of The Chaser are made with jQuery. Whether a user wants to change the view to an earlier or future time interval by clicking the left or the right arrow key, a nice and smooth animation will guide to the next field of view. At first the DIV-element shifts to the right to disappear. If it is positioned 900 pixels from the left, it is not visible any more. Next it is positioned on the left side and the content is refreshed to be shifted afterwords back in the view. The javaScript-code can be seen in Listing 4.3.3. D3 is not used for this as an animation with many history items is more complicated to implement than just shifting the whole DIV including all items. The difference between these two libraries is that D3 offers data-driven functionality to create and manipulate visual documents and jQuery is a more general JavaScript library offering functions for basic tasks.

```
function animateLeftRight() {
  $(chartContainer).animate({
    left:'900px'
  });
  $(chartContainer).promise().done(function(){
    positionBoxLeft();
    ...
    refreshView();
  });
  $(chartContainer).promise().done(function(){
    $(chartContainer).animate({left:'250px'});
  });
  return;
}

function positionBoxLeft() {
  $(chartContainer).css({left: '-720px'});
}
```

Listing 4.7: Functions for DIV animation.

4.3.4 D3

The main purpose of D3 [BOH11] is the visualization and integration of dynamic data. D3 enables a direct manipulation of the document object model (DOM). By selecting different classes of SVGs initialized in the HTML file, developers can bind arbitrary data to them. D3 is a JavaScript framework and is really fast even with very large data sets. Therefore it is perfect for a history visualization as the list size of visited pages can be a four-digit number or more. To get a better understanding on how D3 works, the following excerpt of `historyVis.js` 4.3.4 is listed below. First a SVG element has to be selected to bind data to it. In this case after selecting .chart, rectangles and text items are appended to it. How this works will be discussed in the following section.

```
...
  var chart = d3.select(".chart")
  .attr("width", width)
  .attr("height", heightOfBox);
...
```

Listing 4.8: Selecting a DIV with D3

At the beginning of integrating elements to a SVG an ID will be assigned to them which are the URLs of the history items. Additionally the class `clickAble` is set as an attribute to track all clicks with an event listener. In that case it will be possible to jump directly to a located site by just clicking on the rectangle or the displayed URL. After adding several attributes, including a translation in y orientation to focus on the appropriate domain row, positioning of the rectangles and text items in x-direction is the next step to perform. The code snippet at 4.3.4 shows how the x-positioning is done

with different variables and how time stamps of the elements are used to convert minutes to pixel values. `timeDivisor()` returns a value according to the current level of zoom to map the elements within the displayed box. The attribute `width` has the value 4 and therefore all rectangles are exactly 4 pixels wide. Every item from `listToDraw` runs through this code section until every history element is displayed. These are only a few potentials of D3 since this library offers many other functions such as smooth animations and interaction. All transitions included in the extension are made with jQuery [jqu], a JavaScript library which is described in Subsection 4.3.3.

```
...
var bar = chart.selectAll("g")
  .data(data)
  .enter().append("g")
  .attr("id" , function(d, i) { return elements[i].url; })
  .attr("class" , "clickAble")
  .attr("transform", function(d, i) { return "translate(0,"
   + (elements[i].groupNumber * barHeight) + ")"; });

bar.append("rect")
  .attr("x", function(d , i) { return (chartWidth -
    (elements[i].timeStamp - timeIntervall + nowPosition)
    / timeDivisor()); })
  .attr("width", 4)
  .attr("height", barHeight - 10)
...
```

Listing 4.9: Append data to a selected SVG.

4.4 Limitations

Google Chrome has many features included in its chrome.history API [his]. Adding and removing of URLs is well supported. The extension The Chaser displays all site calls with rectangles exactly the same width. That is because Chrome offers the attributes `lastVisitTime` and `visitItem`. The Chaser uses the second value and therefore it is not possible to illustrate for how long a user visited a page. Another interesting thing to display would be the paths from one site to another by clicking a link. It means that a user can go back in time the exact way the URLs have been called and it is illustrated from which page he was forwarded to another one. An approach to realize this could use `visitItem` mentioned above. `visitItem` has properties called `referringVisitId` and `transition` which describe how a user was forwarded from a site and what the unique identifier of the first web page was. By using these two values paths could be illustrated but a problem occurs if users start to browse in multiple tabs. The graphical interface of The Chaser is not able to display several trails without loosing its overview. WebComets [CTKE13] is a tool described in 2.1 whose visualization displays all paths within one browser window. Another program to visualize all user behavior within one website is called VISVIP [CS99]. It gives developers more flexibility to evaluate and test

their own sites by getting a better insight in users online activities and how they move in the website. It is possible by using the third dimension, which illustrates the time. Figure 4.2 displays how the interface looks like.

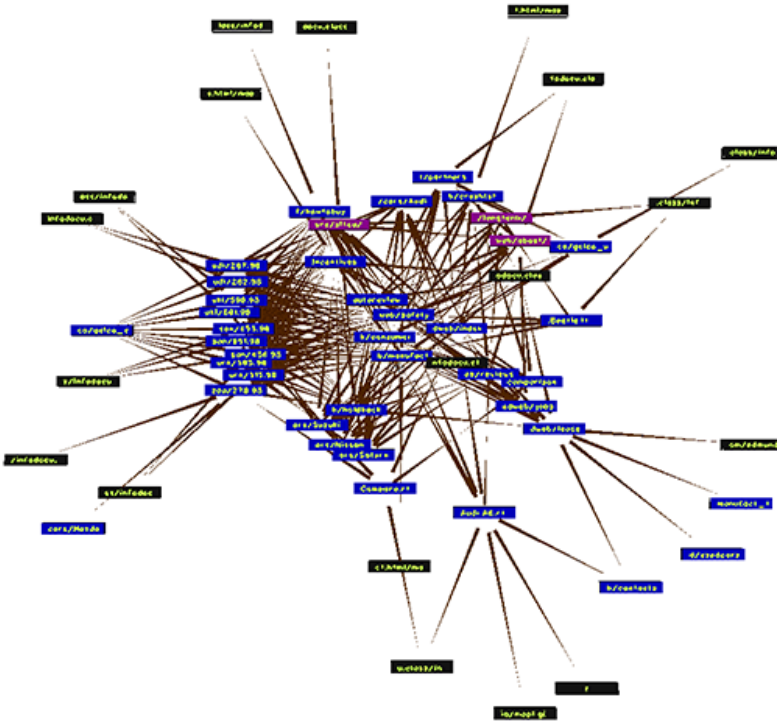

Figure 4.2: VISVIP - A highly connected website.

CHAPTER 5

Evaluation

The Chaser can be used to replace the standard-history-view of the Chrome browser by giving a compact time-line-based representation. In this chapter an evaluation is described. The aim of the extension was implementing a system to help users re-finding their URLs in an easier way and giving them an overview of their browser behavior. The effectiveness was considered in this evaluation. Another goal was to determine whether the interface and navigation is self-explanatory and users do not have problems when using the history. During evaluation six people used The Chaser for a specific time and the list view of Chrome afterwords for the same period. By logging different clicks it can be detected how and how often our extension was used in comparison to Chrome's build in visualization. In addition the test person gave an informal feedback of the benefits or disadvantages in their opinion and if they would prefer The Chaser instead of Chrome's built in list-view. In the following Section 5.1 the evaluation is described in detail and in Section 5.2 the results are presented.

5.1 Evaluation Method

At the beginning of the evaluation six people were selected. All of them are between 23 and 26 years old and use a computer and the internet at least once a day. Two of them are studying software engineering, one is studying marketing and three are working in a non-IT-related company. The female male ratio is five to one. The complete evaluation took two weeks which consisted of a one week test phase using The Chaser and in the second week the same subjects had to use a list-view history visualization. Both systems contain a logging function to save how they interacted with the illustrations. The task for all users in both weeks was to use the browser and its history like they would do it normally and give a informal feedback after the test phase on both histories in comparison. The feedback-session took about twenty minutes each and included several questions in general about The Chaser and difficulties or other

disadvantage as well. It was a face-to-face survey and the conversation was recorded and analyzed afterwards. Asked questions were for example 'Did you have problems using the extension in case of navigation problems?' or 'Was the interface and the displayed chart self-explanatory?'. Besides the oral feedback the logging files point out the usage in numbers. As an example, list 5.1 shows a fragment of a logging-file. The point in time they opened the visualization and furthermore all clicks within the extensions were recorded. In the listing beneath the user has clicked on a favicon, an URL and other buttons as well.

```
1 ----------------------------
2         "new session"
3 Wed, 30 Jul 2014 13:39:57 GMT
4 ----------------------------
5 backward key pressed
6 favicon: "http://www.stackoverflow.com clicked"
7 up key pressed
8 url: "http://chrisboesch.at/ clicked"
```

Listing 5.1: Part of a logging file.

5.2 Results

After collecting all log-files we made two charts representing all usage statistics from The Chaser and Chrome's list view, see Figure 5.1 and 5.2. The height of the bars indicates how often the histories have been opened. In total the standard view was used ten times and for comparison The Chaser 46 times. Our extension was opened about five times more often than Chrome's history. This indicates that The Chaser offers more interesting facts or easier re-finding algorithms because it is more attractive to the users. To figure out how they were satisfied with the navigation or the overview the feedback gives further information. 'Thanks to the appealing and well-arranged design of The Chaser I can now easily browse my history whereas the standard interface would be too confusing. Especially the timeline-feature was extremely helpful if I needed some in-depth information.' These are the words of a proband and indicates the fact that the aim to make the history more usable and compact is achieved by The Chaser. Others say that they don't use the standard view at all and now they have the possibility to get a great overview of their online history. One feedback contains disadvantages. Users cannot adjust the width of the rectangles to see all individual URL calls if they are lying too close resulting in one big element. Re-finding was not covered by any feedback because either it is not relevant for the users or the illustration does not support this well. Therefore we asked a question additionally: 'Does The Chaser offer an easier way re-finding previously visited pages compared to the standard list?' The given answers reveal that revisitation for most of the users is not a mandatory functionality. If they know parts of their browser history re-finding of a page with our extension is easier because URLs are grouped by domains and shown in a temporal visualization. After asking if they will use this extension in the future, some answered that they do not need

any history visualization at all and others said that they like the better overview and will use The Chaser in the future as well. To sum everything up we conclude that our Chrome extension gives a better overview of the user's history in different temporal levels of zoom and that they would prefer this extension against Chrome's standard visualization.

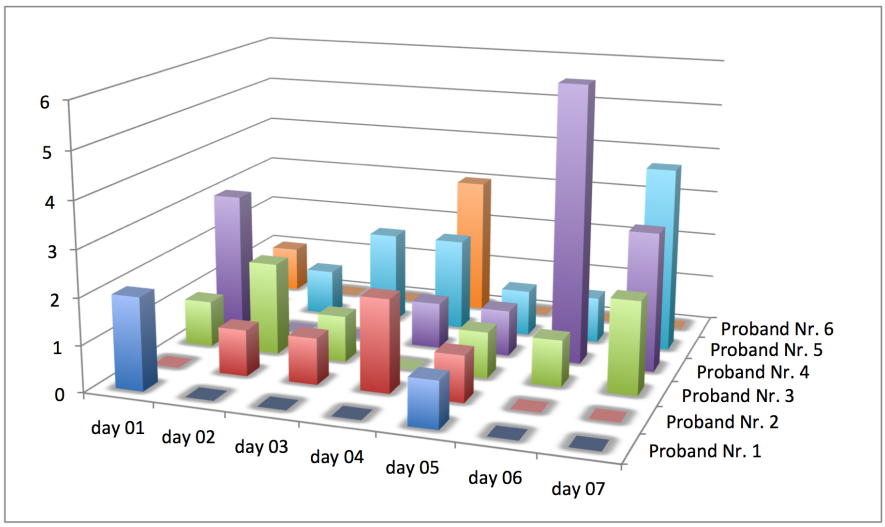

Figure 5.1: Chart with usage of The Chaser.

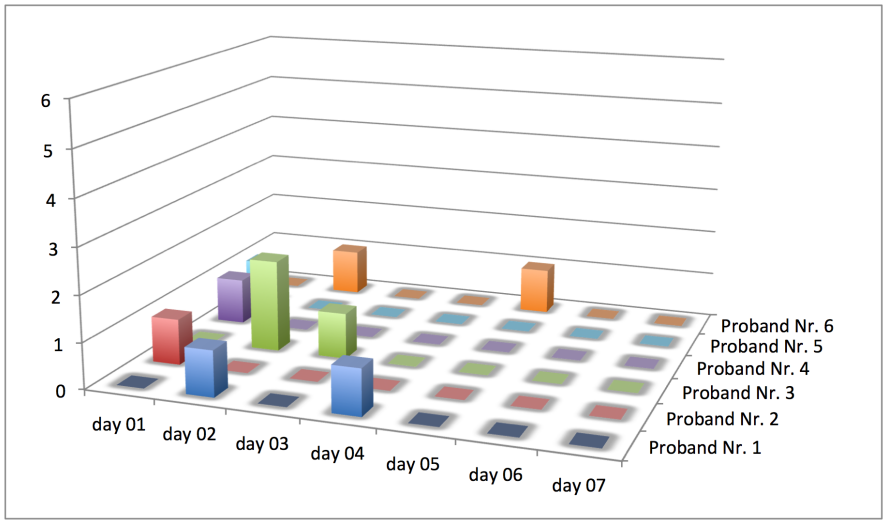

Figure 5.2: Chart with usage of Chrome's list view.

CHAPTER 6

Conclusion and Future Work

We presented an alternative visualization for Chrome's history view. It is implemented with a time-line-based approach in two dimensions. All URL calls are grouped by their domains and listed in the y-axis. The x-axis represents the time and gives the user the ability to change the level of detail from one day to one year. The main focus was on creating a compact illustration by which a user is supported re-finding a visited page and and getting a good overview. The implementation contains the functionality to display all URLs in a day-, a month- or a year-view. Different time-intervals can be queried but manipulation or filtering of the data in more detail is not supported. As an example, the domains listed on the left side including their URL-marks could be blanked out by the user. That means, that if s/he wants to hide several columns, boxes in front of them can be ticked and these groups are removed and showed up in another chart, where all hidden domains are displayed. Another improvement can be that by hovering over rectangles or hosts, this line will be highlighted to facilitate the correlation from domain to rectangles and vice versa. The Chaser has one disadvantage worth mentioning. It deals with the illustration of paths between site calls. With this extension, a user cannot view trails or jump from the last opened URL to the site before. Due to the fact that our extension uses this time-line-based visualization, problems with the loss of the overview occur and the illustration will be overcrowded by elements. To realize this, the interface would have to be redesigned and the appearance of The Chaser would change completely.

Bibliography

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics*, pages 2301–2309, 2011.

[CM01] A. Cockburn and B. McKenzie. What do web users do? an empirical analysis of web use. In *International Journal of Human-Computer Studies*, pages 903–922, 2001.

[CM02] Andy Cockburn and Bruce McKenzie. Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–210, 2002.

[CS99] J. Cugini and J. Scholtz. VISVIP: 3d visualization of paths through web sites. In *Tenth International Workshop on Database and Expert Systems Applications*, pages 259–263, 1999.

[CTKE13] Daniel Cernea, Igor Truderung, Andreas Kerren, and Achim Ebert. WebComets: A tab-oriented approach for browser history visualization. *International Conference on Information Visualization Theory and Applications*, pages 439–450, 2013.

[FC00] S. Fernandes and T. Catarci. Visualization of linear time-oriented data: A survey. In *Proceedings of the First International Conference on Web Information Systems Engineering*, volume 1, pages 310–319, 2000.

[FS98] E. Frecon and G. Smith. WEBPATH: A three dimensional web history. In *IEEE Symposium on Information Visualization*, pages 3–10, 148, 1998.

[HG09] O. Hoeber and J. Gorner. BrowseLine: 2d timeline visualization of web browsing histories. In *Information Visualisation - 13th International Conference*, pages 156–161, 2009.

[his] chrome.history - google chrome. URL: https://developer.chrome.com/ extensions/history. Accessed: 2013-07-28.

[jqu] jQuery: Javascript library. URL: http://jquery.com. Accessed: 2014-07-30.

[L01] James A. L. WebQuilt: A framework for capturing and visualizing the web experience. In *Proceedings of the 10th international conference on World Wide Web*, pages 717–724, 2001.

[MS03] W. Muller and H. Schumann. Visualization methods for time-dependent data - an overview. In *Simulation Conference*, volume 1, pages 737–745, 2003.

[MVK10] Brennan Moore Max Van Kleek. Eyebrowse: Real-time web activity sharing and visualization. *CHI EA - Extended Abstracts on Human Factors in Computing Systems Pages*, pages 3643–3648, 2010.

[TG97] Linda Tauscher and Saul Greenberg. How people revisit web pages: empirical findings and implications for the design of history systems. In *Int. J. Human-Computer Studies*, pages 97–137, 1997.

[YI11] Wenhui Yu and Todd Ingalls. Trails: An interactive web history visualization and tagging tool. In *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, number 6770, pages 77–86. 2011.