UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Science and Technology
Department of Computer Science

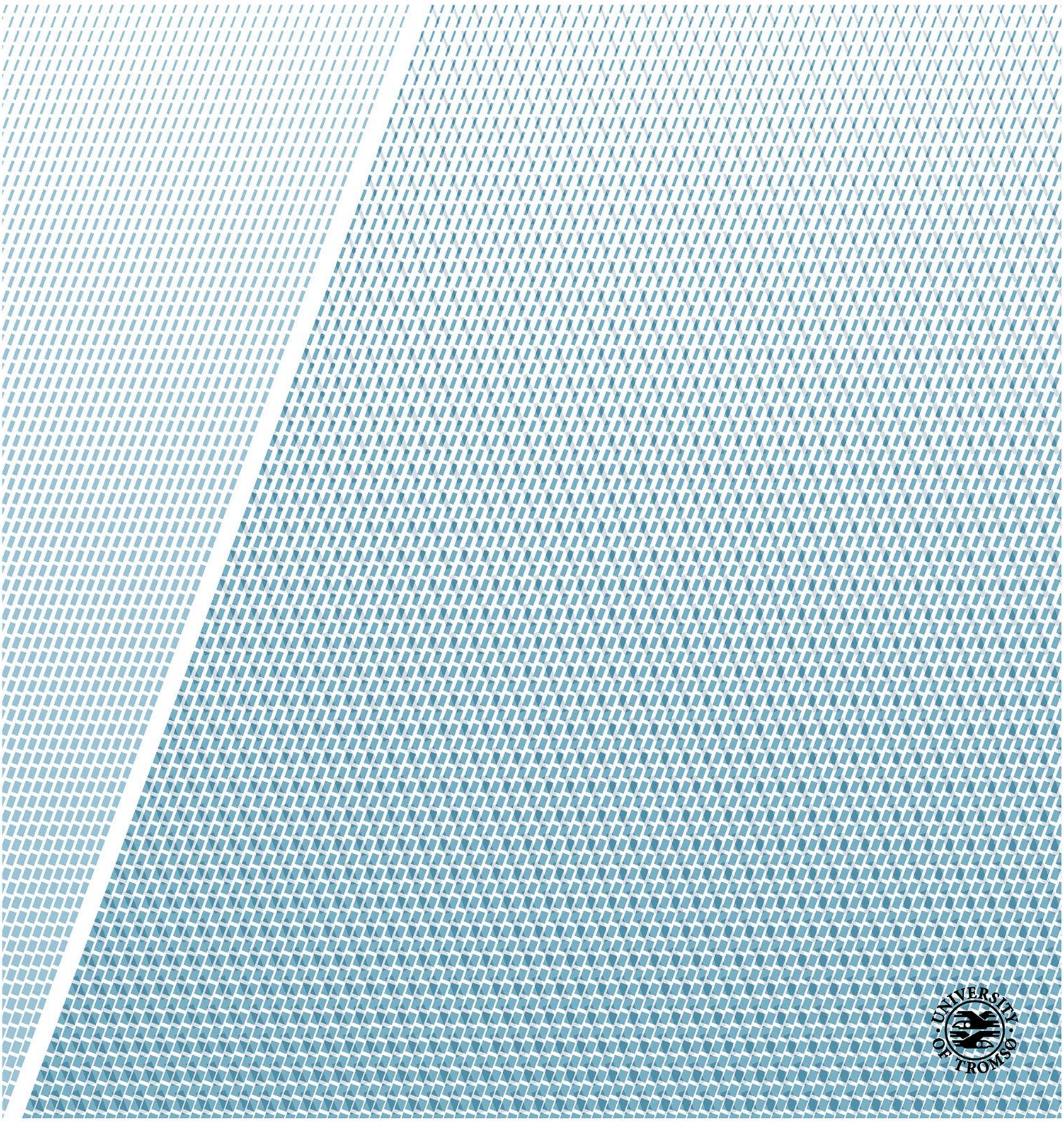
Extending the Chrome browser

Adapting browser functionality to user needs and behavior

—

Adrian A. Skogvold

INF-3990 Master's Thesis in Computer Science - November 2015



Abstract

What today's browsers offer in customization, personalization and functionality can be improved. In this thesis a application, in the form of a chrome extension that extends a web browser's functionality, and that offers customization and personalization has been designed and implemented. The functions implemented in the extension simplifies the access and availability of information to the user through user inputs and monitoring usage history. The extension offers customization by allowing users to edit the visuals and the amount if information provided.

This extension has been evaluated by a group of test users and the results of this evaluation is presented through tables and discussion and shows a diversity of improvement suggestions. Some suggestion was repeated by several users, but most of the suggestions was unique. This indicates that personalization and customization is an important part of an application. The results of the evaluation also serve as an indication to the usability of the functions in the extension and fulfillment of the goal in this thesis.

Acknowledgments

First I would like to give my thanks to my supervisors Professor Randi Karlsen and Professor Anders Andersen for their guidance, good advice and availability.

I would also like to show my appreciation to my fianc Stine E. Enge for being supportive and keeping me motivated through the work on this thesis.

And finally I would like to thank the test users who took their time to try out my application and evaluating it. They came up with a lot of good ideas that turned out to be very valuable.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Today there is a widespread use of the internet using web browsers. And there exists a lot of web browsers that have a somewhat different design. Users can also expand their browsers functionality through extensions. Still, browsers and the functionality they offer is not sufficient.

Bookmarks is a common function embedded in web browsers, but after some time the increasing number of bookmarks can be difficult to manage. A user then needs to organize the bookmarks in several folders since there is no automated function for this. An automated grouping function could sort them based on context, such as where the bookmark was added, so that similar pages go together.

There is a huge page diversity today. Web pages try to differentiate themselves with a unique look at layout. Even pages that serve similar purposes have common functionality hidden away in different locations to separate themselves from the crowd, but might just end up confusing their users. Because of this the users cannot recognize every page because they are all different from each other. To improve user-friendliness considerably a function that ignores the differences in layout of each page, could be useful. This function would have to make a collection of similar functions across several pages and present it in a similar manner regardless of the specific page visited.

The browser history of today's web browsers can be overwhelming. It is a large list sorted by default in chronological order. Over time it becomes quite large and full of duplicates. The problem with it can be that there is too much information for the average user to look at. A very small part of the users(0.1%) is actually using the browser history to access web pages [10] [4].

Creating something that should work the same way in different environments can be a challenge. It is a challenge because there are many factors to take into account and many things that can react in unexpected ways. Figuring out how to counteract these while interacting with the functionality of a page to perform customized functions will require a lot of work and attention to detail. Being able to predict the results of a function would require knowledge that can only be obtained by testing and observing.If this could be done successfully, it would create the feeling of learning something new every day. An environment like this encourages testing stupid ideas as knowledge would be gained from doing so. This knowledge can create new ideas that might improve the application. "The difference between screwing around and science is writing it down."[1]

1.2 Goal

The goal of this thesis is to design, implement and test an application that explores personalization, customization and adaptation possibilities in a web browser. This will include changes requested by the users in a settings menu and what the application could adapt to, based on usage history collected from the user. The base functions of the application will try to resolve issues with current web browsers mentioned in the previous section.

To be able to collect usage history the application will need access to the user's web browser. In addition, it will have to store the information gained from monitoring the user's web browser. The application will implement functions that use this information to offer solutions to some of the issues mentioned in section 1.1. For the remainder of the issues the application will have functions that require user interaction to provide solutions.

The application developed was tested among a small test group. The evaluation filled out by the participants is discussed in the end of this thesis.

1.3 Approach

The application developed in this thesis is a chrome extension. The functions implemented into it are all in some way available to the user in another form. The main improvement provided by the extension is that it simplifies the access of information and usage of the functions. Half of the functions

[1] `http://www.goodreads.com/quotes/639268-the-difference-between-screwing-around-and-science-is-writing-it`

are automatic in the sense that they need no user interaction to present information or services to the user. The other half requires user interaction to provide their service. These functions do supply a simplified way of storing and presenting their information compared to the functionality they are based upon. The functions implemented are discussed in more detail in section 3.

The reason for the application developed in this thesis being a browser extension is because they are attached to a web browser. Extensions can then be allowed to monitor, collect and store the users usage history. The application being an extension also makes it possible to implement functions that directly influence the functionality of the browser Which is one of the characteristics needed for the application to fulfill the goal of this thesis. The alternative to creating an extension is to create a new browser, but that would require a large amount of work that would not contribute to solving the goal of this thesis.

The reason for developing the extension for the chrome browser is because of personal familiarity with that browser and because, as of September 2015, it is the most commonly used browser[2].

1.4 Contribution

This work has explored the possibilities of personalization, customization and adaptation in the chrome browser by creating an extension.

The contribution of this thesis is a chrome extension built on ideas to improve the browsing experience. This is achieved by implementing functions that adapts to the user's needs and behavior either by automatic personalization based on usage history or customization based on user interaction.

This thesis also contributes with a user study and an evaluation of the extension developed in this thesis and the functions implemented into it.

[2]`http://www.w3schools.com/browsers/browsers_stats.asp`

1.5 Structure

The rest of this thesis is structured as follows. Some important background information and technologies used is presented in chapter 2. The design of the system is presented in chapter 3, followed by the implementation of the system in chapter 4. The evaluation done by the test users and the results of it is presented in chapter 5. Chapter 6 will go through some of the known issues and future work with the extension. Chapter 7 contains the conclusion. Code snippets can be found in appendix A and the evaluation form in appendix B.

2 Background

This chapter will present information that is relevant to the work done in this thesis.

2.1 Customization and Personalization

Customization and personalization are two much related terms. The definitions for them can change depending on the context they are mentioned in.

2.1.1 Customization

Customization in the context of computer applications is when the application contains functions that allows a user to change the appearance of the application. This includes colors, positioning and whether an element, for instance a button or an image, is present (showed) in the application. One of the main points of customization is to let the users take control over parts of the application [9]. This allows them to better familiarize themselves with the look and feel of it. Something as simple as changing the outline color of the web browser can make a user feel like ”this is mine” or ”I made this what it is”. This helps create loyalty to the application and make the user prefer this application over similar applications in the future.

Evidence suggests that users spend more and more time customizing the interface of their application and they do so with a great deal of involvement [9]. To do so would require a reasonable amount of time, but the main reasons for a user not wanting to customize an application is either because of lack of time or that the application was too hard to modify [8].

2.1.2 Personalization

Personalization is defined as automatic adjustment, re-structuring, and the presentation of tailored information content for individuals [3]. An application cannot personalize itself to a user without any knowledge about the user's preferences or behavior. So many of the problems concerning personalization involves how, when and what data to collect about a user.

- User profiling. Gathering information about the user to create a user profile. This profile can contain behavioral patterns, demographic information, and the user's interests. This method is commonly used in online stores, where this information is used to predict what the user might buy or be interested in next.
- Log analysis. By analyzing the log and calculating usage patterns, the application can find the user's preferences. Analyzing the log is usually a part of creating a user profile.
- Web publishing. This involves the application or website having a publishing mechanism to present some content to the user. The content presented can be data that was stored locally or received from a third party.

For the rest of this thesis, personalization refers to when the application automatically make changes to itself based on available information. When customization is mentioned it refers to a change that is initiated by the user [2].

2.2 Adaptive User Interfaces

Adaptive user interfaces is a term that is very relevant to the future work of thesis

A user interface (UI) is a key component to software applications since it connects the users to the functionality of the application. A "One design fits all" approach can in many situations lead to a diminished user experience because such a UI is unable to accommodate for all the context variables introduced through different use and users [1]. A well-designed UI will reduce the need of training users to be able to operate the application efficiently. Over time the users will get more experienced and to take advantage of this the concept of an adaptive user interface (AUI) is introduced.

"An AUI is formally defined as, a software artifact which can automatically alter aspects of its functionality and/or interface and improves its ability to interact with a user by construction a user model based on partial experience with that user [7]*."*

2.3 Human-computer interaction

Human-computer interaction is a term closely related to user interfaces.

Oxford reference library defines Human-computer interaction (HCI) as:

"The study of people using computers: a mixture of engineering, design, and behavioural science. HCI can be separated into four dimensions: the task dimension, dealing with goals and purposes of the engineers, designers, and the users of computers; the dialogue dimension, concerning how the computer and user are intended to interact; the structural dimension, dealing with specifics of the layout and the grouping of tasks; and the usability dimension, dealing with the ways in which users and computer actually interact. The human-computer interface is both the hardware and software of the computer and includes elements such as the keyboard, mouse, screen, graphical user interface, windows, drop-down menus, and other means of accessing information[3]*."*

2.4 Web browsers

This section describes customization and personalization options available in today's web browsers. Most of the options mentioned in this section is available for all the other major web browsers(Firefox, Safari, Internet Explorer, Opera), but this thesis focuses on the chrome web browser.

In web browsers, customization is facilitated by allowing the user to install themes that changes the color scheme and appearance of the browser[4], and installing extensions can modify and enhance the browsers functionality[5]. All of these options can be viewed and edited in the browser's own

[3]`http://www.oxfordreference.com/view/10.1093/acref/9780199568758.001.0001/acref-9780199568758-e-1225?rskey=sEoS2j&result=1222`

[4]`https://chrome.google.com/webstore/category/themes`

[5]`https://chrome.google.com/webstore/category/extensions`

settings page.

The chrome browser also allows the user to create their own profile. This is achieved by creating a new, or by using an existing, Google account to sign in to the browser. Doing this will allow for the user to bring all the settings, extensions, bookmarks, apps and themes to other devices using the chrome browser. It also enables a few personalization options such as omnibox(autocomplete suggestions for the address bar), saving usernames and passwords for different sites and to automatically fill out forms[6].

2.5 Chrome extensions

Chrome extensions are small software programs that are used to modify and enhance the functionality of the chrome browser. They are written using the common web languages consisting of HTML, JavaScript and CSS[7]. Most chrome extensions have little to no UIs as many of them works in the background and therefore have no need for user interaction to do their work. The most common UI for an extension comes in the form of an icon on the top right of the browser, either within the address bar (called page action) or in the toolbar (called browser action). The page action icon is shown only on pages chosen by the extension, and it usually means that the page has been injected with a context script. The browser action icon is shown regardless of what page is open in the browser, and is used when the application is relevant to most pages.

A chrome extension consists of the following files [5].

- **A manifest file:** Every chrome extension have a file called "manifest.json". The file contains information about the extension such as the name and version of it, what icons and images it uses, the permissions for it and the scripts and their roles in the extension. As the file extension suggest, the contents of the manifest file are written in JSON format.

[6]`https://www.google.com/chrome/browser/signin.html`
[7]`http://developer.chrome.com/extensions`

- **HTML files:** Unless the extension is a theme for the browser, it will consist of one or more HTML files. The main file is labeled under background page in the manifest file. It will stay invisible in the background and respond if the page action or browser action is pressed. There can also be other HTML files in the extension, they can be used to open a pop-up when the browser action is pressed. The file will usually show as a pop-up window, or the files can be used as a settings page for the application.

- **JavaScript files:** These are optional to have in an extension since the HTML files can contain scripts. If the extension shall interact with web pages it will need one or more JavaScript files, mentioned under content script in the manifest file, together with the URLs to the pages it is to be injected into.

- **Other files:** These are other files the extension may use in some form or another, most commonly image files to use as icons for the page or browser action.

2.6 Technologies

This subsection will go through the different technologies used when developing the extension created in this thesis.

2.6.1 XPath

XPath is used to navigate through elements and attributes in an XML type document. It is quite old, with its first appearance in 1999 and it became a W3C recommendation the sixteenth of November the same year[8].

XPath models a XML document as a tree of nodes. There are different nodes, such as element nodes, attribute nodes and text nodes. XPath defines a way of calculating a string value for the relative location of a node. In the application a XPath string for the users profile icon on YouTube

[8]`http://www.faa.gov/nextgen/programs/swim/documentation/media/compliancy/Xpathv1.0.pdf`

looks like this: /html/body/div[2]/div/div[2]/div/div[2]/span[2]/button - /span/span/span/span/img

2.6.2 Regex

Regex is a popular data cleaning method, and is based on the use of regular expressions. It can be used to filter out irregular characters or remove all uppercase letters. It can also detect more complex patterns in text. In the application developed in this thesis, regex is used to locate values in CSS text.[9]

2.6.3 jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.[10]

2.6.4 JSON

JSON stands for JavaScript Object Notation. It is a format used to create a string representation of the common data structures, such as numbers, lists and key value pairs. It is based on a subset of the JavaScript programming language, hence the name[11]. JSON is language independent and code for reading and writing it can be created in any programming language. While JavaScript contains native support for JSON, other languages can use an external JSON parser library or the developers can write their own parser for it.

[9] `http://en.wikipedia.org/wiki/Regular_expression`
[10] `http://jquery.com`
[11] `http://www.json.org/`[12]

2.6.5 CSS

Cascading Style Sheets (CSS) is a language designed for describing the appearance of documents written in a markup language such as HTML. It is used to separate styling from functionality and it can be used by more than one page. This way, the style of several pages can be edited without having to change each page individually [11].

2.7 Related work

To my knowledge there has been no other work done that uses this many different types of functionality. However there has been work done that bears resemblance to some part of the functionality implemented to the extension developed in this thesis.

An experiment has been done in 2001 studying the effectiveness of visually enhanced history browser mechanisms on web navigation [10]. There were three different history mechanisms used in the study. First was the browser history in the Netscape browser, which is very similar to the browser history page in today's browsers. The other two are called GlobalTree and DomainTree. They are very similar to the recent log and the domain log developed in this thesis. Although the DomainTree bears similarity to a suggestion from one of the test users in this thesis to improve the recent log and the domain log. This suggestion is discussed in depth in section 5.16.

3 System design

This chapter will go through the design of the extension developed in this thesis.

3.1 Software design

3.1.1 Sidebar

The application is a *chrome extension*, this is because extensions are meant to be small software programs that can enhance the functionality of the browser. This fits with the goal of this thesis. The main user interface consists of a sidebar that can be toggled on or off. The user will do this by clicking a button inside the address bar to the right. The location of the sidebar is located along the left side of the browser. In future versions of the extension the location of the sidebar can be changed by the users using the settings page. What options the users will have regarding moving the location of the sidebar around is not set in stone and different options will be evaluated when that time comes. The sidebar, when toggled on, will lie on top of the web page,because pushing it away or rescaling it might ruin the design of the web page and disrupt some of the existing functionality. This is especially true for sites that change their layout based on the window width, so pushing away other elements on a page and taking their place may make the page think it has more space to work with than it actually has.

3.1.2 Custom buttons

For the functionality of the extension it has the ability to *add a custom button* for all the elements on the web page. These will be stored by the application and remembered for each web page. This will allow a user to create bookmarks for a domain. An example of use for this functionality is on pages like Facebook[13] where it can be used to create a list of people and pages you as a user would visit often when visiting that domain. Since

[13]www.facebook.com

some buttons are url references it will be easy just to store the anchor target of the button and it will either be remembered per domain and work like bookmarks for each web page. If there is no href target to the button the user wants to add to the sidebar, the application will calculate the the xpath of the button and store that.

3.1.3 Logs

The application will also function as a log for the user, one for the domain that will store the few last visited pages within a domain such as all pages under vg.no or facebook.com. The number of pages stored in the domain log is by default three, but it is changeable by the user. And another one that will remember the last ten pages visited regardless of the domain. As with the domain log, the number of pages saved is changeable by the user. The idea is to add a new web page to the log using some of the existing function such as page.onload. Some problems have been detected on some domains such as YouTube[14]. Sometimes does not call the page.onload function between pages within the domain.

The domain log and the recent log are programmed so they have the same functionality as each other, but the domain log is stored differently per domain and the recent log is not. When the content script notices that the browser has opened a new page, it will send that page to the background script. The background script will check for duplicates and if it finds one it will delete the duplicate. If it does not detect any duplicates it will find the oldest entry in the log, delete it, and then add the new one. This functionality is the same regardless whether it is the recent log or the domain log the background script is dealing with.

3.1.4 Pre-set buttons

The application also tries to locate some existing buttons such, as a logout button. Login however, might require the application to remember the login credentials of the user. Chrome already has a way of storing usernames and

[14] `www.youtube.com`

passwords for different websites that works quite well so if the login button is to be done it would overlap with the login function in chrome. They probably cannot work together since usernames and passwords stored by chrome are not accessible by extensions. Because of this, the logout button is the only one that has been implemented in the extension. There are other functions that could have been included in the extension, such as next/previous article function.The usability of it could have been great, but it would be difficult to create this function since there is no common way of finding the next article. Therefore the idea of a next/previous button was discarded in favor of the logout button.

3.1.5 Notes

The application is able to save individual notes for each domain. These are stored in the chrome extension localstorage. The idea is to allow the user to save reminder notes on a domain, in a somewhat similar fashion as post-it notes are used.

3.1.6 Images

On forums on the internet it is popular to use pictures as responses, however keeping track of all the pictures can become tiresome unless they are stored and sorted in a proper manner. The application will allow the user to easily save pictures and gif animations to an easily accessible location in the sidebar. Here the images could be added to the clipboard by the user and pasted to where the user wishes.

For future work a solution would be found that allows the images to be sorted and searched through using tags or another potential solution if one is found.

3.1.7 Settings

Since some of the focus of this application is customization, the user should be able to change the appearance of the application according to their wishes. The goal was to explore as many options as possible, or at least figure out what keeps an option from being possible to implement. The first ones to be implemented was the easiest options, such as changing the color of the background of the sidebar, change the size of text, move the buttons around, change the shape of the buttons, change the font of the text for one or all the buttons. The more complex options are left for future work, and include options such as changing the position and the size of the sidebar and making the edges of the sidebar change to a more complex shape. This would require the user to provide an image with transparent areas to be used as the background of the sidebar.

3.2 Storage

In this chapter I am going to go through the 4 different methods of storing data as a chrome extension and the pros and cons of using them in my application. Then it will be explained what I choose and why.

- The first method of storing is the user's *hard drive.* Apart from it being hard to implement, it restricts the user to one computer since it will not synchronize across computers. However, storage space should not be an issue. There also seems to be an issue since chrome only allows extensions to save files to the users chosen download folder. In theory it should be possible to create a subfolder in the downloads folder so that the files will be stored in a somewhat orderly fashion. To use this, a chrome extension will need the downloads permission in the manifest file. This allows an extension to access the user selected download folder.[15]

- The second method is using *chrome extension localstorage.* This has two available ways of storing, one is saving online on the users chrome account. This allows for a maximum of 512 entries and 100 kb of data. On older version of chrome it only allows for 1 update every 2 seconds, so it is not usable for something that needs a lot of rapid updates. Since chrome version 40, the limit has been increased to

[15] `http://developer.chrome.com/extensions/downloads`

2 write operations per second and exceeding this limit will cause a runtime error. The other part of localstorage stores on the user's hard drive and is by default set to a maximum of 5 mb of storage, but the extension can be given the unlimited storage permission which allows for virtually unlimited storage[16]. The localstorage stores items in a key-value store where both the key and the value need to be represented as a string type, so the JSON format can be used. The functions for adding items to the chrome storage are asynchronous for both local and online storage.

- The third method is using the *HTML5 localstorage.* This is, like the chrome extension localstorage, by default set to a 5 mb limit per domain when using the chrome browser. The methods of accessing this storage are synchronous and therefore easier to work with. Storing url values in this could be favorable since 5 mb of storage would allow at least 5 200 000 characters to be stored, assuming an average url length of 70 characters would mean that the localstorage can store over 74 000 entries. The number would be less if the application is to store additional information in each entry, but should still suffice for a lot of users.

- The forth method is using the space available on *Google drive.* Google drive gives away 15 Gb to each user for free, 30 Gb if the account is through school or work[17], this should allow for enough storage for everything the application needs. The drawback with this is that the application will need to get access to the users Google account.

3.2.1 Evaluation storage options

Storing on the user's *hard drive* has the advantage that it allows for virtually unlimited amount of storage and any format can be used. However it is restricted to use only the downloads folder of the user's choosing, and the

[16]`http://developer.chrome.com/extensions/storage`
[17]`http://support.google.com/drive/answer/6558?hl=en`

user can then edit the content in the storage intentionally or unintentionally which can have unknown reactions when the extension is trying to read it.

If the extension were to use the user's available space on *Google drive*, the extension would be able to use the same storage regardless of when computer is used, as long as it is connected to the internet. The size of the storage would not be a problem despite the fact that is rather small compared to what could be available on the user's hard drive. The con of using Google drive as storage is that the user would have to have a Google account and allow the extension to have access to it.

The HTML5 localstorage is fast and easy to use, the downside is that it is too small to store anything else than text values. This will be a problem when it comes to storing images in the extension.

The chrome extension localstorage will potentially have the same storage size as saving on the user's hard drive, but it is only accessible for the user through the extension. Compared to the HTML5 localstorage, it should allow for storing any type of data and not only text. *"User data can be stored as objects (the localStorage API stores data in strings)"*[18] The con is the big throughput limit on online storage, and that it is asynchronous.

Because of similarities in usage between chrome extension localstorage and the HTML5 localstorage, the chrome extension localstorage has not been used in the extension. This error was only noticed in late development and it was then decided that it would cause any issues as long as the extension would not be used over a large period of time so that the 5 mb limit would be exceeded. In a future version of the extension all will be stored using the local part of chrome extension localstorage. The only exception to this would be the settings data, it will be evaluated if it is small enough to be stored using chrome sync instead.

3.3 Hardware adaptation

This section will discuss the design decisions related to making the extension adapt to different hardware devices.

[18]`developer.chrome.com/extensions/storage`

- Chrome for android devices

  Chrome for android in its current state does not support either apps or extensions. Moreover, according to the chrome developer faq[19] there are no plans in the near future to add support for either apps or extensions in chrome for andriod.

- Different screen resolution

  Not all computers have the same screen resolution[20]. Because of this the size of the sidebar should not be declared in amount of pixels, but rather by a percentage of size of the browser window. This will allow the sidebar to adapt to different screen resolutions.

  For a future version of the extension, the user will be able to edit the size of the sidebar.

[19] `https://developer.chrome.com/multidevice/faq`
[20] `http://www.w3schools.com/browsers/browsers_display.asp`

4 Implementation

This chapter covers the implementation and the architecture of the extension developed for this thesis.

4.1 Architecture

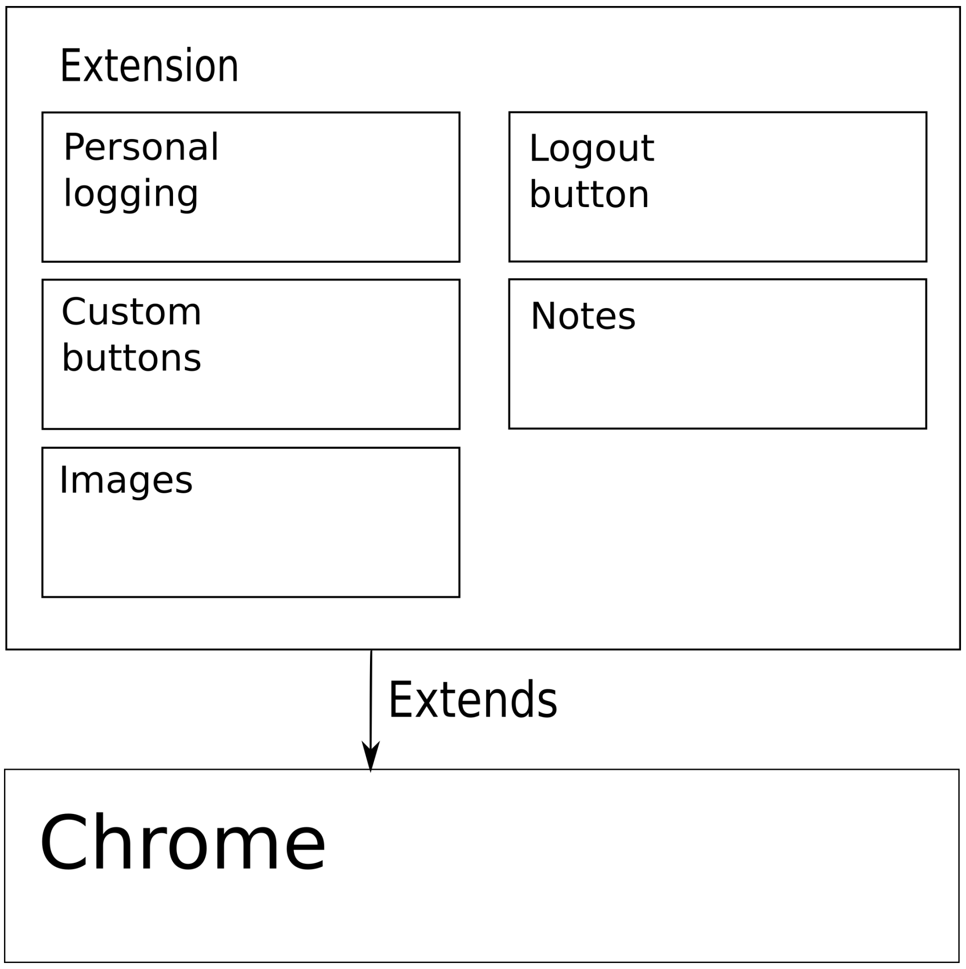


Figure 1: Chrome extension

Figure 1 shows the extension as a extended part of the chrome browser containing additional functionality.

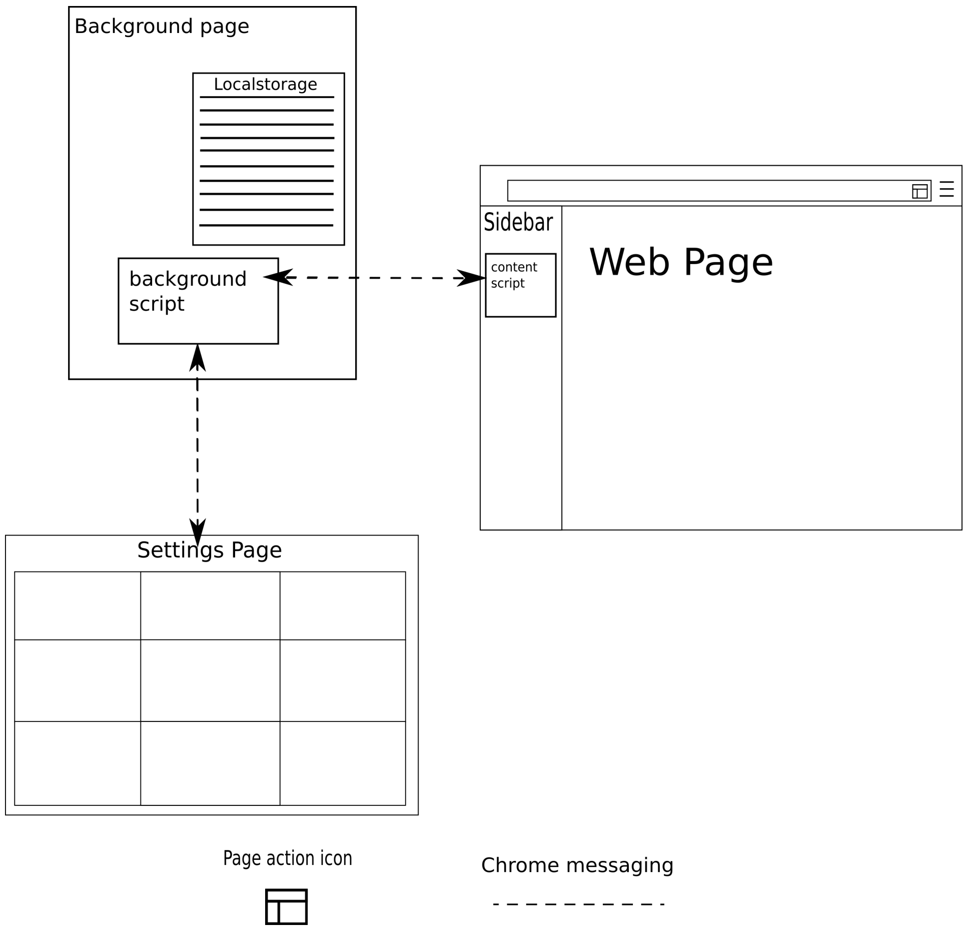


Figure 2: Architecture from a developers point of view

Figure 2 shows the architecture of the extension from a developers point of view. Here you can see where the different parts of the extension are located and what parts of the extension communicates with each other through chrome messages.

4.2 Extension permissions

The extension developed in this thesis has cloned the permissions from the adblock extension[21]. These permissions made it able to read and edit the content of all visited web pages. This gives the content script full access the document so that it can easily add new buttons, highlight areas and delete elements from the visited page.

Only using the permissions from the adblock extension did not allow the extension to store data. Because of this two new permissions was added to the extension to allow experimentation with different storage options . They are the "storage" and "unlimitedStorage" permissions.

The approach taken when adding permissions to the manifest file of the extension has been to allow it to do more than it actually does. This is because it allows for a smoother development with fewer issues related to lack of permissions. From a developer's point of view, too many permissions is not an issue, but from a user's point of view it can be. This is because the user has little control over the extension when it is allowed access to everything at all times. Because of this a future version of the extension will remove all unused permissions and the less used permissions will be moved to optional permissions. This will allow for the user to be more in control and allow for the extension to explain why it needs the optional permissions when it prompts the user for them[22].

4.3 Logout button

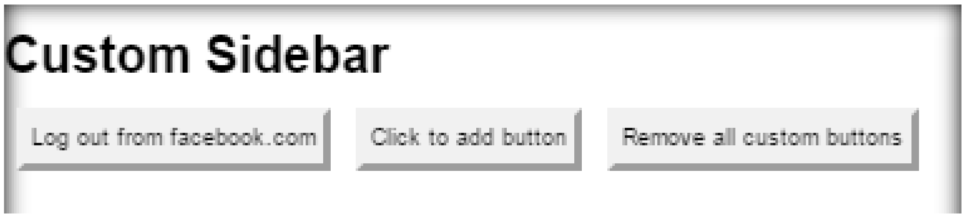


Figure 3: The logout button

[21]http://chrome.google.com/webstore/detail/adblock/gighmmpiobklfepjocnamgkkbiglidom?hl=en
[22]https://developer.chrome.com/extensions/permissions

The first function implemented into the sidebar was a button that would serve a similar purpose on as many sites as possible. It would not be adding new functionality to the site but instead move existing one to a universal location on the sidebar. There was a few functions that were considered to be a pre-set button for the extension, but only one would be implemented as a proof of concept.

The *logout button* was the first function to be implemented into the sidebar. The reason for this are that it does not require the user to supply any information to the application beforehand. In addition, some sites hide the logout button from plain view so the user can experience difficulties finding it. The logout button therefor seemed benefit the user if it was easily accessible for users in a universal location for each website.

The way the logout button works is by parsing through the html document of a site, searching through all its elements for something that hints towards the element being the logout button. This is done by searching through an elements innerHTML, id and value for names matching a common way of spelling logout such as "log out", "logout", "log off", "logoff", "sign out" or "sign off". After finding this element the application simulates mouseevent 1, which is a left mouse button click, on the element. Testing this method has proven not to interfere or have any unexpected results compared to if the application tried to initiate the logout by attempting to call the event handler related to the button. Even though it has not been thoroughly tested, there has not been reported any false positives. There have however been a few pages where the function did not find the logout button when it should have. Usually when it is unable to find the logout button it would be hidden in a drop-down menu so from the extensions point of view the actual logout button does not exist until the drop-down menu has been opened.

There were two other functions that were considered alongside the logout button but were never implemented. One of these functions was *show next/next element*. Pressing this button would open the next related article or next related post on that website. This idea was discarded because there was no common way of accessing the next post on a website. This would force the function to be made different for a lot of websites.

The other function that was considered to be implemented was a *login button*. There is a lot of websites that allows users to log in, but to do so require a username and a password. It would not be a hard task to store this information, but having the user supply this information to the application for each website when chrome itself already has this functionality[23] seemed to be redundant.

In a future version of the extension, the logout button would search through the web page when the sidebar is opened. Moreover, if it cannot find a logout button on the site, it will hide itself.

4.4 Add custom buttons

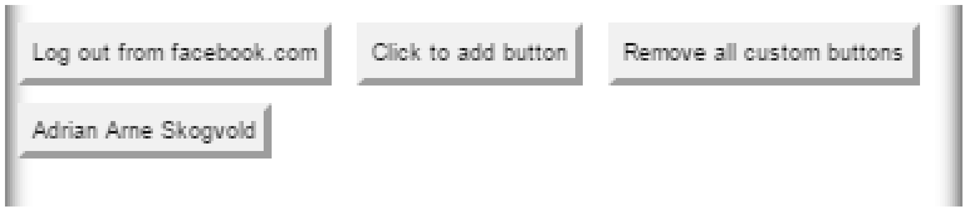


Figure 4: One added custom button

The *add custom buttons* function allows the user to take control and add as many buttons to the sidebar as he/she pleases. From the implementation's point of view it has two sides, one for adding links/anchor elements, and a second for adding other types of elements to use as buttons. The reason for allowing the users to create buttons out of all elements is because they might have been given a click event by a script after page load. The application can struggle to detect whether the element in question is an actual button, image or text element with a click event attached to it. Therefore the application allows the user to decide what can and cannot be turned into a button on the sidebar. This strengthens the user's feel of control.

When the user clicks the *"add custom button"* button, the application

[23]`https://support.google.com/chrome/answer/95606`

creates a highlighter, which is a light blue transparent box with a less transparent and darker blue outline. The highlighter listens to mouse movements and changes its size depending on the size of the element the mouse is hovering over.

This highlighter solves two important issues when creating buttons. The first one being that it is a clear visual representation to the user of what element on the page is being turned into a custom button. The other issue that is solved is when the user clicks on an element to add it as a button the highlighter denies the click event associated with the element to fire, this will prevent the user from opening a link or submitting a form instead of adding it as a button. The alternative to the highlighter would be attaching a click event to every element on the page that would call back to the content script when the element has been clicked. This method can potentially interfere with the sites functionality, so adding the click event to the highlighter removes the need to add a click event to all the elements on the page.

The highlighter keeps a reference to the element it is covering, so that when it is clicked it can give the application a pointer to that element. The application will then check if the element is a link or not. If it is a link then it will add it to an internal list of custom links and send it to the background page for it to be saved in the localstorage. After saving it will create a button for the link and add it to the custom button container. If the element is not a link the application will, using xpath, create a path to its location on the page and send it to the background page for storage. The application will then as it would have done if the element were a link, create a button for it and add it to the custom button container.

To allow the user to delete the added custom buttons a function was created that deletes all the custom buttons that has been added within that domain.

4.5 Log

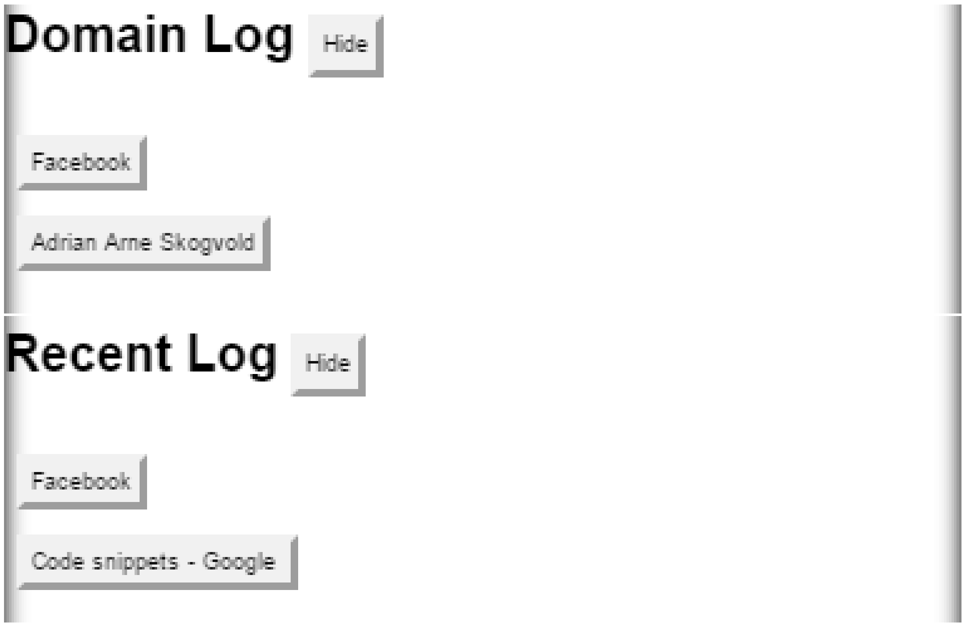


Figure 5: The domain log and the recent log

Another function that has been added to the sidebar is the ability to save the users most recent history in an easily accessible way without being a copy of the browser history currently available in chrome. This function was split into two parts. One logs the users history within a single domain (called the domain log) and the other logs all the websites visited (called the recent log). The maximum length is by default set to three for the domain log and ten for the recent log. On the settings page the user can change the maximum number of items in both the domain log and the recent log to a number between zero and twenty.

A problems associated with logging the users history are when to save a log entry. The most reasonable way to achieve this seems to be when the page is loaded instead of saving a log entry when the sidebar is opened. This is done in the extension by having a function that will fire when a page is loaded. There are two ways of doing that, one is using the DOM-

ContentLoaded event and the other is using the javascript load event. In the context of the extension either of these would be fine since they should both fire when a page has been loaded. The difference is that DOMContentLoaded will fire when HTML document has been loaded and that the load event will wait until everything is fully loaded in the page[24].

A known issue is that on some pages, especially ones that are for streaming video content, such as youtube.com or twitch.tv, seem to not trigger either of these functions. Because of this issue the two different methods mentioned has been regularly swapped with each other to find out if one is strictly better than the other when used in the extension. In the extension that has been given to the test users the load event was used, but this might change for a future version.

```
// Adding a new element to a log
recieve message(element) from content script
if log contains element:
  remove(element) from log
add(element) to start of log
if length(log) > max log length:
  remove(last) from log
```

Figure 6: Psuedocode for adding new entries to a log

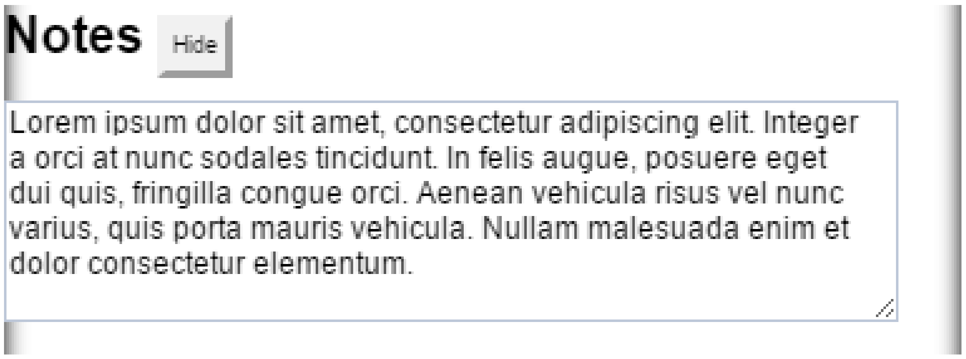
The two logs in the application are not that different from each other from a programming point of view. When a page is done loading the context script reads the url on the page and the domain. Then it sends both the domain and the full url to the background page to be stored. The background script will then go through both the previously stored logs and remove entries that have the same url as the page it is about to store, then it adds it to the list. The recent log will be stored under the key "recentlog" and the domain log will be stored using the domain provided by the content script as a key. It will then count the number of elements in each log and if the log is longer than the user specified maxlength of each log if that has been set, if not it will use the default maxlength. If the background script finds

[24] `https://developer.mozilla.org/en-US/docs/Web/Events/DOMContentLoaded`

the log to be too long it will remove the oldest element in the log and count again.

The log is loaded when the sidebar is opened for the first time on a new page. To do this the content script sends 2 requests to the background page, one requesting the recent log, and one requesting the domain log for the current domain. The requests are asymmetric so logs will be filled once the elements are received, which might be after the sidebar is opened if the background page is slow to respond. Delays could happen, but will never be so slow that it becomes an issue.

4.6 Notes container



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer a orci at nunc sodales tincidunt. In felis augue, posuere eget dui quis, fringilla congue orci. Aenean vehicula risus vel nunc varius, quis porta mauris vehicula. Nullam malesuada enim et dolor consectetur elementum.

Figure 7: The notes container filled with a dummy text

The point of the notes container is to allow the user to save a piece of text in a simple, accessible manner. It is implemented as a text-area located inside a container element. Attached to the text-area is an onFocusLoss event handler. This handler will fire each time the text-area loses focus, such as when the user clicks outside of the text-area after typing. The handler will then send the text inside the text-area and the domain of the site the browser is currently in to the background script. The extension will then store the string in the localstorage using txt + domain as the key. The max length of the text saved in the notes field that has been tested to work, is a text consisting of 2000 words and 13472 characters. The max length of the string saved would theoretically be decided by the maximum available memory on the device used.

4.7 Image container

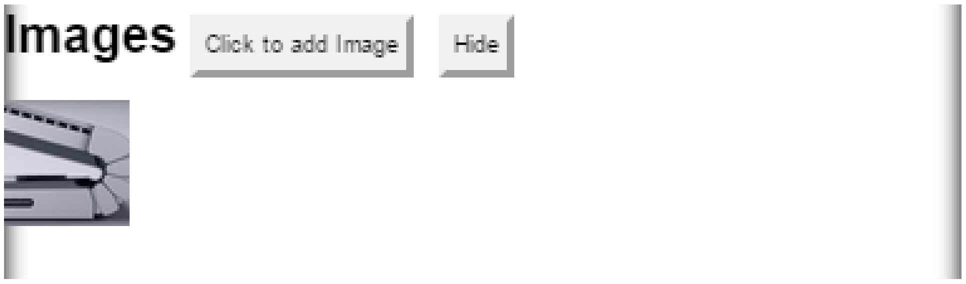


Figure 8: The image container

The image container is there to allow the user to easily save an image without having to think about where to save it to or what name to give it so that it can be found later. This function holds many similarities with the add custom buttons function. When the user clicks the add image button a highlighter, such as the one in the add custom button function, will highlight the element under the mouse pointer with a blue transparent box if the element is of type ”IMG”. If the mouse is hovering over an image, but the highlighter is not shown, it means that the image visible is attached to a link or there is an invisible element on top of the image. When the highlighters box has been clicked, the image url would be put in the added images list in the context script. Then it would be sent to the background script to be stored in the localstorage. The context script would then call the LoadImages function. This fetches the stored images, resizes them to the default small square of 50x50 pixels, and adds them to the image container.

To delete images from the image container the user need to hold down the alt key and click on the image. The application will then iterate through the images in the added images list and delete the image with the same url as the image the user clicked from the list. Then it notifies the background page of the change by sending it the updated list to be added to storage. Then the application will empty the image container and call the LoadImages function to fill it again with the updated list of images.

The way images are deleted is easy and clean. Using it to delete the created buttons in the *add custom buttons* function seemed natural instead of having two different ways of deleting added elements in the sidebar. Due to issues with the custom buttons being two different types of elements, it was not working properly in time for the extension to be distributed to the test users. To compromise the remove all added custom buttons was created.

4.8 Hide buttons

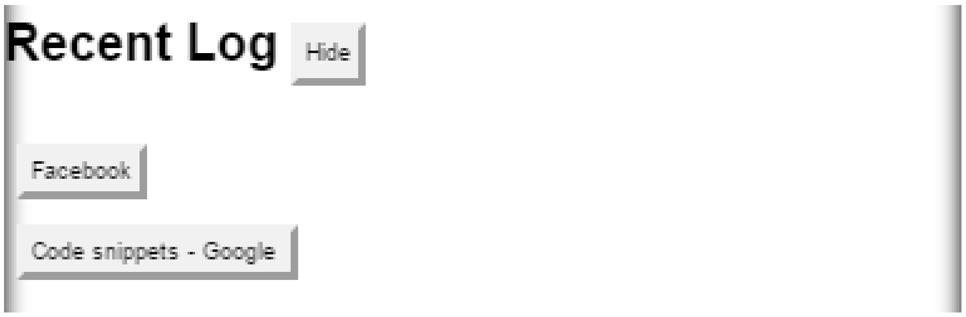


Figure 9: Recent log is shown

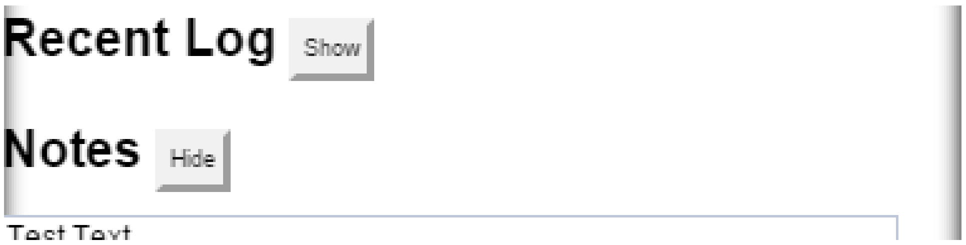


Figure 10: Recent log is hidden

A user might not want all the info in the sidebar shown at all times, for this the *hide buttons* was created. Their function is to hide a part of the sidebar if the user wishes to. In the extension they are located to the right of a header and when clicked, they will hide the container under them. For example, when the hide button located next to the recent log header is clicked, the recent log will appear empty and the text inside the hide button will now read "show". To now show the elements in the recent log the hide button will need to be pressed again.

The extension does not remember if the user has hid parts of the sidebar when opening a new page. This is something that will be added in a future version.

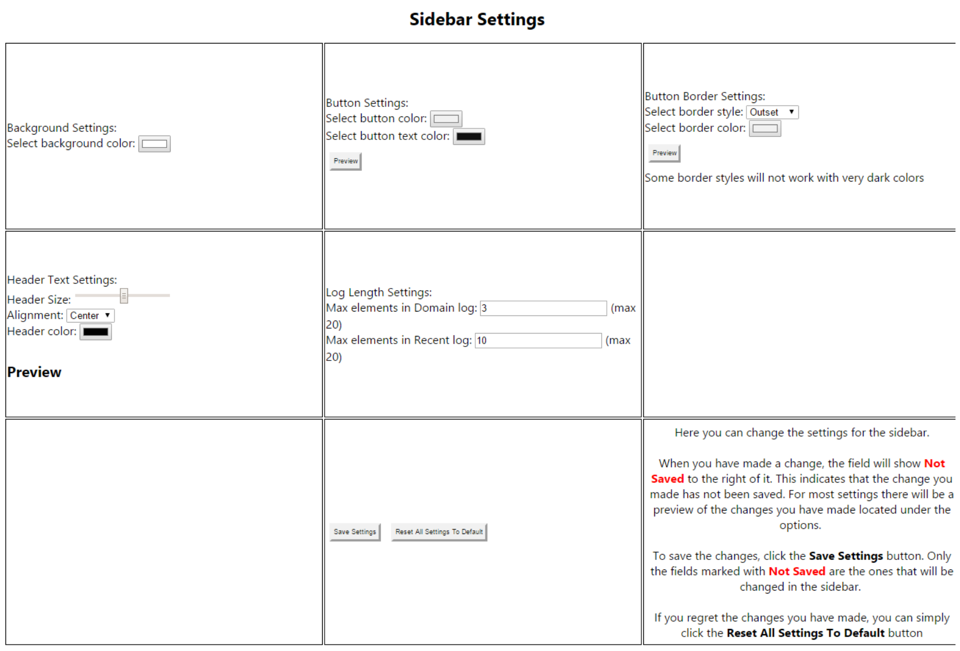
4.9 Background script

At all times, when a browser window is open, the *background page* is running invisibly in the background. It only contains two lines, one for running jQuery and one for the background script. The main function of this is to enable the content script to store values across different pages in a localstorage that is only used by the extension. The background script first order of business is to listen for click events on the *page action icon* of the extension, and then notify the content script when it has been clicked (see appendix A). Listening to messages from content scripts, and responding to them with either a "Stored" message or the value located in the localstorage under a key provided by the content script, are the most important functions of the background script. When receiving requests to add entries to the different logs, it also makes sure to delete duplicates and that the logs does not exceed the user specified maximum length.

Due to some conflicting sources during the development of the extension, it was believed that the unlimited storage permission applied for the HTML5 localstorage. This turned out to be false[25], so for a future version of the extension the chrome localstorage would be used instead.

[25]http://developer.chrome.com/extensions/declare_permissions

4.10 Settings page



Sidebar Settings

Background Settings:
Select background color:

Button Settings:
Select button color:
Select button text color:

Button Border Settings:
Select border style: Outset
Select border color:

Some border styles will not work with very dark colors

Header Text Settings:
Header Size:
Alignment: Center
Header color:

Preview

Log Length Settings:
Max elements in Domain log: 3 (max 20)
Max elements in Recent log: 10 (max 20)

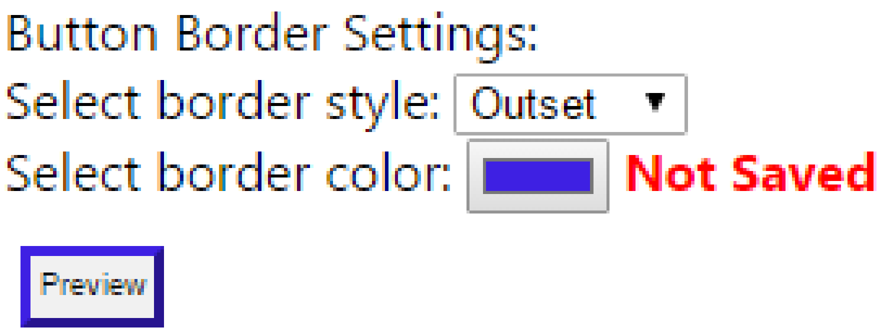
Here you can change the settings for the sidebar.

When you have made a change, the field will show **Not Saved** to the right of it. This indicates that the change you made has not been saved. For most settings there will be a preview of the changes you have made located under the options.

To save the changes, click the **Save Settings** button. Only the fields marked with **Not Saved** are the ones that will be changed in the sidebar.

If you regret the changes you have made, you can simply click the **Reset All Settings To Default** button

Figure 11: The settings page with the default values loaded



Button Border Settings:
Select border style: Outset
Select border color: **Not Saved**

Figure 12: Unsaved option in the settings page

The design of the settings page was originally intended to be scalable with the number of settings options added to it, but for simplicity and because the looks of the settings page was not a priority during the development of the application, it was created as a 3x3 grid with white squares and a black outline that scaled in size with the height and width of the page. In this page it was easy to add settings options as they were made possible by the application. When the application was to be sent out for testing, the few empty fields were used to contain a small user manual to help the test users.

Options for changing the visuals of the application were the first to be added. To make this possible the background page store the custom CSS of the application in localstorage, so that it is ready to be sent to the context script to be applied to the sidebar, or to the settings page to be edited. To allow a custom CSS to be added to the sidebar without changing the visuals of all the other elements on a page, the context script makes sure that all elements added to it has an id that starts with ext. The rules in the custom CSS can be made to only apply to elements with an id that starts with ext. For example, the rules for buttons start with $"button[id\wedge = "ext"]"$.

Once a change has been made to a field, bold red letters spelling "Not Saved" will appear to the right of the changed item (see figure 12). This is to indicate what settings have not been saved in the application. The text will then disappear when the user clicks the save settings button. To add the changes to the CSS every settings field has a selector and an attribute value attached to them. In addition they have a third value, called replacement, which that is fetched from the field the user can change. For example, if the user changes the *button text color* field to white the replacement value would be set to "FFFFFF". These three values would be sent to a function that, using regex, parses through the CSS and finds the rules to change. This is done by using the selector and the attribute variables to find the relevant value to replace the content of the replacement variable (see appendix A). After the save settings button is pressed, the CSS is sent to the background script to be saved in the localstorage. So the next time the context script ask for the CSS to use the user edited CSS, it would be sent in response to be used in the sidebar.

Other than visuals, the settings page allows for the user to specify the number of elements that are to be saved in each log. If the user wants to have only the last three visited sites shown in the recent log, all they need to do is to change the value for the field *"Max elements in Recent log"* and click save settings. This will send the value 3 to the background script to be stored under the key *"recLogLength"*, this will make the background script use this value to determine the max number of elements allowed in the recent log storage next time it is prompted to add a new one.

4.11 Storage methods used

This chapter will go through the method of storing used in the extension developed in this thesis and the reasoning and technical details behind the choices made.

- Using an online storage service such as Google drive[26] or dropbox[27] was discarded because it would require the user to have an account on one of those sites, and provide their username and password to the extension. Using the user's hard drive by storing text files and images in the user specified downloads folder was also discarded, because it would allow the user to have full access to all stored text and settings without going through the extension. For it to work it would have to check for errors every time a value was loaded. Chrome online sync storage was discarded because of the size restrictions, and that the user would have to log in to chrome with a google account. The choice fell on using the HTML5 localstorage in the extension but as mentioned section 3.2.1, the decision was made based on incorrect information that the unlimited storage permission would affect it. The only difference between chrome and HTML5 localstorage seemed to be that one was accessed asynchronously and the other synchronous. Synchronous was preferred, and HTML5 localstorage became the method chosen for storage in the extension.

[26]https://drive.google.com/
[27]www.dropbox.com

- When the content script is given the task of saving a button that is not an anchor type element, it cannot send a reference of the button it wants to save to the background page. The chrome message passing only supports sending string type messages and unlike a regular JavaScript object, it cannot be turned into a JSON object as it contains a circular reference[28]. Using XPath, an element's location on the page can be saved in string form as a path on the document tree structure. This path can be sent as a chrome message to the background page to be stored. When the path is fetched from storage the content script can locate the element, and simulate a mouse click to make the *onclick function* or the *click event handler* associated with it trigger. Another way of doing this would be to copy the onclick event on the element, but detecting these events has proven unreliable and no further attempts has been made to try and make it work.

- For anchor elements it is only necessary to save the href attribute of the element to create a button in the sidebar with the same function. This is because anchor elements only open a new page with the location stored in the element, so there is no need to make it unnecessarily complicated by making the page think the user clicked that link.

- For the three different buttons the sidebar saves, it has three different ways of deciding what the text on the button shall be. The thing they all have in common is that they only store at most 20 characters. This is because some titles can contain entire sentences and trying to fit all into a button would make it incredibly long or span multiple lines. When creating custom anchor elements the content script takes the innerHTML value of it, shortens it to a maximum of 20 characters and then puts it into a JSON object along with the href value before sending it to the background page. The same process will happen for the custom buttons. The difference being that when they are loaded into the sidebar, the content script will use the stored text only as

[28]A circular reference is an error that occurs when trying to stringify a object that contains a pointer that ends up pointing at itself.

a backup if it is unable to find any text in the target button. The idea behind this was that if the layout on the page changed and the stored path lead to a different element than the one it was intended for, the text on the button would imply that so the user can check if the target button had changed text or that the custom button were indeed pointing to the wrong element. When a new page is opened the content script will attempt to save it as an entry in both of the logs. To find the button text for these it takes the title of the page and stores it alongside the url of the entry. This was also intended to be the way the content script used to find the button text when creating custom anchor elements, but there seemed to be no way of getting to the title of a page without opening the link. Doing so would prompt the content script to add new log entries and cause the user to notice that a new page had been opened and then closed again. Finding the title of a link, to use as button text, is put on hold until an easier and less disruptive way of doing so is found.

4.12 Sequence diagrams

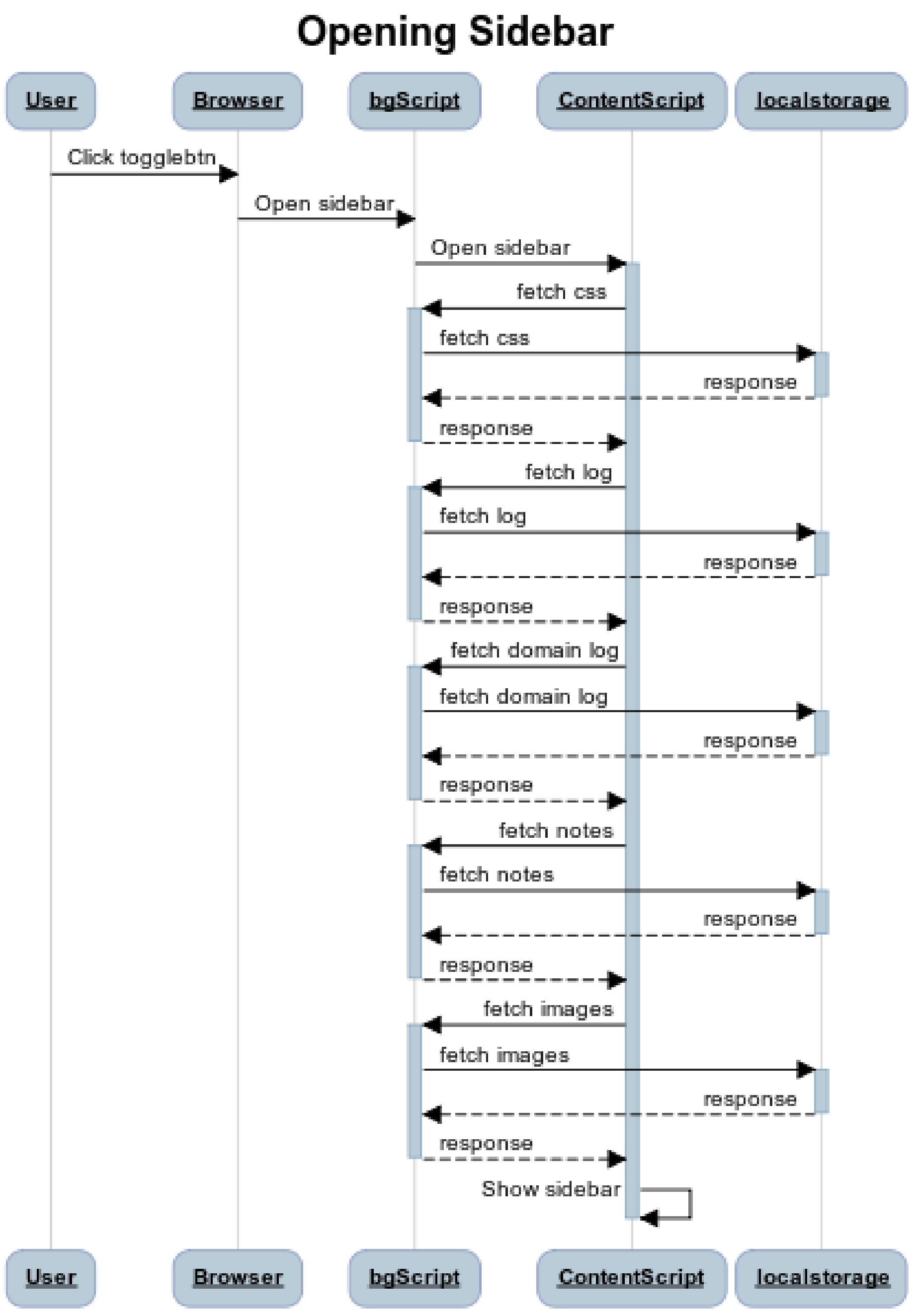


Figure 13: Sequence diagram for opening the sidebar

Figure 13 shows a sequence diagram of the communication occurring when opening the sidebar. The main point is that the user clicks the button to enable the sidebar. The event is then passed on to the context script that will start to fetch the stored links, images and notes. Since the context script does not directly have access to the applications localstorage, the communication needs to go through the background script. All the fetch requests are asynchronous, so the context script will send them all, show the sidebar and then add the responses to the sidebar when they are received.

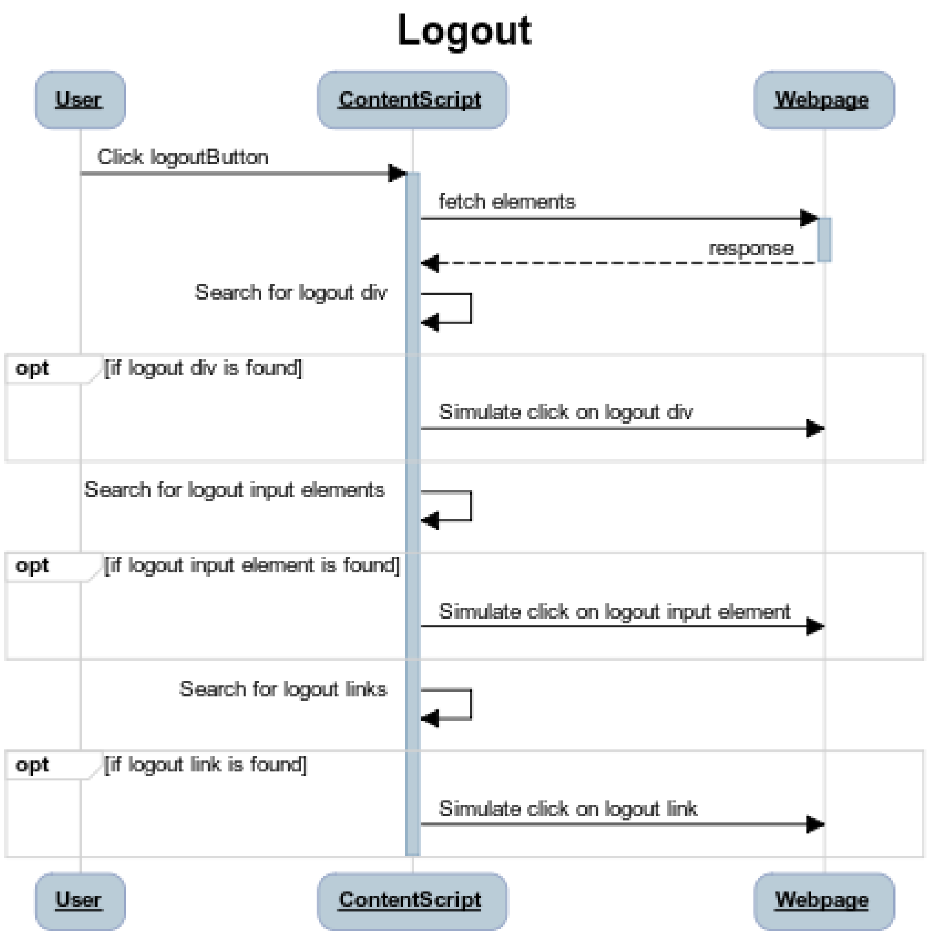

Logout
User
ContentScript
Webpage
Click logoutButton
fetch elements
response
Search for logout div
opt
[if logout div is found]
Simulate click on logout div
Search for logout input elements
opt
[if logout input element is found]
Simulate click on logout input element
Search for logout links
opt
[if logout link is found]
Simulate click on logout link
User
ContentScript
Webpage

Figure 14: Sequence diagram for the logout button

When the user clicks the logout button the content script will fetch all elements on the page, and search through them. First it will search through all the *div* elements to see if they have any element id, name or innerhtml matching known ways of spelling logout, either in English or Norwegian. If it finds a matching element a click event will be simulated on the element. If it does not find a matching element it will do the same matching with all input elements, if it does not find a matching element there either it ill do the same matching for anchor elements. If no matching element is found, the extension will do nothing.

4.13 The sidebar

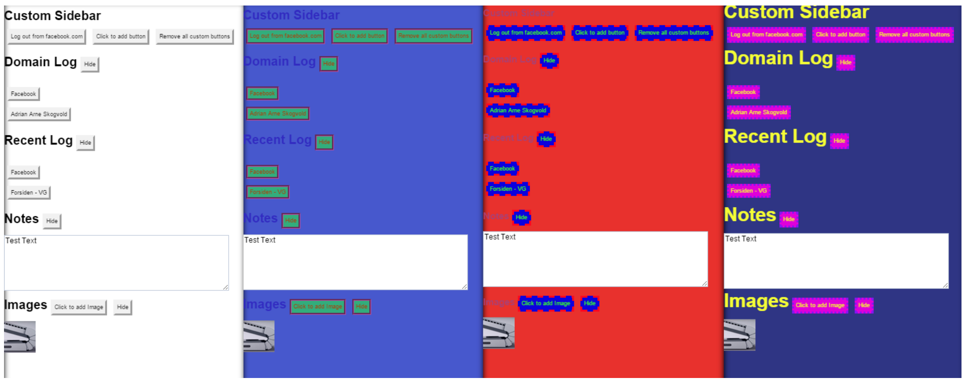


Figure 15: Final version of the sidebar

Figure 15 shows how the final version of the sidebar looks like in four different color options. The sidebar in the figure has no added custom buttons. If one were to be added it would lie under the three buttons at the top and right above the header spelling "Domain Log". It would look the same as all the other buttons a in the sidebar but would contain a different text. Both the domain log and the recent log contain two different entries each and the notes field contains the text "Test text". Lastly the image container has one image stored and displayed in the bottom of the sidebar.

5 Evaluation

An evaluation of the extension developed in this thesis has been completed by a group of nine selected test users. This section contains the information about the evaluation. A user manual was written and distributed to the test users to get them started with the testing of the extension. The manual can be found in section 5.1. The evaluation form distributed to and filled out by the test users after testing the extension can be found in appendix B. The evaluation form contains a few questions that required the test users to answer on a scale from one to five. The answers to these questions have been put in table 2. Info about the test users and other info about the evaluation can be found in section 5.2. Each section from 5.3 to 5.16 cover the answers given by the test users to the questions in the evaluation form. At the end of this chapter you find section 5.17, which contains the conclusion to the users evaluation.

5.1 The user manual

This is the user manual that was given to the test users.

- The sidebar:

  After you have installed the extension you will notice a small icon to the right in the address-bar. Clicking it will open the sidebar. The sidebar will not stay open when you open a new page.

- Logout button:

  To the top left in the sidebar you will find the logout button. Clicking it will, if possible, cause you to log out of the page you are visiting.

- Add custom button:

  The button with the text "Click to add button" is to add your own buttons to the sidebar. After it is clicked you will notice a blue square following your mouse pointer. Clicking this square will create a new button that when clicked will do the same as the element the blue square was covering, and will also have the same text as it. If what you clicked had no text inside it then the button will show up as an empty button. If it had a image inside the button will attempt to use that images as well, but this function is not yet perfected.

To cancel adding a button, just click on the "Click on a button or click here to cancel" button.

To delete the button you have added click the "Remove all custom buttons" button. This will remove all the custom buttons that are shown in the sidebar. It is not possible to only remove one custom button at this time.

The button you created will only show within the same domain you added it in, so the buttons you created on facebook.com will not be shown if you are visiting vg.no.

- Domain log:

  Under the title Domain Log you will find links to the pages you last visited the previous time you were in that domain. So for example, if you are on vg.no you might find links to the articles you visited the last time you were there. The default number of items show in the domain log is 3, this can be changed on the settings page.

  There are some pages that the application have trouble saving in the domain log, these are usually pages that have something to do with video and streaming. Some of those pages are: facebook.com, youtube.com and twitch.tv.

- Recent log:

  Under the title Recent Log you will find links to all the pages you have recently visited. Its function is to work as a smaller but more accessible version of your browser history. The default number of items shown in the recent log is 10, this can be changed on the settings page.

  As with the domain log, this also has problems saving some pages. These pages are the same ones as the domain log would have trouble with.

- Notes container:

  Under the title Notes you will find a textbox, here you can write some notes, like reminder notes. Clicking outside of the textbox will save the notes you have written. However if you close the browser or click the reload page button directly after writing something in textbox, what you wrote will not be saved.

  Notes are saved within a domain. That means that the notes you wrote on facebook.com would not show if you go on vg.no.

- Image container:

  On the bottom of the sidebar you will find a title called "Images". To the right of it there is a button with the text "Click to add Image". After clicking this you will notice a blue square when the mouse is hovering over an image. If you then click on it the image under it will be saved in the sidebar. If a blue square is not shown when hovering over a image then there is an invisible element covering the image, most likely a link. Clicking will then not save the image, but instead open the link attached to it.

  All the images you have saved will be shown on the bottom of the sidebar. To fit as many as possible in an orderly fashion, the images are all scaled to the same square size when shown in the sidebar. To show the full image just right-click and select "Open image in a new tab" or your language equivalent.

  Deleting a saved image is as simple as holding down the alt key on the keyboard and clicking on the image you want to delete.

  All the images you have saved will be available in the sidebar regardless of what page you were on when you added the image.

- The hide buttons:

  On the sidebar there are four buttons with the text "Hide". Clicking one of these buttons will hide the elements directly below it. So clicking the hide button next to the recent log title will hide the links in the recent log, clicking the hide button next to the notes title will hide the textbox below it, and so on. Clicking the button again will show the elements it hid.

- Settings page:

  The easiest way of getting to the settings page is to right-click on the icon to the right in the address-bar and then clicking on "Options", or your language equivalent, as shown in the figure:

  To the bottom right of the settings page you can find an explanation of how the it works.

5.2 General info

The test users that participated in this evaluation are identified as U1, U2, U3 and so on until U9. Direct quotes from the test users are written in this format: *"quote from test user."* The quotes from the test users are for the most part identical to the answer they gave in the evaluation form apart from fixing some spelling errors. Answers written in another language than english are translated in a way that attempts to retain its meaning.

	1. Age	2. Gender	3. Occupation or line of study
U1	23	Female	Student - Psychology
U2	16	Male	Student - Electro
U3	22	Male	Student - Economy and administration
U4	23	Male	Student - Accounting
U5	47	Female	Occupation - Teacher
U6	18	Male	Student - General studies
U7	25	Male	Student - Business
U8	22	Female	Student - Law
U9	25	Male	Student - Computer science

Table 1: Personalia questions

Table 1 contains a summary of personalia information provided by the test users in the evaluation. The first row contains the questions asked, sorted in the order they appear in the evaluation form (see section B). They also have the same number as they have in the evaluation form. In the second row you can see the answers from U1, in the third you can see the answers from U2 and so on until the answers from U9 in the tenth row.

Questions	U1	U2	U3	U4	U5	U6	U7	U8	U9	Avg
4. How useful do you think the logout button is?	4	4	3	1	4	3	4	3	5	3,4
5. How useful do you think the Add buttons function is?	5	4	5	3	4	4	3	4	4	4
6. How useful do you think the domain log is?	3	4	4	1	4	2	5	5	2	3,3
7. How useful do you think the recent log is?	4	4	2	1	3	3	4	5	4	3,3
8. How useful do you think the notes field is?	4	5	2	3	4	3	3	2	4	3,3
9. How useful do you think the image function is?	5	3	4	3	5	3	2	3	4	3,5
10. How would you rate the settings page?	4	3	4	3	4	2	4	4	5	3,67
13. Overall, how difficult/easy to use did you find the application?	2	2	2	1	3	2	1	2	1	1,8

Table 2: The numbered answers in the evaluation

Table 2 contains the questions in the evaluation that the test users would answer with a score from 1 to 5 and what the test users answered to them. The first column contains the questions asked, sorted in the order they appear in the evaluation form. They also have the same number as they have in the evaluation form. In the second column you can see the answers from U1, in the third you can see the answers from U2 and so on until the answers from U9 in the tenth column. The eleventh column contains the average score for the question asked in the same row. So the number in the eleventh column, second row contains the average score for the question asked in the first column, second row and so on until the ninth row.

5.3 How useful do you think the logout button is

The answers here are mostly positive except for U4 who gave the logout button the lowest score possible. On the other side of the spectrum U9 gave it the highest score possible. The average score of this question is 3,4. The conclusion from this result is that according to the test users the logout button is the third most useful function in the sidebar. However the logout button is only mentioned two times in later questions, and both times were bug related issues with it (see section 5.15).

This button was intended as a proof of concept, but the group seem to think it was useful. This fact opens up for putting more work into perfecting it, by fixing the issues mentioned by the test users, and creating more buttons of this concept. Future work could start with functions such as login and next page/article as mentioned earlier in this thesis.

5.4 How useful do you think the Add buttons function is

The add buttons function received an average score of 4, which is the highest of all the functions in the sidebar. This response was surprising because this function has the most issues associated with it of all the functions in the sidebar. It was also suspected that because the users did not have the option to only remove one custom button, instead of having to remove all of them, would have a negative impact on the score of this function. The lack of ability to remove a single button added is mentioned as a suggestion for improvement by U1 and U8.

The add custom buttons function is based on improving the bookmarks function without making it redundant. Because this is a function present in all major web browsers, it is safe to assume that the test users are familiar with it. U1, U3 and U5 has mentioned in their answers that they found a use for the add custom buttons function in a way that works very well in combination with the bookmarks function. This is reflected in the score that these three test user gave this function (scores: 5, 5, 4).

Based on the answers to this question the add custom buttons function is found useful by the test users and should therefore stay in the sidebar in a future version of the extension.

5.5 How useful do you think the domain log is

The domain log shares the lowest average score (3,3) with the recent log and the notes field. The domain log also has the most varying score of all the functions, with scores ranging from really low (scores: 1, 2, 2) to really high (scores: 5, 5). These results indicate that the opinion on the domain log is split between being either the worst function in the extension or the best.

5.6 How useful do you think the recent log is

The recent log shares the same average score (3,3) as the domain log, but it has a more balanced distribution of scores consisting mostly of 4 and 3. The reason behind this plane distribution of scores could be because browsers already have similar functionality with the browser history, and other functions based on it such as the back button. The test users therefore know what to expect from the recent log and how to use it.

5.7 How useful do you think the notes field is

The notes field shares the same average score (3,3) with the domain log and the recent log. The reason behind this low score is uncertain, since it is only mentioned in later answers by U1 who suggested an improvement to it (see section 5.16) and by U2 who mentioned the notes function being the thing he liked the most about the extension (see section 5.14).

5.8 How useful do you think the image function is

The image function received the second highest average score (3,5) of the functions in the sidebar. Of the test users, U1 and U5 gave the image function the highest score possible (score: 5), and only U7 gave it a score below 3. U7 also provide a reason why he gave the image function a low score in his answer to question 16 (see section 5.14).

5.9 How would you rate the settings page

The reason behind asking the test users to rate the settings page is to find out if they liked it despite the low number of adjustable settings. With an average score of 3,67 and five test users giving it a rating of 4, it is safe to assume that the test users like the settings page. The settings page is discussed more in the next two sections.

5.10 Do you think the settings page was easy to understand

The reasoning behind this question in the evaluation was to find out if the design behind the settings page is user friendly enough that little, or no misunderstandings would happen when using it. This seems to be the case as all nine of the test users mentioned in some way or another that the settings page was easy to understand.

In addition to saying that the settings page was easy to understand U7 said *"The layout of the settings is easy to understand, might be more clear if it was below each other."* as a suggestion to improve the layout of the settings page. This suggestion is similar to one of the design alternatives to the settings page mentioned in section **??**.

U1 and U9 gave an explanation as to why the settings page was easy to understand. U1 said *"It was very easy since the change that I made showed up immediately after, and I could see what I changed."* and U9 said *"Not hard at all, very concise description of all options."*

5.11 Was there anything you were missing on the settings page

The idea behind this question was to find out if there were any options that were dearly missed on the settings page. To this U5, U6 and U8 answered that there was nothing that they were missing on the settings page, and U4 left the question unanswered.

Only U2 and U7 mentioned missing the ability to change the placement of the sidebar, but later when asked about suggestions for improvements to the extension U1, U2, U7, U8 and U9 had suggestions that involved the location and size of the sidebar. These options are discussed in section 5.16.

U1 had a few suggestions. She wrote *"Maybe there could be more options? So I could decide if the notes should be for a domain or a specific page or make some buttons different from others."* All of these suggestions are doable and the test user also mentioned them in the suggestions for improvements question in section 5.16. They will be discussed there.

U3 said *"Text font?"* This is easily doable and would use the same functions as the options changing the color of the text on buttons. The difficult part would be finding a good solution for the user to view and select the available fonts, as the collection of fonts supported by chrome is quite large[29].

U9 answered *"The possibility to set the amount of time the sidebar should use to fade in/out."* During testing, this test user said that he was unable to open the sidebar. It turned out that the test user was double clicking the page action icon which caused the sidebar to open and close itself without giving any visual signs to the user that it had done so. Adding this suggestion to the settings page would not solve the issue, but it would provide a useful option to users wishing a quicker response when clicking the page action icon. This issue is discussed further in section 6.1.4.

5.12 Overall, how difficult/easy to use did you find the application

The average score on this question is 1,8 so according to these results the extension is quite easy to use. The test users were to answer this question on a scale from 1 to 5 where 1 would mean that the user had next to no difficulty in using the extension and 5 would mean that the user found it very difficult to use. The highest answer to this question was from U5 who answered 3. This answer being higher than the others could be explained by the test user's age or that she is less experienced in using web browsers than the other test users.

[29]`https://www.google.com/fonts`

5.13 What did you find difficult/easy to use

This question was added to the evaluation to find out if there is a piece of the extension that the test users found particularly difficult to use and to find out if there was a part of it that they found easy to use.

U2, U6, U7 and U9 did not mention anything that they found difficult to use.

U1 did not get the logout button to work on the uit-mail page. The issue here is that the logout button does not exist in the html document until the menu on the site is opened. This is discussed in more detail in section 6.1.7.

U4 and U7 both said that reading the user manual made the extension easy to understand, but U4 found the add custom button function a little confusing. He said *"when I first tried it the add a new button function was a little confusing, since i didn't know how it would work, or what i was going to use it for."* Other mentions about the add buttons function were mentioned by U1. She said *"It was easy to add buttons or cancel them, but i would like to be able to delete only one button at a time, not all of them. "*

U1 also said *"I can not make buttons if the link is to the far left, since the sidebar is in the way."* This could be solved by allowing the user to move the sidebar to the right side of the browser without having to go to the settings page and do it.

U3 said *"Difficult to remove when in fullscreen"*. Here the user refers to the issue discussed in section 6.1.5.

U8 mentions a noticeable issue that the sidebar does not work at all on the site lovdata[30]. She said *"The application was difficult to use on the site: lovdata.no. The add button and the domain log does not work consistently on the site. The domain log and the add buttons disappears when you refresh the site."* The cause of this issue is unknown, and it only occurs on U8's computer. The severity of this issue is quite large as U8 mentions that the site lovdata is where the extension would be most useful for her (see section 5.14).

Only two users (U4 and U5) mentioned difficulties that are not related

[30]`www.lovdata.no`

to lack of functionality or bugs. U4 found the add custom function a little confusing, although he answered that the extension was very easy to use in question 13(see table 2). The test user who found the extension most difficult is U5. She answered *"Needed some help/explanation, but when I got it it was easy."* Here the test user is referring to taking contact with me because of difficulties understanding what the functions in the sidebar was supposed to do, even after reading the user manual.

5.14 Is there anything you liked about the application

The idea behind this question is to identify the parts of the extension that the test users liked the most and allow for them to explain why if they wished to.

The functions the test users liked the most is the add custom buttons function, as it is mentioned by four test users (U1, U3, U5 and U9). The add custom buttons function is intended to work as bookmarks inside a domain. By the answers from the test users it seems like the intended way also is the way they were using it. U1 saying *"And that I could save my favourite recipies on a blog aboute food. Then I could have a shortcut to the blog on the toolbar with bookmarks for easy access to new entries and use the sidebar to find my favourites."*. U3 saying *"Finding the series you watch the most on a streaming site."* and U5 saying *"I could add a button within FB and go directly to that page."*

The domain log is not mentioned directly and the recent log is only mentioned by name once by U9, but answers similar to what U7 answered *"I like the ability to go back a few pages in one single click."* is interpreted to mean both the recent log and the domain log. So including this, the recent log is mentioned three times (by U1, U7 and U9) and the domain log twice (by U1 and U7). What the test users liked was the ability to jump back a few pages without having to click several times on the back button or finding the page they are looking for in the browser history.

U1 and U6 mentioned the image function. U6 simply said *"The picture saver"* and U1 said that it was *"very easy to share a picture via instant* *messages or to post f.ex. on facebook."*

When asked about how useful the notes field is, only U2 gave it the highest score possible (score: 5). To describe why, he wrote *"I can finally write down were i stopped watching a youtube video."* Due to the wording of this answer it sounds like the notes field is a solution to an issue that has been bothering the test user for some time.

5.15 Is there anything you did not like about the application

The idea behind this question was to find out what parts, in specific, the test users did not like about the extension. The answers are diverse as no two test users mention a similar issue.

U1 did not like that she was unable to *"have pictures as buttons, and then see a miniature picture in the sidebar."*.

The log out button did not work properly on facebook for U2 as he *"had to press multiple times to log out from some sites(Facebook)."*.

U3 did not like that he had to close the sidebar before entering fullscreen.

U4 found the extension *"Very clunky/manual to use"*.

U5 is uncertain in what she did not like in the extension as she answered *"Don't know really. Must use it for more than two days to tell."*

U6 simply answered *"no, but it looked ugly."*

U7 said *"The image function has no worth for me. Maybe its useful if you are creative and like saving a lot of images about things you could make."*

U8 has issues using the application at the same time as the functions on a website because *"A part of the page disappears behind the application."*

The function that got the lowest score in usefulness from U9 was the domain log (score: 2, see table 2). He misunderstood how the domain log works as his answer here is *"The domain log. All links (as far as I found) do appear in the recent history, so the domain log is redundant"* This misunderstanding could either be that the domain log is not properly explained in the user manual, or that the test user did not open a large enough number of sites during testing that the recent log and the domain log never contained different entries.

5.16 Do you have any suggestions for improvements

This question was added to the evaluation because it will be useful for the future work of the extension to know what the test users would like to change in the extension, either by changing, removing or adding functionality to it.

This section will go through all of the suggestions from the test users first sorted by the amount of users who suggested it and then by user number (U1, U2, U3 and so on). The only test user who did not have any suggestions for improvement was U5.

- - *"In addition it would be nice if the sidebar "integrated" with the site, so it does not cover the left side."* Suggested by U1
  - *"I would like if you could add a setting so the sidebar dosn't cover the left side of page your on."* Suggested by U2
  - *"The ability to resize the bar and changing it to the other side."* Suggested by U7
  - *"Make the application so it does not hide parts of the page."* Suggested by U8
  - *"There should also be a possibility to adjust the sidebar width and height."* Suggested by U9

  These suggestions are grouped because they all mention the size or location of the sidebar. The suggestions by U2, U7 and U9 could be implemented by adding options to the settings page that allowed the user to change the size and location of the sidebar. The only thing the user would change would be the x and y coordinates (called the top and left attributes in html) of the sidebar to change the location, and the width and height percentages (so that the sidebar would adapt to the user changing the size of the browser window) to change the size of the sidebar.

  The suggestion by U1 was attempted in the early stages of development, but was scrapped because it involved moving all the elements on the website, an action that should be avoided if possible because it could interfere with the website's functionality.

Adding a function that allows the user to move the sidebar to the opposite side of the window without having to enter the settings page, would be the easiest way of appeasing the suggestion by U8.

- - *"being able to delete one button."* Suggested by U1
  - *"It would be nice if you can remove specific added buttons."* Suggested by U8

  Allowing the user to delete only one of the added custom buttons in the sidebar at a time has been the intention from the start. Due to issues related with custom buttons being two different types of elements with different content made it difficult to create a reliable function that would delete only the button the user chose. A last minute workaround function that deleted all the added custom buttons was made to allow for some way to delete added buttons in the extension that was sent to the test users. A future version of the extension should have a method of deleting the added custom buttons similar to the way saved images are deleted in the extension.

- - *"Prettier design."* Suggested by U6
  - *"Some web-designer should look at the design."* Suggested by U9

  These two suggestions both involve the visual design of the extension. U6 mentioned several times in the evaluation that the design of the extension looked ugly (in questions 14, 16 and 17). There is no denying that the sidebar with its default visuals is not pretty. Other than the options that the user get on the settings page, there has been barely any focus on making the sidebar and the settings page look appealing. For a future version of the extension there is definitely work needed to make the default design of both the sidebar and the settings page better looking.

- *"different styles for different buttons."* Suggested by U1

This could be done by giving the elements in the sidebar a more diverse standard id. All the elements in the sidebar already have a id that starts with ”ext”, so for the buttons in the domain log they can have ”extdom” as id. Then the settings page could be altered so the user can change the CSS for all the buttons that has an id that starts with ”extdom.”

- *”notes for a specific page.”* Suggested by U1

  The reason for notes being connected to a domain rather than each specific web page was because there are a lot of pages that include variables in their URL's. These variables can change between each visit so for a user it would seem as the extension sometimes remembers the notes and sometimes not. So for a compromise, the notes could allow for the users to create tabs for notes within a domain or just in general. Selecting between the note's tabs could be through a drop down box to allow for a large number of tabs.

- *”One log for different domains and one for sites in the domain.”* Suggested by U1

  This suggestion would make the difference between the recent log and the domain log clearer. It would also give the user a better overview of the recent browsing history. This is an idea that should be experimented with in future work of the extension.

- *”I would also like to be able to save gifs.”* Suggested by U1

  The extension is already able to save gifs. The only requirement is that the web page the gif is found on has put the gif in an IMG element. Some pages are known to display gifs in VIDEO tags so the users are able to pause them when played. It should be possible to save such gifs if the source location of the gif is accessible by the extension.

- *”Fullscreen issue.”* Suggested by U3

  The issue U3 is referring to here is mentioned in section 5.13 and the issue is discussed in section 6.1.5.

- *"Need to refresh the page after opening it through Google."* Suggested by U3

  The issue U3 is referring to here happens when the user is doing a web search through Google[31] and opening a web page through one of the search results. This issue is discussed in more detail in section 6.1.3

- *"If it remembered what domains you like to use it in, it could automatically open on those domains, instead of having to open it manually every time"* Suggested by U4

  This suggestion describes a functionality that allows the extension to, over time, adapt to a user's behavior. This functionality is discussed more in future work (see section 6.2.4).

- *"The recent log I feel is not very necessary because you have that option in the recent tabs menu, which is equally accessible to the sidebar log."* Suggested by U4

  Here U4 refers to the recently closed tabs menu. In the chrome web browser this is located in the tool menu and it contains a list of the recently closed tabs. It differs from the recent log in the extension in how new items are added to the list. The recently closed tabs menu only adds a new item to its list when a tab is closed. The recent log adds a new item to its list whenever a new page is loaded. If a user only uses one tab when browsing the web, the recently closed tabs menu will only contain the page that was open the last time the user closed the web browser. In the same situation, the recent log will show the last 10 web pages that the user opened.

  U4 must either have misunderstood how the recent log works, or how the recently closed tabs work. This misunderstanding could be the reason as to why U4 gave the recent log (and the domain log) a score of 1 in the evaluation form.

- *" I feel like the hide buttons are not very useful, you could program*

[31] `www.google.com`

the button into the text, or have a button named "recent log" and have hide/show programmed into it." Suggested by U4

The suggestion from U4 here is to make the functionality of the hide button execute when the header next to it is clicked, or to move the text from the header into the hide button. This suggestion would make the sidebar use the space it has more efficiently. For a future version of the extension the functionality of the hide buttons would be implemented into the headers, with a small visual indication (such as an arrow pointing up or down next to the header) as to whether clicking it would hide or show the elements in the container below.

5.17 Conclusions

Based on the results of this evaluation it is concluded that different users have different needs. No one function in the extension received a score higher than 3 from all test users, and similarly no one function received a score lower than 3 from all test users. This indicates that a potential feature for the extension should be to allow the user to decide what functionality the sidebar should contain.

The users who gave the lowest scores in usefulness are U4 and U6 (see table 2). The similarities between them is that they both mention that they did not like the design of the extension. By their answers, making the extension more pleasant to look at would improve the usefulness of the extension without making changes to the functionality. U9 also mention the design of the extension, but the scores he gave are higher than both on almost every function. This could be explained by U9's background, which is from computer science and therefore has a much bigger focus on functionality rather than visuals.

If a test user misunderstands what a function is supposed to do he is very likely to give it a lower score than the other test users. This is shown by U4 having misunderstood the differences between what the recent log does and the recent tabs menu, and by U9 having misunderstood the difference between the domain log and the recent log. The conclusion here is that functions must be clear in what they do and in how they differ from each other. In this case the suggestion from U1 (discussed in section 5.16) would be something to consider as it would make the differences between the recent and the domain log clearer for the users.

The amount of customization options is insufficient for the majority of users. Five of the test users could think of options that they though were missing on the settings page. However four of the test users suggested options that can be added to the settings page by reusing existing functionality already implemented in the settings page.

Out of nine test users, seven of them found at least one function in the extension that they found very useful and none of the functions received a average score below 3. This proves that all of the functions in the sidebar are useful, but there is room for improvement. The test users have given a lot of good suggestions that are going to be valuable during the development of a future version of the extension.

6 Known issues and future work

This section will go through some of the known issues in the application and the future work for the extension.

6.1 Issues

6.1.1 Custom buttons have no text in them

The create custom button function allows the user to turn almost everything into a button, and there is a lot of empty elements on a page, so if the user creates a button for an empty element the button itself is going to be empty. This can also happen if the user clicks on a another element than the one intended. Such as when trying to make a button of a text link and the user clicks between the letters the application might think that the user intended to click on the element behind the link.

6.1.2 Some pages have changed after installing the extension

The CSS of the extension is applied to most elements that has an id that starts with ”ext” so this might apply to some other elements on a page. They way to solve this would be to have a longer startid than ”ext”. This issue has only happened on one site[32] so far.

6.1.3 The page action icon does not show up sometimes

This sometimes happens if the user types a url into the addressbar manually or after opening a link on the result page of a web search. The underlying cause of this issue is unknown and it is uncertain if it has something to do with the way the extension detects that a new page has been opened. The frequency of this issue also seems to vary between computers following no noticeable pattern.

[32]dl.acm.org

6.1.4 Double clicking the page action icon makes the sidebar not show

This happens because when the user clicks the page action icon the sidebar starts to fade in over the course of half a second. So if the user clicks the icon again fast enough the sidebar will start to fade out again before it has even started to show itself. So for the user it will look like the sidebar never started to appear at all. This could be fixed by adding a timer to the event listener so that the sidebar cannot be closed before it has completely shown itself.

6.1.5 The sidebar cannot be closed in fullscreen mode

This issue happens when the sidebar is open and then entering fullscreen mode on the browser either through pressing the f11 key or opening a video in fullscreen. This happens because the only way to close the sidebar is to click the page action icon, and when fullscreen is enabled this button is hidden. The way to fix this would be to create a button in the sidebar that closes it or to allow the users to choose a hotkey for closing and opening it on the settings page.

6.1.6 Some pages does not show up in the recent log or the domain log

It is a known issue is that on some pages, especially ones that are used for streaming video content, such as youtube.com or twitch.tv seem to not trigger the events that the extension use to notice when a new page has been opened. This is likely to remain an issue until a reliable method of noticing the opening of a page has been found.

6.1.7 Logout button not working on pages with menus

The logout function is unable to detect the logout button on most pages that use a drop down menu. This is because the logout button simply does not exist in the html document until the menu is opened. To fix this issue the extension would need a reliable way of opening the menu before attempting to locate the logout button.

6.1.8 The extension does not work properly on some sites

This is an issue that was noticed by one of the test users (U8). The sidebar simply refuses to work properly in the domain lovdata.no, but only on U8s personal computer. Due to being unable to recreate this issue without using U8s computer, no solution has been found.

6.2 Future work

This section contains ideas for future work on the extension developed in this thesis.

6.2.1 Image tags

A future version of the extension would allow the user to add tags to a saved image. This would allow for functionality that can search through the saved images based on a input string from the user and present the images with tags that match the input string. This would make it much easier to locate an individual image as the collection of images grows large.

6.2.2 Notes

An improvement to the notes field would be to allow the users to search through all of their saved notes and find the page that the notes were saved from. This would make the notes field work as tagging service for web pages. So for example if the user added the word ”news” in the notes field on different news sites. Then the user could search for the tag ”news” and receive a list of pages that contains the word news in the notes field.

6.2.3 Settings page

Future work for the settings page would include changing the layout into one that would scale with the number of options added. The test users suggested a few options that could be added to the settings page, but adding them would exceed the space currently available in the 3x3 grid layout on the settings page. The most logical solution would be to put the options into rows. So when adding one new option to the settings page would only require adding one new row to the settings page. If the current layout would be kept then adding one more option to the settings page would require the layout from changing from a 3x3 grid to a 4x3 grid. This would make the settings page to create two empty squares, or new options would have to be added in groups of three at a time.

6.2.4 Adaptive behavior

The future work for the extension, in the context of adaptive behavior, would be to make the extension able to detect patterns in the users behavior. This would involve detecting what pages the user prefers to open the sidebar and then open it automatically on those pages. Moreover, the extension could take notice if the user tends to only use one of the functions in the sidebar when visiting a specific page, and the automatically hide the other functions.

These changes would make the extension have the characteristics of an adaptive user interface.

6.2.5 Google translate

Chrome has a translate function. It works by asking what languages the user understands and when a web page written in a language that the user does not understand a toolbox will appear asking if the user would like the page to be translated. An improvement to this functionality would be to allow the user to translate a web page to whatever language on request. It is not certain if this is possible using the existing translate functionality within chrome or if the extension would need to use other solutions, such as the google translate api[33].

[33] `https://cloud.google.com/translate/`

7 Conclusion

In this thesis an extension that explore the customization, personalization and adaptation possibilities in the chrome web browser, while simplifying the users access to information, has been developed and evaluated.

This thesis states that the functionality of todays web browsers offer is not sufficient and can be improved. To attempt prove this the extension implements functions that improves the existing functionality available in the chrome web browser. These functions include adding custom buttons within domains, easy access to the web pages most recently visited (globally and per domain) and a simplified method of storing and accessing stored images. Moreover, this thesis presents an evaluation whose results indicate the usability of the functions implemented in the extension.

The goal of this thesis was to *design, implement and test an application that explores personalization, customization and adaptation possibilities in a web browser.* The result of the evaluation confirms that this goal is achieved to some degree. As the settings page of the extension was rated highly by the test users (avg score: 3,67) and only one of the test users suggested a customization option that can not be implemented by reusing the existing functionality on the settings page. Future work on this extension have the potential of notably increasing the web browsers capability to adapt to a user.

References

[1] Akiki, P. A., Bandara, A. K. & Yu. Y., 2014. Adaptive Model-Driven User Interface Development Systems. *ACM Comput. Surv.* 47, 1, Article 9 (May 2014), 33 pages.

[2] Eirinaki, M., Vazirgiannis, M., 2003. Web mining for web personalization. *ACM Trans. Internet Technol.* 3, 1 (February 2003), 1-27.

[3] Gao, M., Liu, K., & Wu, Z., 2010. Personalisation in web computing and informatics: Theories, techniques, applications, and future research. *Information Systems Frontiers 12*, 607629.

[4] GVUs WWW Surveying Team. *GVUs 10th WWW User Survey*, 1998.

[5] Google inc., *Chrome Extensions Developer Overview* (Accessed September 10, 2015) URL:`https://developer.chrome.com/extensions/overview`

[6] Google inc., *Chrome Extensions Developer Storage* (Accessed September 10, 2015) URL:`https://developer.chrome.com/extensions/storage`

[7] Jason, B., Calitz, A., & Greyling, J., 2010. The evaluation of an adaptive user interface model. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists* (SAICSIT '10). ACM, New York, NY, USA, 132-143.

[8] Mackay, W. E., 1991. Triggers and barriers to customizing software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '91), Scott P. Robertson, Gary M. Olson, and Judith S. Olson (Eds.). ACM, New York, NY, USA, 153-160.

[9] Marathe , S., & Sundar, S. S., 2011. What drives customization?: control or identity?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 781-790.

[10] Nadeem, T., Killam, B., 2001. "A study of three browser history mechanisms for Web navigation," in Information Visualisation, *Proceedings. Fifth International Conference on* , vol., no., pp.13-21.

[11] York, R., & Pouncey, I. 2011. Beginning CSS : Cascading Style Sheets for Web Design (3rd Edition). Hoboken, NJ, USA: Wrox. Retrieved from `http://www.ebrary.com`

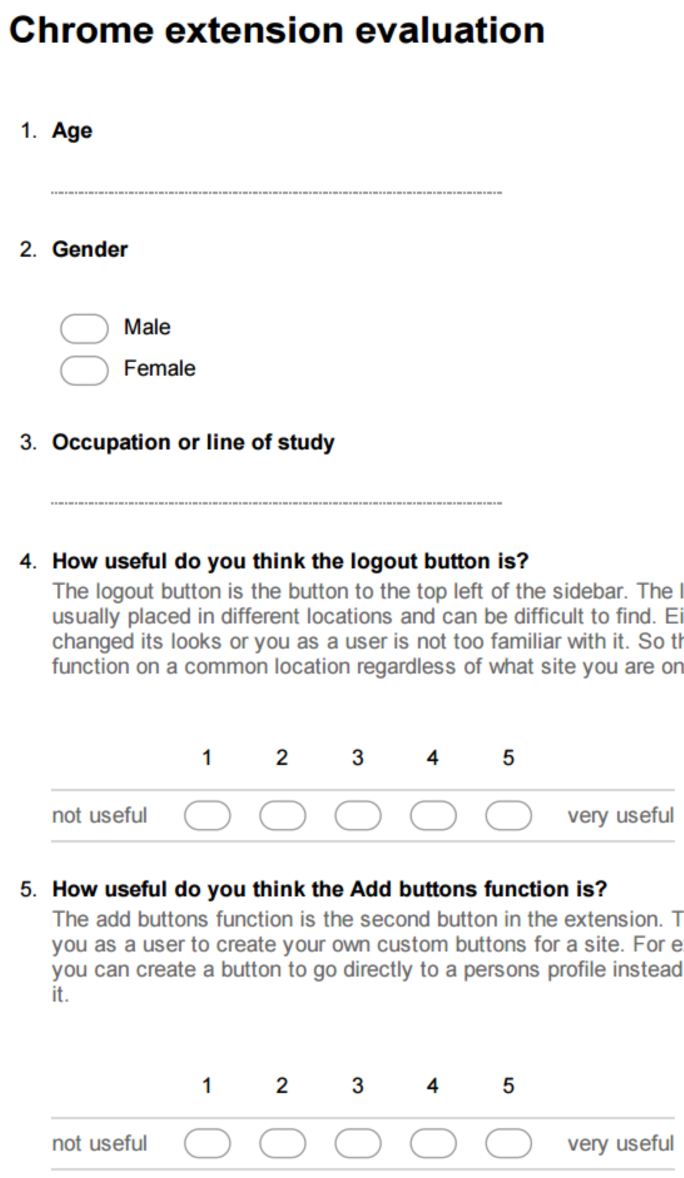
A Code Snippets

```javascript
// Change one attribute of the css using regex
function changeCSStext(selector, attribute, replacement){
  var searchtext = csstext;
  // split css text into two parts, before the value of the attribute to change and after
  var block1 = searchtext.search(selector);
  var block2 = searchtext.substring(block1).search(attribute);
  var block22 = searchtext.substring(block1+block2).search(/:/) + 2;
  var block3 = searchtext.substring(block1+block2+block22).search(/;/);

  var replacementCSSstart = searchtext.substring(0, block1+block2+block22);
  var replacementCSSend = searchtext.substring(block1+block2+block22+block3);
  // put the css back together
  csstext = replacementCSSstart + replacement + replacementCSSend;
  // update the page css so the changes are shown in the preview elements
  var style = document.getElementById("extCSS");
  style.parentNode.removeChild(style);
  $("head").append(csstext);
}
```

```javascript
// Page Action listener
chrome.pageAction.onClicked.addListener(function(tab) {
  chrome.tabs.getSelected(null, function(tab) {
    chrome.tabs.sendRequest(
      tab.id,
      {callFunction: "toggleSidebar"},
      function(response) {
        console.log(response);
      }
    );
  });
});
```

B Evaluation form



Chrome extension evaluation

1. **Age**

2. **Gender**

   Male

   Female

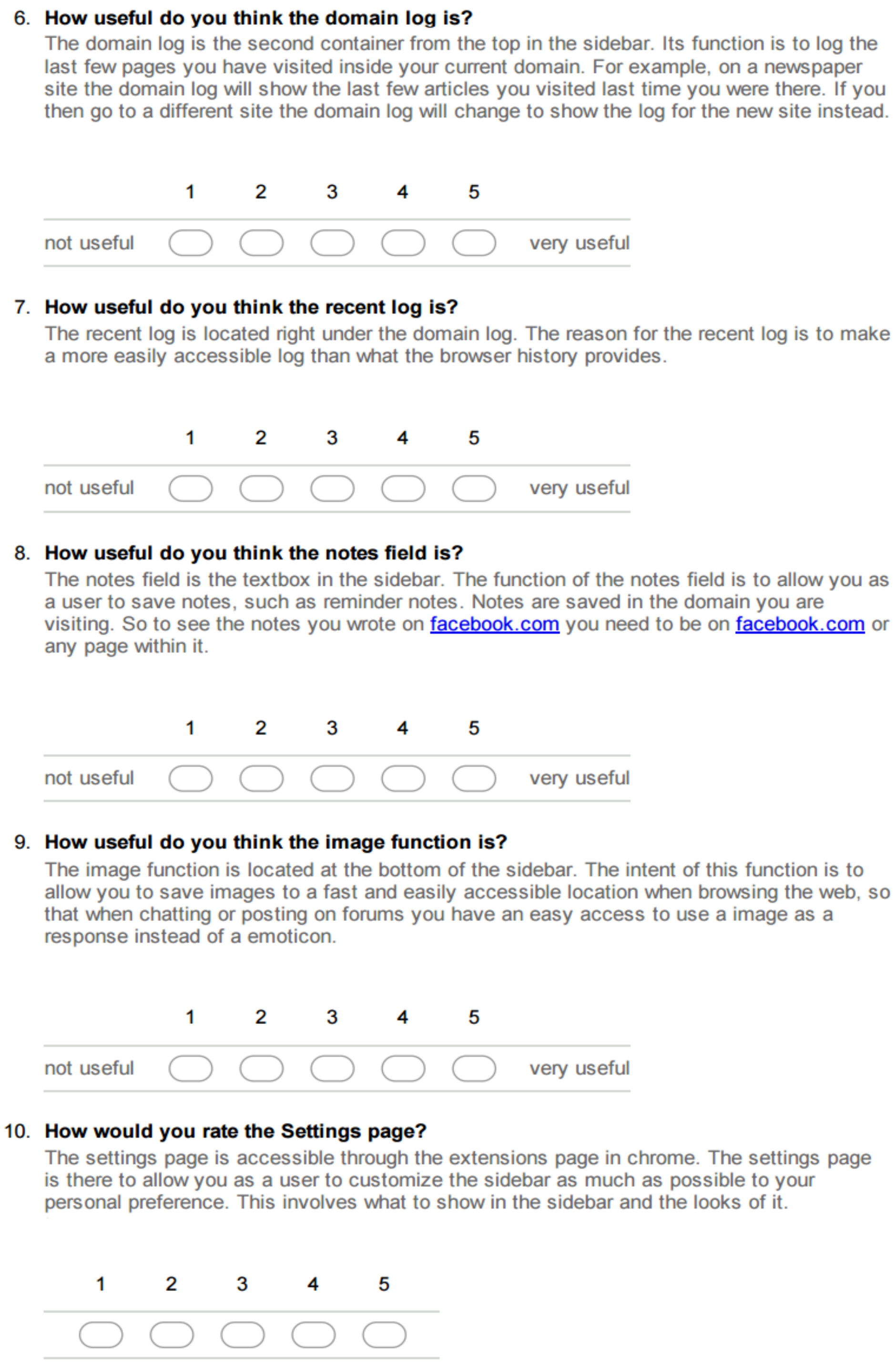
3. **Occupation or line of study**

4. **How useful do you think the logout button is?**
   The logout button is the button to the top left of the sidebar. The logout button on a site is usually placed in different locations and can be difficult to find. Either if the site has recently changed its looks or you as a user is not too familiar with it. So the idea is to place such a function on a common location regardless of what site you are on.

	1	2	3	4	5	
not useful						very useful

5. **How useful do you think the Add buttons function is?**
   The add buttons function is the second button in the extension. The intent of this is to allow you as a user to create your own custom buttons for a site. For example, on Facebook.com you can create a button to go directly to a persons profile instead of having to search to find it.

	1	2	3	4	5	
not useful						very useful



6. **How useful do you think the domain log is?**
The domain log is the second container from the top in the sidebar. Its function is to log the last few pages you have visited inside your current domain. For example, on a newspaper site the domain log will show the last few articles you visited last time you were there. If you then go to a different site the domain log will change to show the log for the new site instead.

	1	2	3	4	5	
not useful	○	○	○	○	○	very useful

7. **How useful do you think the recent log is?**
The recent log is located right under the domain log. The reason for the recent log is to make a more easily accessible log than what the browser history provides.

	1	2	3	4	5	
not useful	○	○	○	○	○	very useful

8. **How useful do you think the notes field is?**
The notes field is the textbox in the sidebar. The function of the notes field is to allow you as a user to save notes, such as reminder notes. Notes are saved in the domain you are visiting. So to see the notes you wrote on facebook.com you need to be on facebook.com or any page within it.

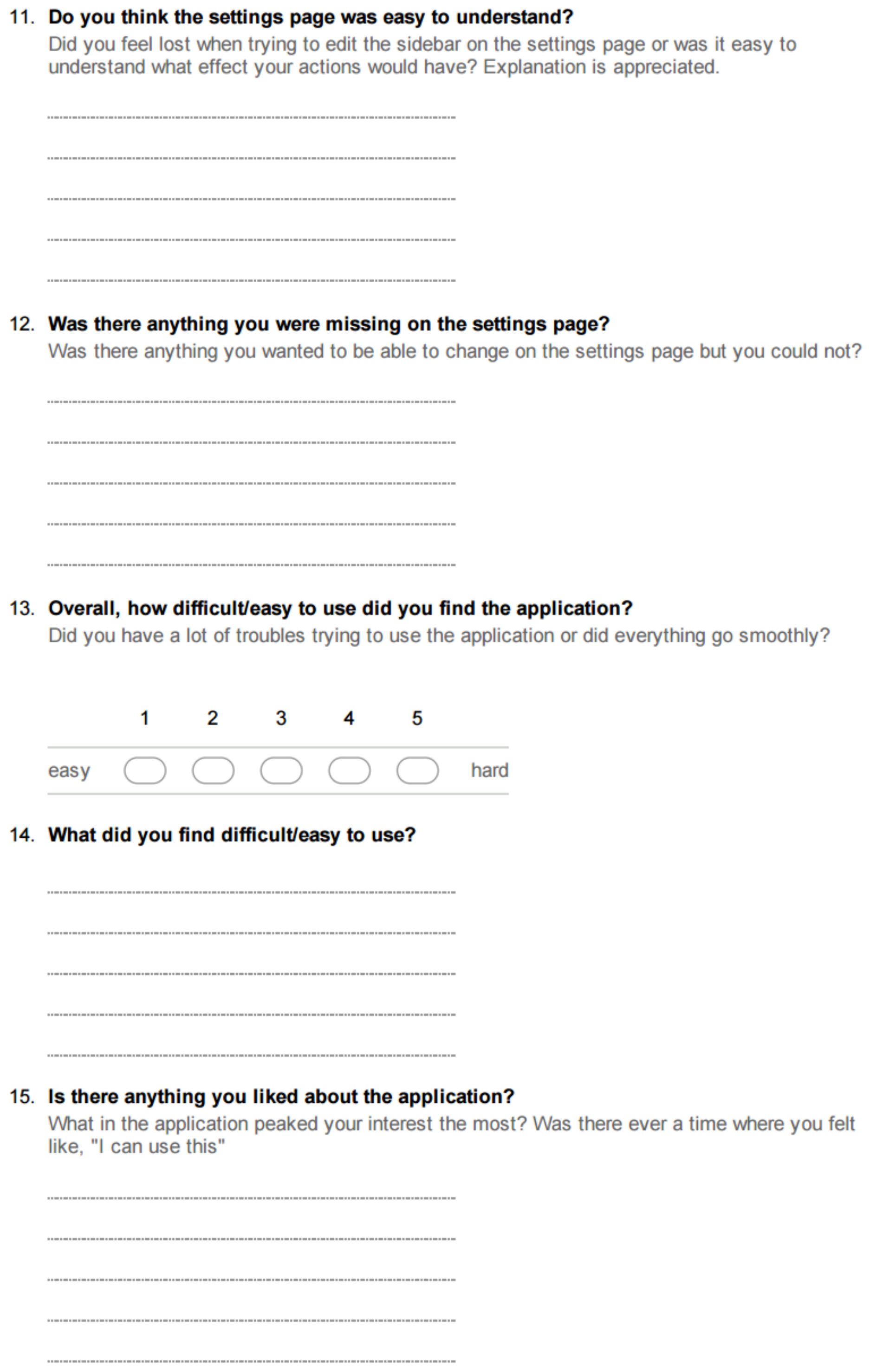
	1	2	3	4	5	
not useful	○	○	○	○	○	very useful

9. **How useful do you think the image function is?**
The image function is located at the bottom of the sidebar. The intent of this function is to allow you to save images to a fast and easily accessible location when browsing the web, so that when chatting or posting on forums you have an easy access to use a image as a response instead of a emoticon.

	1	2	3	4	5	
not useful	○	○	○	○	○	very useful

10. **How would you rate the Settings page?**
The settings page is accessible through the extensions page in chrome. The settings page is there to allow you as a user to customize the sidebar as much as possible to your personal preference. This involves what to show in the sidebar and the looks of it.

1	2	3	4	5
○	○	○	○	○



11. **Do you think the settings page was easy to understand?**
Did you feel lost when trying to edit the sidebar on the settings page or was it easy to understand what effect your actions would have? Explanation is appreciated.

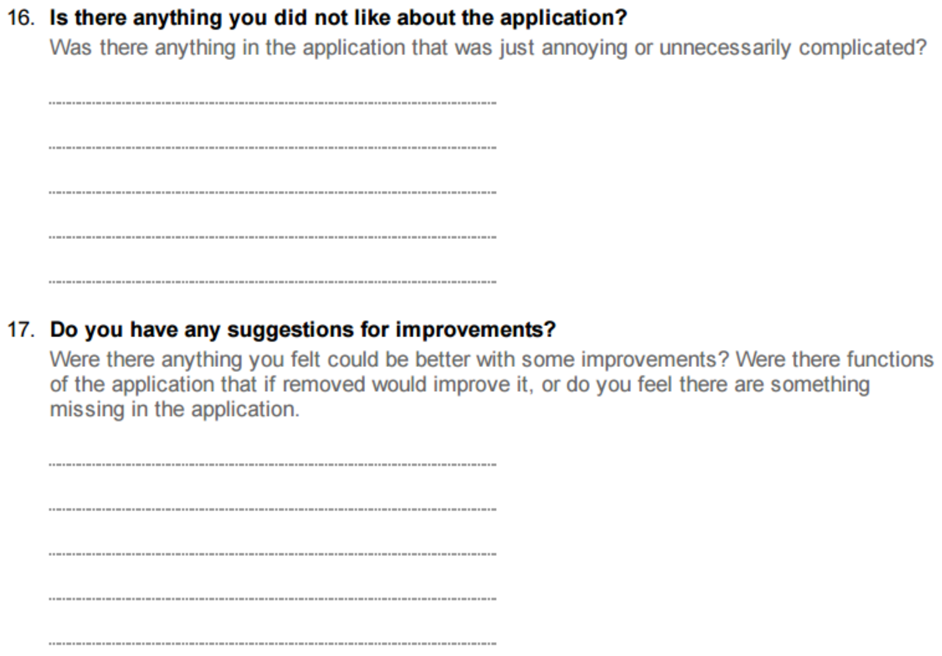
12. **Was there anything you were missing on the settings page?**
Was there anything you wanted to be able to change on the settings page but you could not?

13. **Overall, how difficult/easy to use did you find the application?**
Did you have a lot of troubles trying to use the application or did everything go smoothly?

	1	2	3	4	5	
easy	◯	◯	◯	◯	◯	hard

14. **What did you find difficult/easy to use?**

15. **Is there anything you liked about the application?**
What in the application peaked your interest the most? Was there ever a time where you felt like, "I can use this"



16. **Is there anything you did not like about the application?**
    Was there anything in the application that was just annoying or unnecessarily complicated?

    ....

    ....

    ....

    ....

    ....

17. **Do you have any suggestions for improvements?**
    Were there anything you felt could be better with some improvements? Were there functions of the application that if removed would improve it, or do you feel there are something missing in the application.

    ....

    ....

    ....

    ....

    ....