Hochschule für Technik
und Wirtschaft Berlin

University of Applied Sciences

Development of a Chrome Extension for Personalized Filter Suggestions

Abschlussarbeit

zur Erlangung des akademischen Grades:

Bachelor of Science (B.Sc.)

an der

Hochschule für Technik und Wirtschaft (HTW) Berlin
Fachbereich 4: Informatik, Kommunikation und Wirtschaft
Studiengang *Angewandte Informatik*

1. Gutachter_in: Prof. Dr. Elena Schüler
2. Gutachter_in: Birol Aksu

Eingereicht von Sunan Regi Maunakea 566144

22. August 2022

Danksagung

[Text der Danksagung]

Zusammenfassung

[Text der Zusammenfassung]

Abstract

[Summary of the thesis]

Contents

List of Figures

1 Introduction

1.1. Background

It is a common belief in modern society that the more choices, the better–that the human ability to manage, and the human desire for, choice is unlimited. One study showed that the existence of choice increases motivation and enhances performance on doing tasks [12]. However, another study has shown that although having more choices might appear desirable, it may sometimes have negative effects on human motivation [6]. In our digital and website-driven era, these studies can be applied on e-commerce and one of the solutions to this problem is a search-and-filter functionality. Amazon is one of the corporate giants for e-commerce, that implemented this feature in their platform. Another example would be marta.

Marta as a business is currently best described as a marketplace between caregivers and families requiring 24-hour care. 24-hour care can be defined as living in a household with the person in need of care for a certain period of time. This means that caregivers are primarily responsible for basic care and household chores. In addition, they support the person in care's relatives in need of assistance in carrying out the activities they wish to do. Marta as a marketplace connecting families with caregivers is competing against more traditional agencies, where it can take several days or even weeks to find a family for a newly signed up caregiver or the other way around.

1.2. Problem Statement and Objectives

In this rapidly growing world, human has access to almost everything. On the other hand, having too many options may cause negative effects on the human motivation. Imagine browsing for a cheap, small vacuum cleaner on Amazon to replace your trusty 5-years-old broom. You'd find a thousand cheap small vacuum cleaners from different brands, even in different colors. You'd end up spending most of your time comparing which vacuum cleaner would deliver the best performance, of any sort, for its price, even though all of those vacuum cleaners are exactly what you wanted, cheap and small. As a multinational technology company, Amazon aims to allow users to complete a transaction as quickly as possible. In order to achieve this goal, Amazon introduced a search-and-filter bar in their website. It sounds like a fair idea at first, but if users searched for a similar item over and over again, they would need to type in the same letters or words over and over again.

Similar to Amazon, marta would need to enhance their product, to compete in this expeditiously developing business. One way marta can provide a superior experience for both caregivers and families is to speed up the matching process. The caregiver and family inquiry forms are designed to record as much information as possible which can be used during the matching phase. The matches are created by the teams in Berlin and

Romania, but to make their job easier, the technical team in marta compute a "matching score" by which possible matches are sorted. As a result, a number of caregiver profiles with high matching scores will be shown to the family. To quicken the search, filter functionality is also provided. This includes caregiver's earliest starting date, German skills, experience with diseases, etc.

Marta needs to introduce a continually improved, user-friendly filtering functionality to adjust to the needs of families. An example of a user-friendly filter would be to provide quick filter suggestions which the user can click once and the desired results will be shown, instead of letting users select the same filter manually over and over again. In order to determine which filter suggestions are the most beneficial, frequently used filters need to be identified.

1.3. Structure of the Thesis

The bachelor thesis is structured as follows: Besides the introduction in chapter 1, browser extensions are discussed in more detail in Chapter 2 along with the reasons why they are a suitable solution to this issue. Additionally, it describes the extension's architecture and implementation. Based on this, Chapter 3 describes the experimental methodology used for studies conducted with the Chrome extension. The extension's implementation and use in a practical situation are discussed in some detail in Chapter 4 along with some lessons learned. Subsequently, the extension is evaluated and analyzed. Finally, problems are identified and an outlook for possible improvements of the extension is given.

2 Theoretical Basis

2.1. Browser Extension Background

2.1.1. Definition

Browser extensions or addons are third party programs, that can extend the functionality of browsers and improve users' browsing experience [11]. A browser extension, as opposed to a standard web page, is created specifically for a given browser and uses that browser's extension API. It was necessary to choose a browser as a result. There are frameworks that try to make it feasible to create an extension for several different browsers at once. Although the caliber of these frameworks was unclear, it was decided that the expense of potential problems and additional time spent debugging in many browsers outweighed the benefits.

2.1.2. Choice of Implementation Browser

The selection of a browser was based on a number of factors. The first was the browser's usage rate. This is significant since anyone who doesn't already have the necessary browser will need to download and install it. Drop-outs due to the installation process or being unfamiliar with a new browser can significantly raise the cost of conducting the survey. The ease of use of the API and simplicity of implementation were additional crucial criteria. The extension programming process should ideally only need a basic understanding of the extension API. The capabilities of the API offered by the browser was another factor considered. The extension needs to:

1. Store a big amount of data on the client side
2. Read URL - so query parameters can be passed to the extension
3. Modify URL - so frequently used query parameters can be utilized

[outdated] Internet Explorer was the most challenging in terms of implementation simplicity. The features appeared to be restricted, and it needed knowledge of the COM. [8] After reviewing the documentation, it was still unclear how tasks like making HTTP queries or changing the DOM would be carried out. The instructions appears to be primarily concerned with optional features like adding menu items and explorer bars. Thus, IE was eliminated from the list, leaving Firefox and Chrome as the only options.

[outdated] Both the Firefox and Chrome extension APIs had similar features. The fact that Firefox required knowing XUL (XML User Interface Language), Mozilla's XML-based language for creating application user interfaces, was one drawback. [9]

Extensions for the Google Chrome browser can be created entirely in HTML, CSS, and JavaScript. The team members' familiarity with the languages and the simplicity of the solution were appealing. Additionally, it was discovered that the Chrome documentation was easy to grasp and was divided up into sections. Firefox's market share was roughly double that of Chrome. As a result, it was decided to develop a Google Chrome browser extension.

2.2. Chrome Extension Architecture

2.2.1. Chrome Extension Basics

Extensions are built on web technologies such as HTML, JavaScript, and CSS. They run in a separate, sandboxed execution environment and interact with the Chrome browser [4]. They also have access to the APIs that browsers provide for tasks like XMLHttpRequests and HTML5 features on web sites. The following files can be found in an extension:

- A manifest.json file
- One or more HTML files
- Any other files such as CSS or JavaScript needed by the extension to run

The majority of extensions have a background page that contains their primary logic and state. They frequently also contain content scripts that can communicate with websites. Asynchronous message passing is used to communicate between the background page and the content scripts. Additionally, extensions can save data via localStorage and other HTML5 storage APIs.

2.2.2. Manifest Files

A manifest.json file is required for each extension. It includes crucial information about the extension, such its name, version, scripts used for its content, minimum Chrome version, and permissions. The content-scripts field was the most crucial one for this expansion. Each study and content-related webpage need its own content script. Each one was defined in the scripts column, which also mapped each one to the appropriate URLs.

2.2.3. Content Scripts

Content scripts are JavaScript files that are used on websites to add new functionality. They have full control to modify the entire web page because they can directly access the DOM of these web sites. The content script is injected into a tab when the web page is loaded, and runs in the same process space of the renderer of the web tab and can thus access its DOM objects. Injected content scripts in a tab can only communicate with the extension core via Chrome's IPC [7].

2.2.4. Service Worker

The Chrome extension platform switches from background pages to service workers in Manifest V3. A service worker is a script that your browser runs in the background, distinct from a web page, opening the door to features that don't need a web page or user input [3]. This technology allows native-like experiences over the open web, including push notifications, robust offline support, background synchronization, and "Add to Home Screen." Service providers drew some of their inspiration from the background pages in Chrome Extensions, but improvements were added for the web.

Service workers are specialized JavaScript assets that act as proxies between web browsers and web servers. They aim to improve reliability by providing offline access, as well as boost page performance [3]. Websites are gradually improved by service workers through a lifetime akin to that of platform-specific programs.

2.3. React

React is a product of Facebook's engineering team, which is a JavaScript framework for creating user interfaces [2]. Because of its simplicity and straightforward but efficient development process, React is quite well-liked in the developer communities. Interactive user interfaces are simpler to develop with React. It effectively updates by accurately drawing each state's view's constituent parts, and it updates the application's data [5]. The core objective of React is to provide the best possible rendering performance. Its strength comes from the focus on individual components. Using reusable components, it is found to be easy development for developers to design rich UI's. React incorporates with View part from MVC model [8]. React implements One Way dataflow so that it gets easier than traditional data binding. React uses virtual DOM it offers not so complex programming with faster execution. It makes use of composition to create intricate user interfaces out of simple building blocks known as Components [1].

2.3.1. Component

Each component has a render method, which can either return HTML or another React component, and returns a description of what to render. A combination of HTML and Javascript known as JSX is used to describe what should be rendered. A more detailed explanation of JSX is included in the JSX section. It is possible to specify a component as a class or a function. Props and state are crucial components when creating React applications.

2.3.2. States and props

The state of a component allows it to "remember" things, and it can change in response to user interaction or other application-wide actions. The State is optional; components without a state are referred to as presentational components. Components with a state are referred to as stateful components.

Immutable data supplied into a component during development is known as props, or properties. Because a component can function and seem differently depending on the properties supplied into it, props enable React components to be flexible and reusable.

Using props, data moves down the component tree in React. Callback functions are supplied as props so that a child component can communicate with its parent. Callback methods and other data must be passed down numerous layers in large React apps due to the component tree's depth. Props-drilling is a technique that results in tightly connected components and a less maintainable program. This is one of the reasons why complex React apps require architectural patterns.

2.3.3. Class-based components

A class-based Component is created by extending the React.Component class. The state of a class-based component is updated using the method setState() and read by using this.state within the class. Using the setState() method makes the component re-render which is not the case when mutating the state directly. Class-based components include life-cycle methods that can be used to create more complex behavior. It also requires a render method to return JSX elements.

2.3.4. Life-cycle methods

Life-cycle methods are built-in functions that are called whenever a state or prop updates, a component renders, is destroyed, or both.

2.3.5. Function components

A pure function that takes props as input and outputs a JSX element is referred to as a function component. React Hooks are used to give the function component the same access to state and life-cycle methods as class-based components. In compared to class-based components, using function components with hooks might make React applications smaller and more manageable. There are a couple reasons why function component is preverable [9]:

- Faster development, easier to read and test, debug and reusable; because function components do not have state and life-cycle-hook. Function component is a straightforward JavaScript function.
- Performance will be improved since function components are smaller and compiles more quickly than class components.
- There is no need to consider how to divide the component into a container and a standard UI component when utilizing a function component.

2.3.6. React Hooks

For more complex function components, React Hooks are utilized; they "hook onto" React capabilities. React hook names begin with the word "use," per tradition. By default, state is absent from function component; however, the useState React hook can be used to keep state for the duration of the component. All hooks, including the useState one, are repeatable inside a single component.

Function components do not come with built-in life-cycle methods, but they can be added using the useEffect hook. By default, the useEffect hook will run a function for each time the component is re-rendered, but it may be modified to just run for specific modifications.

The useContext hook, which will be described in the Context API section, is another React hook. In addition to built-in hooks, custom hooks can be made, allowing functionality to be reused throughout components.

2.3.7. Virtual DOM

For all components in the application, React generates a new View based on immutable states and props; a change to either a state or a prop causes the View to be re-rendered. The Views are now predictable and testable as a result. The user experience is negatively impacted by the time-consuming process of re-rendering views by swapping out the DOM for a new version, which also causes scrolling position and input information to be lost. Using a Virtual DOM, React provides a solution for this [1].

By establishing a new Virtual DOM subtree when data in the application changes and comparing it to the previous one, the Virtual DOM re-renders Views that need to update in an affordable manner without disrupting other DOM nodes. React determines the minimal number of DOM alterations required to match the virtual DOM with the real DOM. Javascript is used to do DOM manipulations, which are then queued up and executed in batches to save time. React takes care of manipulating the DOM to get to the various states, while developers utilize JSX to define how components should render in various states.

2.3.8. Context API

The Context API is a built-in tool that allows data to be shared between React components without the need for props-drilling. The createContext method in the React library is used to initialize a Context, which can be used by several components in the component tree. A Provider present on the Context instance is used to add data to the Context and make it accessible to children components. The data for the Context is specified in the Provider, a component that has a single property named value and accepts variables, functions, or objects as values. The Context data can be retrieved from any child component using a Consumer that is also available on the Context instance by enclosing children components inside the Provider component. The application uses a variety of contexts for various purposes rather than being restricted to a single instance.

Context API's callback function inheritance allows a child component to change the state of a parent component. Context is used, according to the official documentation, to exchange information that is regarded as global for the tree of React components, such as whether a user is logged in, the application's theme, or the language preferences. Because they depend on information provided by context from another component, using context may reduce the reusability of components. UseContext functions as the Consumer and grants access to the data for a functional component encased in a Context Provider.

2.3.9. JSX

A React extension called JSX makes it simple for web designers to change the DOM using straightforward HTML-style code. Additionally, as all current web browsers are supported by React JS, JSX is interoperable with any browser platform.

Most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages [10]. The advantages of JSX include the fact that it makes writing templates for users who are familiar with HTML simpler and faster. Performance improves when code is compiled into JavaScript [9].

3 Methodology

3.1. Requirements Elicitation and Analysis

3.2. Design Concept

4 Project Implementation

References

[1] Johansson David. *Building maintainable web applications using React: An evaluation of architectural patterns conducted on Canvas LMS*. 2020.

[2] Cory Gackenheimer. "Introducing flux: An application architecture for react". In: *Introduction to React*. Springer, 2015, pp. 87–106.

[3] Google. *Service worker overview*. Online. 2022. URL: `https://developer.chrome.com/docs/workbox/service-worker-overview/`.

[4] Google. *What are extensions?* URL: `https://developer.chrome.com/docs/extensions/mv3/overview/`.

[5] Naimul Islam Naim. "ReactJS: An Open Source JavaScript Library for Front-end Development". In: (2017).

[6] Sheena S Iyengar and Mark R Lepper. "When choice is demotivating: Can one desire too much of a good thing?" In: *Journal of personality and social psychology* 79.6 (2000), p. 995.

[7] Lei Liu et al. "Chrome Extensions: Threat Analysis and Countermeasures." In: *NDSS*. 2012.

[8] Pratik Sharad Maratkar and Pratibha Adkar. "React JS - An Emerging Frontend JavaScript Library". In: *Iconic Research And Engineering Journal s* 4.12 (2021), pp. 99–102.

[9] Hong Duc Phan. "React framework: concept and implementation". In: (2020).

[10] React. *Introducing JSX*. Online. 2020. URL: `https://reactjs.org/docs/introducing-jsx.html`.

[11] Dolière Francis Somé. "Empoweb: empowering web applications with browser extensions". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2019, pp. 227–245.

[12] Miron Zuckerman et al. "On the importance of self-determination for intrinsically-motivated behavior". In: *Personality and social psychology bulletin* 4.3 (1978), pp. 443–446.

5 List of Abbreviations

API	Application Program Interface
COM	Component Object Model
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
JS	JavaScript
JSX	JavaScript XML
MVC	Model-View-Controller
Props	Properties
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kenntlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Datum, Ort, Unterschrift