Angular.js vs Vue.js

Cornelia Langsenlehner, Sandra Hartlauer, Michael Nußbaumer

BACHELORARBEIT

eingereicht am

Fachhochschul-Bachelorstudiengang

Mobile Computing

in Hagenberg

im Mai 2024

Advisor:

FH-Prof. Dr-Ing. Krösche Jens

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed copy is identical to the submitted electronic version.

Hagenberg, May 22, 2024

Cornelia Langsenlehner, Sandra Hartlauer, Michael Nußbaumer

Contents

Preface

This is version **2023/11/06** of the LaTeX document template for various theses at the School of Informatics, Communication and Media at the University of Applied Sciences Upper Austria in Hagenberg. We are pleased to learn that this document collection is meanwhile also used at various other institutions in Austria and abroad.

The document was initially created in response to requests from students after the 2000/01 academic year when an official LaTeX introductory course was offered in Hagenberg for the first time. The fundamental idea was to "simply" convert the already existing *Microsoft Word* template for diploma theses to LaTeX and possibly to add some unique features. This quickly turned out to be not very useful since LaTeX, especially concerning the handling of literature and graphics, requires a substantially different way of working. The result is— rewritten from scratch and much more extensive than the previous document—a manual for writing with LaTeX, supplemented with additional (meanwhile removed) hints for *Word* users. Technical details of the current version can be found in Appendix.

While this document was initially intended exclusively for the preparation of diploma theses, it now also covers *master theses*, *bachelor theses*, and *internship reports*. The differences between these documents have been deliberately kept small.

When creating this template, an attempt was made to work with the basic functionality of LaTeX and—as far as possible—to achieve this without additional packages. This was only partially successful; several supplementary "packages" are necessary, but only common extensions have been used. Of course, there is a large number of additional packages which can be helpful for further improvements and refinements. Everyone is encouraged to experiment with these as soon as they have the necessary self-confidence and sufficient time to experiment. Many details and tricks are not explicitly mentioned in this document but can be explored in the underlying source code at any time.

Numerous colleagues have provided valuable support through careful proofreading and constructive suggestions for improvement. We thank Heinz Dobler for consistently improving our "computer slang" and Elisabeth Mitterbauer for her proven "orthographic eye".

This document and the associated LaTeX classes have been available since November

2017 on CTAN[1] as package `hagenberg-thesis`,

https://ctan.org/pkg/hagenberg-thesis.

The current source code, as well as additional materials—such as a wiki with instructions for the integration of often requested functionalities and extensions—can be found at

https://github.com/Digital-Media/HagenbergThesis.[2]

Despite great efforts, a document like this always contains errors and shortcomings. Comments, suggestions, and helpful additions are welcome. Ideally, as comments or issues on GitHub.

By the way, here, in the preface (which is common in diploma and master theses but dispensable for bachelor's theses), you may briefly describe the genesis of the document. This is also the place for any acknowledgments (e.g., to the supervisor, the examiner, the family, the dog, etc.) as well as dedications and philosophical remarks. These should be balanced and limited to a maximum of two pages.

W. Burger (em.) and W. Hochleitner

University of Applied Sciences Upper Austria
Department of Digital Media, Hagenberg
https://www.fh-ooe.at/campus-hagenberg/

[1] Comprehensive TeX Archive Network

[2] https://github.com/Digital-Media/HagenbergThesis/blob/main/CHANGELOG.md contains a list of chronological changes (formerly included in the appendix of this document).

Abstract

Here goes an abstract of the work, with a maximum of 1 page. Unlike other chapters, the abstract is usually not divided into sections and subsections. Footnotes are also not used here.

By the way, abstracts are often included in literature databases with the author and title of the work. It is, therefore, essential to ensure that the information in the abstract is coherent and complete in itself (i.e., without other parts of the work). In particular, *no literature references* are typically used at this point (as is the case also in the *title* of the thesis and the German *Kurzfassung*)! If such is needed—for example, because the paper is a further development of a particular, earlier publication—then *full* references are necessary for the abstract itself, e.g., [Zobel J.: *Writing for Computer Science – The Art of Effective Communication*. Springer, Singapore, 1997].

It should also be noted that special characters or list items are usually lost when records are added to a database. The same applies, of course, to the German *Kurzfassung*.

In terms of content, the abstract should not be a list of the individual chapters (the introduction chapter is intended for this purpose). However, it should provide the reader with a concise summary of your thesis. Therefore, the structure used here is necessarily different from that used in the introduction.

Kurzfassung

Dies sollte eine maximal 1-seitige Zusammenfassung Ihrer Arbeit in deutscher Sprache sein.

The German “Kurzfassung” should contain the same content as the English abstract. Therefore, try to translate the abstract precisely but not word for word. When translating, remember that certain idioms from English have no counterpart in German or must be formulated differently. Also, word order in German is very different from English. Without knowledge of the German language, it is acceptable to resort to translators. Nevertheless, hiring a skillful person for proofreading is recommended even with the highest confidence in one’s German knowledge.

The correct translation for “diploma thesis” is *Diplomarbeit*, a “master thesis” is called *Masterarbeit*. For “bachelor’s thesis”, *Bachelorarbeit* is the appropriate translation.

By the way, for this section, the *language setting* in LaTeX should be switched from English to German to get the correct form of hyphenation. However, the correct quotation marks must be set manually.

Chapter 1

Introduction - M

1.1 Motivation

The motivation for this analysis lies in the increasing importance of web applications in the modern digital landscape. Businesses and services are increasingly migrating to online platforms, thereby increasing the demand for efficient, user-friendly, and powerful web applications. The platform independence of web applications enables developers to reach a broad user base without needing to create separate applications for each operating system or device. [0]

1.2 Challenges

1.2.1 Hardware Platforms

Developing web applications must support a variety of hardware platforms, including desktop computers, laptops, tablets, and mobile devices. The challenge is to ensure that applications perform consistently and efficiently across different devices. [0]

1.2.2 Frameworks

The selection and management of frameworks significantly influence the efficiency and scalability of a web application. Different frameworks offer various advantages and disadvantages in terms of performance, maintainability, and user-friendliness. [0]

1.2.3 Evaluation Methods

Evaluation methods are crucial for assessing the performance, security, and user-friendliness of web applications. This includes benchmarks, test scenarios, and other empirical methods to verify application quality. [0]

1.3 Goals

This analysis aims to provide a comprehensive overview of both frameworks and serve as a decision-making guide by offering valuable insights into the strengths and weak-

nesses of both technologies. These insights assist development teams in making informed decisions about the most suitable technologies for specific application requirements. Additionally, they help identify optimization opportunities to continuously enhance user experience and improve development efficiency.

Chapter 2

Theoretical Background - S

2.1 Web Frameworks

A framework is a program skeleton that serves as a foundation for software development. Frameworks already contain numerous functions in software development as proposed solutions for individual problems faced by various developers. This way, they do not have to start from scratch whenever they want to write new software.

A framework describes a collection of interacting classes and thus also fixes the design structure for software developed based on the framework. When a framework serves as the foundation for web applications, it is referred to as a web application framework [0].

2.2 Web Frameworks - Pros and Cons

Web frameworks offer many advantages, but they also come with significant disadvantages.

2.2.1 Advantages

The use of web frameworks aims to reduce time and cost in software development. The focus is on reusing code for basic functions such as database connections, templates, caching, and security, which are provided as pre-built modules. This allows development to concentrate on the specific code of the new application.

Since most web frameworks are available as open-source, there are generally no licensing costs. Frameworks also promote the creation of clean and maintainable code, as developers can rely on proven building blocks. These are regularly improved by the community and security vulnerabilities are quickly addressed [0].

2.2.2 Disadvantages

On the internet, there are numerous frameworks available for web development, which differ in their design principles and functionality. Depending on the project, a specific framework may be required, which can involve compromises.

Although frameworks are intended as general solutions, developers often do not use all the available functions, leading to unnecessary code, known as bloat. Another disadvantage is the dependency on the framework and its provider, as well as potential licensing restrictions. Problems can also arise if the development of the framework is discontinued.

Developers must also familiarize themselves with the structure and use of the framework, which requires time, although this is compensated by pre-built functions and code modules. Since the source code of many web frameworks is publicly accessible, anyone can view it, leading to potential security risks, especially if enterprise applications are based on public code [0].

2.3 Introduction to JavaScript Frameworks

JavaScript is a versatile programming language that is particularly suited for work in web browsers. Originally developed by Netscape, it has since become one of the most widely used programming languages on the web. JavaScript enables developers to create dynamic and interactive content on web pages by allowing manipulation of the Document Object Model (DOM).

However, working directly with the DOM can be challenging for many programmers. This is where JavaScript frameworks and libraries come into play: they provide developers with tools to simplify these and other aspects of programming.

JavaScript frameworks are particularly suitable for developing complex web applications. They provide a structured approach and pre-built components that make development more efficient. Once developers become familiar with the concepts and guidelines of a particular framework, they can work very effectively with it [0].

2.4 Comparison criteria - M

When evaluating web development frameworks, metrics play a crucial role as they enable an objective analysis of performance and efficiency. By comparing load times, responsiveness, and resource utilization, frameworks like Angular and Vue can be assessed in various scenarios to determine their performance.

2.4.1 Performance

Angular provides robust performance and scalability for complex applications. In contrast, Vue stands out for its fast loading times and low overhead, making it particularly attractive for smaller projects. [0]

2.4.2 Developer-Friendliness

Angular offers a robust structure and comprehensive features, which come with a steeper learning curve but are supported by extensive documentation and a strong community. Vue, on the other hand, is characterized by its easy integration and intuitive syntax, facilitating quicker onboarding and flexibility for smaller projects. [0]

2.4.3 Community Support

Active and engaged community support is crucial for framework development and bug fixing. Referring to the study by Jelica Cincović and Marija Punt titled "Comparison: Angular vs. React vs. Vue," which examined community support through GitHub repository comparisons, Figure illustrates the following:

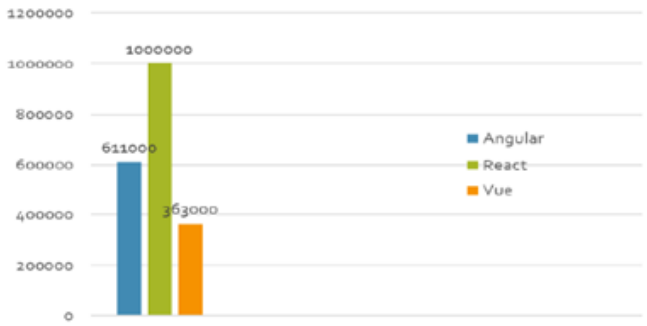

Figure 2.1: Number of GitHub repositories by frameworks

The graph indicates that React enjoys the largest community support, followed by Angular in second place and Vue in third. [0]

2.5 Brief Description of Angular.js

AngularJS, developed by Google, is a JavaScript framework for web development that emphasizes structure and quality. It was introduced in 2010 and was the first framework suitable for large enterprise applications, emphasizing clear architecture, high testability, and isolated components. Techniques like Dependency Injection enable efficient and maintainable software development based on JavaScript.

Angular, introduced in 2016 as a successor to AngularJS, is a complete rewrite and uses TypeScript as its foundation. This allows for better scalability and performance. Angular offers an extensive development environment with features such as two-way data binding, a powerful Command Line Interface (CLI), and a well-defined architecture. It has evolved from a pure framework to a comprehensive platform for developing web applications [0].

2.6 Brief Description of Vue.js

Vue.js, pronounced /vju:/, like "view", is a JavaScript framework for designing user interfaces. It was developed by Evan You in 2014 and has quickly become one of the most popular frameworks. Vue.js is based on standard HTML, CSS, and JavaScript and offers a declarative, component-based programming model. This model allows for efficient development of user interfaces of any complexity. Key features of Vue.js include the virtual DOM, reactive data binding, and easy integration with other projects and libraries.

Vue.js is flexible and lightweight, suitable for both small projects and large applications. It allows developers to incrementally adopt it into existing projects and add additional features as needed [0].

2.7 Differences Between Angular.js and Vue.js

Angular and Vue.js are both JavaScript frameworks that can be used to develop modern web applications. A key difference is that Angular uses an MVC (Model-View-Controller) architecture, while Vue.js is a more progressive framework.

Angular provides a robust and well-established development environment with features such as efficient two-way data binding, a comprehensive CLI, and a well-defined architecture. It is particularly suitable for large and complex applications due to its strict structure and extensive tools.

On the other hand, Vue.js is more flexible and lightweight. It uses a virtual DOM, which improves performance, and offers simple ways for CSS transitions and property computation. Vue.js is easier to learn and integrate, making it a good choice for smaller projects or projects that require rapid development.

Both frameworks have their advantages and disadvantages, and the choice between them depends on the specific requirements and preferences of a project. While Angular offers a comprehensive solution for complex applications, Vue.js excels in simplicity and flexibility [0].

Chapter 3

Case Study - C

3.1 Introduction to the Case Study

In this case study we will be comparing two versions of a web application called "ProTrack" that has been developed using two different JavaScript frameworks, Angular.js and Vue.js. ProTrack is a project management tool that allows users to track, manage and edit project work. The application provides a range of features that enhance administrative efficiency and project management capabilities.

3.1.1 Objective of the Case Study

The objective of this case study is to compare Angular and Vue in the context of project management applications. The focus will be on three primary functionalities: user registration, project creation, and time tracking. By developing and analyzing these applications, we aim to highlight the strengths and weaknesses of each framework in practical use.

3.1.2 Application Description

User Registration and Login

Implementation in Angular: In Angular, user registration and login were implemented using Angular Forms and an authentication service. Angular's template-driven forms were employed to handle user input, and validation was managed through built-in validators. Authentication was handled using JWT (JSON Web Tokens) to ensure secure access.

Implementation in Vue: In Vue, user registration and login were implemented using Vue Forms and Vuex for state management. The form components handled user input and validation within the components themselves. Vuex managed the user state, including authentication tokens.

Project Management

Project Creation:
Angular: Project creation in Angular was achieved using Reactive Forms and services. Reactive Forms allowed for complex form validation and dynamic form control. The data handling was managed through Angular services, ensuring a clean separation of concerns.
Vue: In Vue, project creation utilized Vue Forms and Vuex. The form logic was encapsulated within the components, while Vuex managed the state of the project data, ensuring consistency across the application.

Start and Stop Functionality:
Angular: The start and stop functionality for time tracking in Angular was implemented using a timer service and RxJS for managing time intervals. This approach allowed for reactive programming, enabling real-time updates and precise time tracking.
Vue: In Vue, the time tracking functionality was built using a timer component and Vuex for state management. The component handled the timer logic, while Vuex ensured that the state was updated accurately and consistently.

3.2 Technical Implementation

Application Architecture

Angular: The Angular application followed a component-based architecture with modules and services. This hierarchical structure ensured maintainability and scalability. Strong typing with TypeScript added an additional layer of robustness.
Vue: The Vue application also utilized a component-based architecture with Vuex for centralized state management. Vue's flexible structure allowed for rapid development and easy integration of new features.

State Management

Angular: State management in Angular was handled using services and BehaviorSubjects from RxJS. This approach facilitated reactive updates and a clear separation of business logic.
Vue: Vuex store was used for state management in Vue, providing a single state tree and using mutations for state changes. This made the state predictable and easier to debug.

Routing

Angular: The Angular Router was used for navigation, with lazy loading to improve performance. Route guards were implemented to secure routes and manage user access.
Vue: Vue Router was employed in the Vue application, with dynamic imports to optimize loading times. Navigation guards ensured that only authenticated users could access certain routes.

3.3 Performance and efficiency

Loading Times and Responsiveness

Angular: Performance metrics indicated that Angular had a slightly longer initial load time due to its larger bundle size. However, once loaded, the application was highly responsive and efficient.
Vue: Vue exhibited faster initial loading times and smaller bundle sizes, contributing to a smoother user experience from the outset. Its lightweight nature made it particularly suitable for rapid development cycles.

3.4 Challenges and Solutions

Common Challenges

State Management and Routing: Both frameworks faced challenges in synchronizing state and handling complex navigation scenarios. Angular's use of RxJS and Vue's Vuex provided robust solutions to these issues.

Framework-Specific Challenges

Angular: The main challenge with Angular was its complexity and boilerplate code, which required a steeper learning curve and more initial setup time.
Vue: Vue's flexibility sometimes led to inconsistencies, especially in larger applications. Ensuring code quality and maintaining state consistency were key challenges.

3.5 Comparison and analysis

Advantages and Disadvantages

Angular: Angular's strengths lie in its stability and comprehensive ecosystem, making it ideal for large-scale enterprise applications. However, its verbose syntax and higher initial complexity can be drawbacks.
Vue: Vue's simplicity and flexibility make it suitable for rapid development and smaller projects. Its main weakness is a less extensive ecosystem compared to Angular.

3.6 Conclusion from the Case Study

Summary:

Angular and Vue each have unique strengths and are suitable for different types of projects. Angular excels in large-scale, complex applications, while Vue is ideal for smaller, agile developments.

Recommendations:

Based on this case study, Angular is recommended for enterprise-level applications requiring robust architecture and extensive features. Vue is recommended for projects needing quick prototyping and flexibility.

Future Outlook:

Future research could explore the evolving best practices in both frameworks and their impact on development efficiency and performance.

Chapter 4

Results - S

4.1 Performance

4.2 Developer-friendliness

4.3 Community support

4.4 Ecosystem

Chapter 5

Conclusion and recommendations - M

5.1 Conclusion

5.2 Recommendations

References

Literature

[0] Angular.de. "Was sind Angular und AngularJS?" (2017). Abgerufen am 31.05.2024. URL: https://angular.de/artikel/was-ist-angular/.

[0] Jelica Cincovi'c and Marija Punt. "Comparison: Angular vs. React vs. Vue. Which framework is the best choice". *Universidad de Belgrade* (2020).

[0] Ionos. "Die beliebtesten JavaScript-Frameworks und -Bibliotheken" (2023). Abgerufen am 31.05.2024. URL: https://www.ionos.at/digitalguide/websites/web-entwicklung/beliebte-javascript-frameworks-und-bibliotheken/.

[0] Ionos. "Webframeworks – Überblick und Klassifizierung" (2022). Abgerufen am 31.05.2024. URL: https://www.ionos.de/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/.

[0] Kinsta. "Angular vs. Vue: Ein Kopf-an-Kopf-Vergleich" (2023). Abgerufen am 31.05.2024. URL: https://kinsta.com/de/blog/angular-vs-vue/#angular-vs-vue-hnlichkeiten-und-gemeinsamkeiten.

[0] Xiang Mao and Jiannong Xin. "Developing Cross-platform Mobile and Web Apps". *CIGR Proceedings* (2014).

[0] United Nations Conference on Trade and Development. "Digital Economy Report 2019" (2019). Abgerufen am 31.05.2024. URL: https://unctad.org/publication/digital-economy-report-2019.

[0] Dhruv Verma. "A comparison of web framework efficiency: performance and network analysis of modern web frameworks" (2022).

[0] Vue. *Introduction to Vue.js.* https://vuejs.org/v2/guide/. Accessed: 2024-06-24. 2024.

[0] Vue.js. "Einführung" (). Abgerufen am 31.05.2024. URL: https://vuejs.org/guide/introduction.html.