

Analysis of the use of Angular and Svelte products in mobile web applications

Analiza wykorzystania produktów Angular oraz Svelte w mobilnych aplikacjach internetowych

Michał Nurzyński*, Marcin Badurowicz

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The aim of this article is to analyze the use of Angular and Svelte tools in the development of a web application. The analysis focused on a custom application that allows measuring component rendering times and application build parameters. The research scenarios also considered the popularity of the frameworks. The results indicate higher performance for a small application created with Svelte, while Angular enjoys greater popularity.

Keywords: Angular; Svelte; analysis of usage; web application

Streszczenie

Celem niniejszego artykułu jest dokonanie analizy wykorzystania narzędzi Angular oraz Svelte w oparciu o budowę aplikacji webowej. Pod analizę została podjęta autorska aplikacja, pozwalająca na mierzenie czasów renderowania komponentów oraz parametrów budowania aplikacji. Scenariusze badawcze dodatkowo uwzględniały temat popularności frameworków. Uzyskane wyniki wskazują na wyższą wydajność małej aplikacji utworzonej za pomocą Svelte, zaś większą popularnością cieszy się Angular.

Słowa kluczowe: Angular; Svelte; analiza użycia; aplikacja webowa

*Corresponding author

Email address: michal.nurzynski@pollub.edu.pl (M. Nurzyński)

Published under Creative Common License (CC BY 4.0 Int.)

1. Introduction

The dynamic development of web technologies has created a demand for tools that allow users to efficiently and more easily use web applications. Libraries, frameworks, and compilers enabling the creation of single-page web applications based on JavaScript have found widespread use. The abundance of available solutions presents a challenge in selecting the appropriate tool for the needs of the project being developed. Individual frameworks differ in how they perform operations on the DOM (Document Object Model), which allows for the dynamic modification of document structure and displayed elements [1].

Angular is a framework for building web applications, based on TypeScript. It uses the MVC (Model-View-Controller) model, which separates the view and model layers through a controller layer that transfers data between the other layers. The view layer handles the user interface and defines visual components, while the model layer encapsulates the data logic and represents it within the application. A virtual DOM is not used for rendering the interface state, only direct manipulation of the DOM is supported. Angular uses factory functions to convert existing components into classes, which are then registered and displayed in the DOM [2, 3].

Svelte eliminates the need for using virtual DOMs in real-time by pre-rendering the user interface. This makes Svelte described as a compiler. Additionally, unlike Angular, which uses the MVC pattern, Svelte removes the dedicated controller layer, applying a reactive system that

updates the user interface in response to data changes. This results in lower requirements for running the application, as well as simpler and easier-to-maintain code. Svelte syntax resembles plain JavaScript with a component-based structure, and the compiled code is contained in a single file that is interpreted at build time [4-6].

2. Aim of the work and research hypotheses

This article presents a comparison and performance evaluation of web applications created using Angular and Svelte tools on mobile devices. Both frameworks are written in TypeScript and have a similar interaction with the DOM [2]. For the purpose of conducting the analysis, a proprietary application was developed in both frameworks. The criteria by which the tools will be tested in the study include usability analysis, performance under limited internet conditions, overall support evaluation, popularity and problem-solving capabilities.

The article also aims to address several research hypotheses:

- H1: Svelte is more efficient in building smaller web applications and in conditions of limited network connection speed.
- H2: An application built in Svelte consumes fewer resources than an application built in Angular.
- H3: Angular has broader community support and a richer ecosystem of tools and libraries compared to Svelte.

3. Literature review

Comparing different frameworks has been the focus of many articles and theses, while their dynamic development and continuous updates mean that research results can change over time. A comparison of frameworks provides insight into their performance, advantages over other solutions, and helps determine which tool is best suited for a given project.

Few studies have directly compared Angular and Svelte. The article [5] discusses a comparison of these two frameworks based on the results obtained from applications implemented in each framework. The applications were tested in terms of startup time, recompilation after changes, the size of the final package after compiling the application, and the size of the dependencies required for the application to run. The applications contained the same logic, with differences in implementation depending on the framework's functionality. Performance was compared using the Lighthouse tool, which analyzes the application in several categories. The results in the tested categories favored Svelte. The applications did not fully utilize the capabilities offered by the frameworks, and the conclusion was that Angular is a good solution for large and medium-sized applications, while Svelte performs excellently in small applications, but in larger ones may require a longer implementation time.

Many more articles focus on the comparative analysis of several frameworks. Article [7] aims to compare the multifaceted requirements of web applications, outlining the specifications of the most popular tools and libraries on the market. As a conclusion, no single, specific framework was pointed out, but the strengths and weaknesses of each tool were discussed. Another article [8] compares test applications, their design, library support, and other application parameters. The launched applications used the same package manager (Node Package Manager), and the projects included a start page. The amount of support in the form of dedicated libraries, extensions, and plugins for the more popular frameworks, such as React and Angular, was also mentioned, while Svelte and Vue have fewer of these. Other articles [1, 2, 9] also contain practical benchmark statistics, full of calculations and charts.

The article [10] considers the analysis of comparing frameworks for building highly scalable applications. The advantages as well as disadvantages of both frameworks are highlighted, such as the lack of a virtual DOM. The main concepts and functionalities of Angular are also mentioned in the article [11]. The mechanisms that are offered for application design are discussed here, i.e. data binding, use of components and directives or routing. The disadvantages presented are increased performance overhead and a high entry threshold. In the case of Svelte, on the other hand, the article [4] points to features in favor of this framework. In addition to its reactive approach, which offsets the need for state management and small requirements, the advantages include a simple and easy-to-understand syntax. However, Svelte is quite unpopular, and consequently the ecosystem of tools is also smaller.

An important aspect discussed in articles [7, 12] is community support. Communities aim to promote learning or software development. The popularity of a framework can be indicated by the number of projects created or changes submitted on the GitHub platform or the number of issues and queries created on Stack Overflow. Article [2] also contains information regarding the learning curve of the frameworks, addressing how difficult a given framework is to learn or use in development work. It also touches on the availability of documentation, descriptions of the framework's functionality, and the number of languages into which it has been translated.

4. Results


The analysis of Angular and Svelte products was conducted using two applications written in the Svelte framework (version 5.0.0) [13] and Angular (version 18) [1, 14]. The main goal of the applications was to present results related to component generation times, and additionally, the build time and the compiled bundle size were compared. The application consists of a field for entering the number of components and buttons for generating and clearing the view of components (Figure 1). Additionally, it includes a field that displays the measured time.

Number of components:

Generate components Remove components

Generation time: 0.00 ms

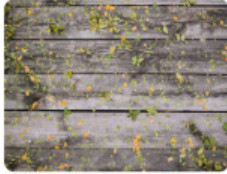
Rendering time: 82.80 ms



Ab quasi et nihil qui ex.

In dolorum quod id ipsam nesciunt cum. Quia eos incidunt sed perferendis eum aperiam. Nulla et praesentium et enim tenetur. Harum ipsa quidem doloremque aut autem officiis.

by Irving Schmeler East Natalia, Serbia



At eos et sit molestias.

Sit est provident eos molestias. Illum natus placeat magni laudantium voluptatum expedita. Ullam rerum totam dolorum quis occaecati sint.

by Simeon Keebler Lonieborough, Syrian Arab Republic

Figure 1: View of the tested application.

Environment specifications:

- Processor: Intel Core i5-9300H, 2.40GHz
- RAM: 16 GB
- Disk: SSD 256GB, SK HYNIX HFM256GDJTNG-8310A
- Operating System: Windows 10 Home 22H2
- Browser: Google Chrome, version 131.0.6778.69

For the research methodology, test scenarios were developed to compare the tested products. The tests were repeated 30 times to obtain a larger sample of times.

- Analysis of Application Build Times and Compiled File Sizes.

This scenario examines the times required to build the applications, determining which application is built faster. The size of the file is crucial in deployment scenarios where the compiled file is required to launch the webpage.

- Generating and Removing Components Using Buttons Available on the Page in Various Network Modes (5G, LTE, Slow 4G, 3G).

This scenario assesses the functionality of generating components in the deployed application. The generated components consist of simple elements - text and images fetched via API from fakerapi.it [15]. The data is pre-fetched during the first page load, so the time to fetch data from the API is not included in the component generation time.

- Analysis of Framework Popularity Comparison.

The third scenario focused on analyzing the popularity of technologies and community support. Some platforms conduct surveys that allow the determination of what percentage of users utilize a particular technology and how many wish to continue working with it. StackOverflow offers an annual overview of many technologies, categorized by professional developers and those learning to program.

A similar analysis focusing primarily on frontend frameworks can be found on State of JavaScript. In that case, data can be segmented using multiple filters to provide more detailed data representations.

5. Results

5.1. Analysis of Application Build Times and Compiled File Size Comparison

Significantly lower compilation times for the production version of the tested application and the development version occurred for the application built in Svelte, as shown in Table 1. Similar observations apply to dependency installation time, where Svelte took an average of 6.6 times less time than Angular. As for the size of the built package, Angular has a larger size than Svelte by more than four times.

Table 1: Results of application build tests

Activity	Angular	Svelte
Dependency installation time [s]	115.04	17.4
Full production build time [s]	20.95	5.58
Dev server startup time [s]	3.85	2.06
Final production package size [KB]	302	72.4
Final dependency package size [MB]	249	52.5

5.2. Analysis of Component Generation in Different Network Modes (5G, LTE, Slow 4G, 3G)

The average component generation time for a small application with a relatively small number of components is about half as fast for Svelte (Figure 2). For a larger number of components, the difference in measurements between applications decreases, but Angular still generates components more slowly. Comparing the graphs in Figure 3 and Figure 4, the difference between the obtained measurements of two applications decreases from approx. 2 times for 1,000 components to roughly 1.6 times for 10,000 components.

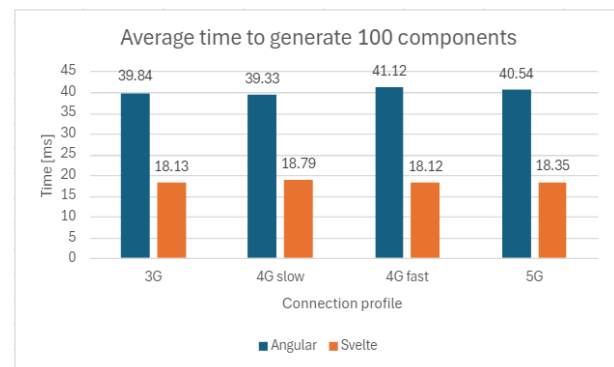


Figure 2: Average time to generate 100 components.

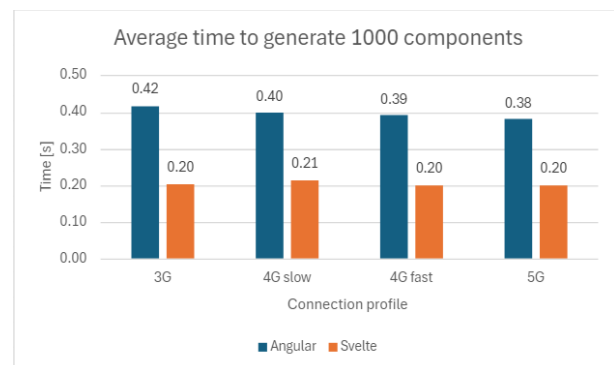


Figure 3: Average time to generate 1000 components.

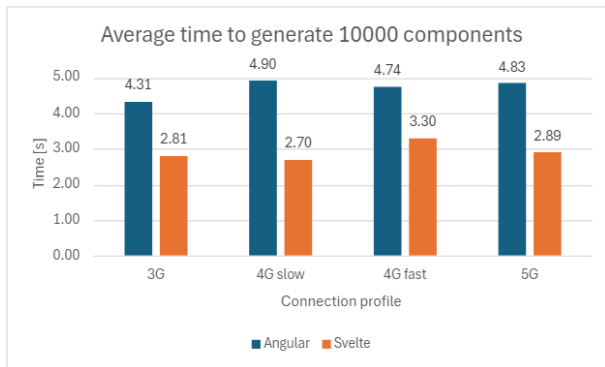


Figure 4: Average time to generate 10000 components.

5.3. Analysis of Framework Popularity Comparison

The popularity of frameworks depends on many factors, such as performance, but it is also influenced by industry trends and the quality of documentation. The third scenario focused on analyzing the popularity of technologies and community support. Some platforms conduct surveys that allow the determination of what percentage of users utilize a particular technology and how many wish to continue working with it. StackOverflow [16, 17] offers an annual overview of many technologies, categorized by professional developers and those learning to program. Table 2 presents the results of changes in framework popularity over the years. It can be noted that Svelte gains popularity each year among those learning to program, while interest in Angular remains relatively stable. Specifically, this fact is illustrated in Figure 5, where there is a chart showing the popularity of frameworks. The blue points represent the percentage of people who want to work with a given framework in the following year, while the red ones indicate respondents who worked with it this year and want to continue working with it in the next year. Svelte dominates in terms of the latter condition, while Angular shows results below average, suggesting that it is not highly sought after.

Table 2: Popularity results of Angular and Svelte

Group	Year	Angular	Svelte
All respondents	2022 (58743 respondents)	20.39%	4.58%
	2023 (84152 respondents)	17.46%	6.62%
	2024 (48503 respondents)	17.10%	6.5%
Professional developers	2022 (45297 respondents)	23.06%	4.11%
	2023 (65333 respondents)	19.89%	6.01%
	2024 (38132 respondents)	19.4%	5.9%
People learning to program	2022 (4932 respondents)	10%	4.35%
	2023 (4391 respondents)	7.15%	6.42%
	2024 (5010 respondents)	7.7%	9.6%

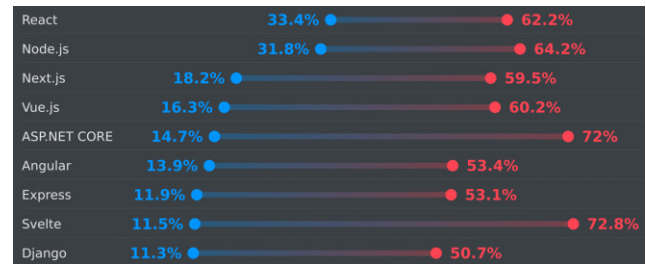


Figure 5: Framework popularity chart [16].

Very similar results are presented by the State of JavaScript website [18]. Figure 6 shows the popularity of frameworks based on data gathered by this site from 23,532 respondents. A decrease in Angular usage is also visible, while Svelte is becoming more popular in web application usage. Figure 7 more emphatically illustrates the scale of interest in both frameworks. Angular records the weakest results in recent years, while Svelte has been at the top of the list since it first appeared in the ranking.

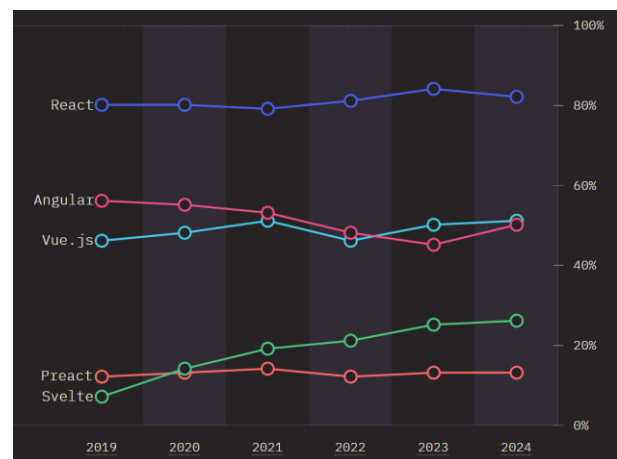


Figure 6: Front-End framework popularity [18].

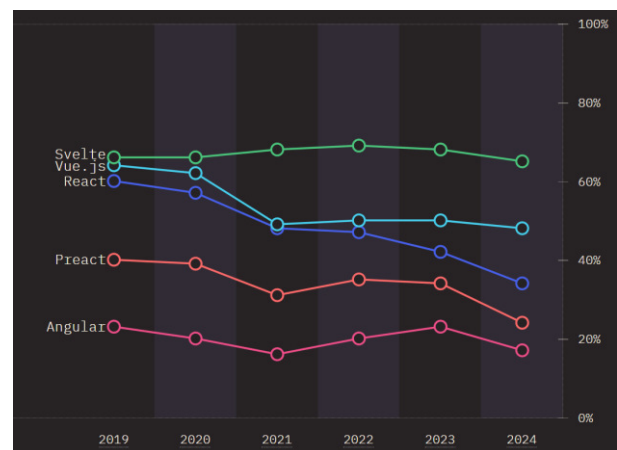


Figure 7: Interest in front-end frameworks [18].

6. Discussion

The research conducted in this article makes it possible to ascertain key aspects against the hypotheses. Based on the results obtained, it is possible to determine the truth of the hypotheses.

The analysis of the application in terms of component generation under different network conditions states the truth of hypothesis H1, where the greater efficiency of the Svelte application over the efficiency of the Angular application is taken into account (Figures 2-4). For generating 100 components, there are no significant differences in the ratio between the applications, and changing network modes, for each measurement Svelte generates components about 2 times faster than Angular. For 1,000 components, Svelte's performance drops to about 1.9 times faster component generation, while for 10,000 components the difference is most significant, ranging between 1.4 times for 4G Fast mode and 1.8 times for 4G Slow mode in favor of the Svelte tool.

The results obtained during the application building process, the results of which can be found in Table 1, also confirm the truth of hypothesis H2, which indicates an application built in Svelte as less resource-intensive than one created in Angular. The time to build the production and developer versions of the Svelte app was 73% and 46% shorter, respectively, relative to Angular. The sizes of the built versions of the production and developer versions of Angular applications were larger by 317% and 374%, respectively, relative to the size of the built Svelte application on disk. The results of the H2 hypothesis analysis are in line with the results reported in the articles [5, 9, 12], where Svelte also proved to be a framework that consumes fewer resources.

Analysis in terms of the last hypothesis, H3, also allows us to conclude that the given hypothesis is correct. Table 2, as well as Figures 5-7, strongly indicate that Angular is more popular among the categories studied. However, it should be noted the steadily increasing popularity of Svelte among programming learners.

7. Conclusions

The research that was carried out within the scope of the work, along with the established research methodology, allows us to conclude that the research hypotheses set are true and the results coincide with those observed in other works of results.

Svelte proves to be much better in terms of performance than Angular in the context of small applications, even when network conditions change. In addition, Svelte uses fewer resources and much faster startup and build times for applications. However, the community built around Angular is definitely wider, with more tools available to extend the capabilities of the framework.

The choice of an appropriate framework depends on the set requirements for the application to be developed, its size, technical limitations and scalability. Performance analysis and analysis of the framework market, allow developers to better select tools for the marked requirements and improve the software development process.

Future work could expand on analyzing the long term maintenance and scalability of applications. An additional aspect might include further analyses of the popularity and usage of frameworks, particularly in the context of frequent updates and changing trends.

References

- [1] R. Ollila, N. Mäkitalo, T. Mikkonen, Modern web frameworks: A comparison of rendering performance, *Journal of Web Engineering* 21(3) (2022) 789–183, <https://doi.org/10.13052/jwe1540-9589.21311>.
- [2] M. Levlin, DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte, Master's Thesis, Åbo Akademi University, Turku, 2020.
- [3] K. A. Stokke, M. Barash, J. Järvi, The Ultimate GUI Framework: Are We There Yet?, In *Eelco Visser Commemorative Symposium (EVCS)* (2023) 25:1–25:9, <https://doi.org/10.4230/OASIS.EVCS.2023.25>.
- [4] S. Bhardwaz, R. Godha, Svelte.js: The most loved framework today, In *2023 2nd International Conference for Innovation in Technology (INOCON)* (2023) 1–7, <https://doi.org/10.1109/INOCON57975.2023.10101104>.
- [5] T. D. Tripon, G. A. Gabor, E. V. Moisi, Angular and Svelte frameworks: a comparative analysis, In *2021 16th International Conference on Engineering of Modern Electric Systems (EMES)* (2021) 1–4, <https://doi.org/10.1109/EMES52337.2021.9484119>.
- [6] J. Žižka, Modernizing AEON web frontend, Bachelor's Thesis, Masaryk University, Brno, 2024.
- [7] A. Naik, The Front-End Dilemma: How to Choose the Perfect Technology for your Application, *Journal of Computer Science and Technology Studies* 6(1) (2024) 211–216, <https://doi.org/10.32996/jcsts.2024.6.1.24>.
- [8] M. Bauer, Hello Framework! A heuristic method for choosing front-end JavaScript frameworks, Bachelor's thesis, Uppsala University, Uppsala, 2021.
- [9] T. Marx-Raacz Von Hidvég, Are the frameworks good enough?: A study of performance implications of JavaScript framework choice through load-and stress-testing Angular, Vue, React and Svelte, Bachelor's thesis, Linnaeus University, Växjö & Kalmar, 2022.
- [10] S. Denys, A comparative analysis of component-based architectures in web design for scalable applications, *Холодная наука* 8 (2024) 24–31.
- [11] S. Khati Chhetri, Comparative Study of Front-end Frameworks: React and Angular, Bachelor's Thesis, Oulu University of Applied Sciences, Oulu, 2024.
- [12] S. A. Jameel, Comparison of Modern JavaScript Web Frameworks, Bachelor's thesis, Tomas Bata University in Zlín, Zlín, 2022.
- [13] Svelte documentation, <https://svelte.dev/>, [26.10.2024].
- [14] DOM APIs in Angular, <https://angular.dev/guide/components/dom-apis>, [26.10.2024].
- [15] Fake API instance, <https://fakerapi.it/>, [26.10.2024].
- [16] StackOverflow 2024 survey, <https://survey.stackoverflow.co/2024/technology#most-popular-technologies-webframe-prof>, [26.10.2024].
- [17] StackOverflow 2023 survey, <https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe-learn>, [26.10.2024].
- [18] State of JS 2024 survey, <https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/>, [27.02.2025].