Jurnal Mantik, 6 (3) (2021), ISSN 2685-4236 (Online)

Published by:Institute of Computer Science (IOCS)

Jurnal Mantik

Journal homepage: www.iocscience.org/ejournal/index.php/mantik

Comparative Analysis Study of Front-End JavaScript Frameworks Performance Using Lighthouse Tool

Mangapul Siahaan[1], Vincent Octarian Vianto[2],
[1,2]Computer Science Faculty ,Universitas Internasional Batam, Indonesia

ARTICLE INFO

Article history:

Received Oct 07, 2022
Revised Oct 14, 2022
Accepted Nov 04, 2022

Keywords:

Framework
JavaScript
Lighthouse
Performance
Web

ABSTRACT

JavaScript is one of the most widely known and widely used programs. The most common determining factor when choosing the proper framework for a job. Through a comparative analysis of several JavaScript frameworks like React, Angular, Vue, Svelte, and SolidJS, this research aimed at understanding how web applications perform. As part of this research and to compare JavaScript Framework, system development used SDLC Waterfall for each framework to build Weather App. According to the performance testing results, The FCP value of the five frameworks has a score of fewer than 1,8 seconds. The SI value of the five frameworks is less than 3.4 seconds. The LCP value of the five frameworks is less than 4 seconds. The TTI value of the five frameworks is less than 3.8 seconds The TTI value of the five frameworks is less than 3.8 seconds. Based on the results of the analysis and testing conducted in this research using Google Lighthouse, Vue delivers the best performance when implementing the Weather App.

Corresponding Author:

Vincent Octarian Vianto,
Computer Science Faculty,
Universitas Internasional Batam
Baloi-Sei Ladi, Jl. Gajah Mada, Tiban Indah, Kec. Sekupang, Kota Batam, Kepulauan Riau 29426, Indonesia
Email: 1931098.vincent@uib.edu

1. INTRODUCTION

Web applications (Web Apps) are becoming increasingly important with the increasing number of internet users and rich internet applications such as social networking, e-commerce, online banking, webmail, polls, and chat (Sherin et al., 2021). It can be proven that there are now 1.9 billion websites in the world and the number will continue to grow (Tjhoernandes et al., 2022). Web applications are also called web browsers and are often used to access static information on the internet. In web application development, JavaScript is one of the most widely known and widely used programs. JavaScript is usually implemented with HTML and CSS3 on the client side to create attractive, interactive, and creative web designs (Levlin, 2021). The JavaScript language itself is still in its infancy, but its framework, the JavaScript Framework (JSF), has played a major role in web application development over the years.

According to (Persson, 2020), the average web page size has increased from year to year too. With the increasing number of web sizes and the number of internet users, the performance metrics have a greater impact on energy usage on the client side (Malavolta et al., 2020). So, a challenge that web developers face is when choosing the right programming language or framework to fulfill their job. The most common determining factor when choosing the right framework for the job is usually centered on the familiarity of the developer, which according to him is not the right basis for choosing a framework because it tends to be subjective (Mariano, 2017). For web developers, it is very important to choose a framework that best suits their needs and that provides code with high-quality and performance.

Based on research (Yusron & Wibowo, 2020), There is a constant debate between Angular and React among JavaScript developers, with each technology tending to be biased towards one side. Angular and React are both popular technologies for developing interactive one-page applications. After conducting the research obtain the result found that websites developed with Angular had smaller code sizes than those developed with React. However, the performance and response time of websites built using React is better when accessed on browsers than Angular.

The purpose of the research (Patra, 2022) was to choose a framework by comparing React and Vue based on different criteria, such as performance and speed tests in the same web application in both JavaScript Frameworks by analyzing data binding, volume, performance, and technical support of framework from secondary statistic and analyzing the popularity and usability of users around the world. Analysis shows that Vue.js is suitable for lightweight applications, suited to existing applications where speed is an important consideration. When it comes to performance, React is a handy framework for responsive apps, apps that require scalability and which need to be SEO-friendly.

There was also research about comparing Angular with Vue for the development of games. The comparative criteria are as follows: time of data exchanging with the server and rendering of various application components, memory consumption during refreshing the current game information and restoring the user to the current game, browser load level, and size of the final application files. The test results show that the Vue.js framework is more efficient (Baida et al., 2020).

There are also other studies has a goal to check if the Svelte-based client part of a web application is more effective than the standard Angular approach and presents a comparison of page components' rendering times based on two test applications prepared in both frameworks. For the performance tests, scenarios were prepared in which the times of adding and removing a different number of page components were examined. Application tests were performed using the Selenium Webdriver package. The research results clearly showed that the Svelte is more efficient than Angular (Białecki & Pańczyk, 2021).

(Ollila et al., 2022) describe the rendering strategies used in the frameworks Angular, React, Vue, Svelte, and Blazor, which represent some of the most influential and widely used modern web frameworks. they find significant differences in the scaling of costs in their rendering strategies with potentially equally significant practical performance implications. To verify these differences, we implement several benchmarks that measure the scaling of rendering costs as an application grows in complexity. The results of their research, performance differences between frameworks can range up to several orders of magnitude when performing the same tasks. Furthermore, we find that the relative performance of a rendering strategy can be effectively estimated based on factors affecting the input sizes of render loops. The best-performing rendering strategies are found to minimize input sizes using techniques such as compile-time optimization and reactive programming models.

(Carniato, 2020) conducted SolidJS research in which SolidJS is a new JavaScript framework. By building the Realworld app and comparing other frameworks with the lighthouse tool, He found that websites developed with SolidJS perform better. This is across the board from the lighthouse score which is quite high, and the page load time is quite short.

Therefore, this research was made to provide an understanding of web performance by conducting a comparative analysis of several JavaScript Frameworks, namely React, Angular, Vue, Svelte, and SolidJs with the Lighthouse Tool as a performance measurement tool. One of the benefits obtained from this research is to provide input to website application developers so that they are considered as well as learning about JSF, especially React, Angular, Vue, Svelte, and SolidJs.

2. RESEARCH METHOD

This research consists of five phases. First, the selection of the JavaScript Framework will be the object of study. Second, determining the indicators for measuring website performance metrics. Third, the design and development of a website-based weather information application (Weather App) using several JavaScript Frameworks that were selected as the object of study. Fourth, the measurement of each selected JavaScript Framework based on performance metric indicators. Fifth, recording measurement results. Sixth, analysis of measurement results from the JavaScript Framework that has been tested. Finally, give a conclusion on which is best JavaScript Framework from the best in terms of performance. An overview of the research stages can be seen in Figure 1.

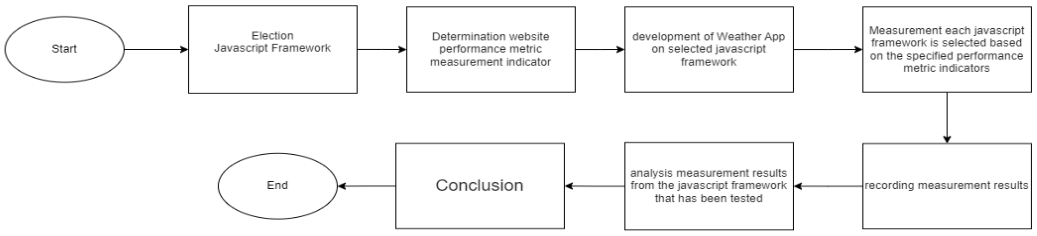

Start
Election Javascript Framework
Determination website performance metric measurement indicator
development of Weather App on selected javascript framework
Measurement each javascript framework is selected based on the specified performance metric indicators
recording measurement results
analysis measurement results from the javascript framework that has been tested
Conclusion
End

Figure 1. Research stages

To analyze this research and to compare several JavaScript Frameworks, a system development method is carried out where each framework that is implemented for making Weather App will have the same specifications. The method used for the development of the Weather App for each JavaScript Framework is the SDLC Waterfall method. The SDLC Waterfall method is a method that has the characteristic that the work on each phase must be done first before proceeding to the next phase (Nugraha et al., 2018).

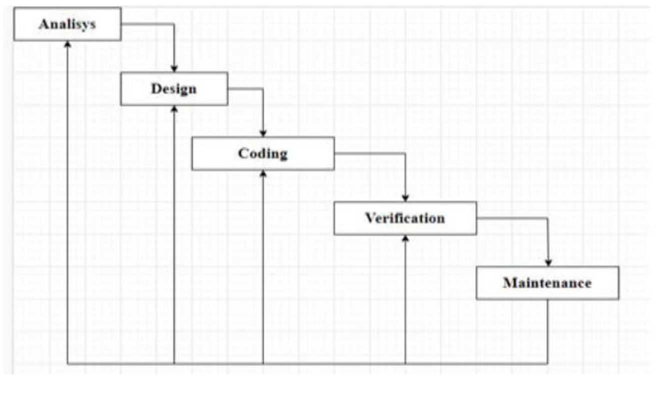


Figure 2. Watefall Model (Ritonga et al., 2022)

The following are the stages of the SDLC Waterfall Method.

a. Analysis
At this stage of collecting aims for some data such as images and information. The images and information used are images and weather information that has been made available by API OpenWeatherMap.

b. Design
At this stage, the aim is to model the application in the form of use case diagrams and to design the user interface.

c. Coding
At this stage, the aim is to build an application that is by analysis and design.

d. Verification
At this stage, the aim is to do testing to find out whether the application can run well or not.

e. Maintenance
In the last stage of the Waterfall Method, the finished software is operated and can go to performance testing.

Performance testing is a testing technique to measure software performance based on performance metrics, namely response time, data load, traffic, resilience, and stability when accessed jointly by many requests and resources (Pradeep & Sharma, 2019) Performance criteria were chosen as comparison parameters because the ideal website can provide services with good performance. Calculation of performance metrics used by utilizing the Lighthouse tool. Lighthouse tool is an open-source tool that measures website quality by auditing performance, accessibility, progressive web apps, and search engine optimization (SEO) (Riady et al., 2019). The measurement of website performance uses the 6 criteria shown in Table 1

Table 1. Criteria performance metric

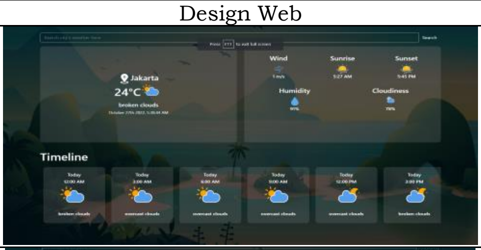
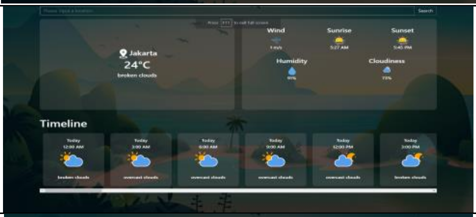
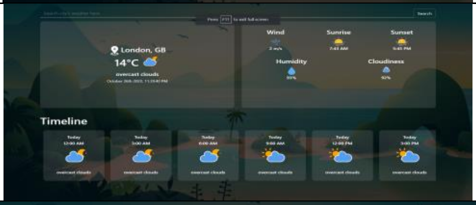
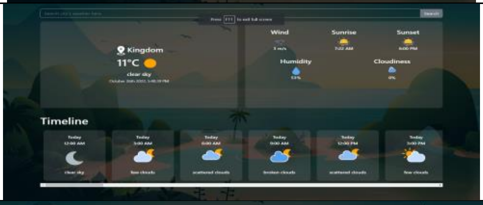
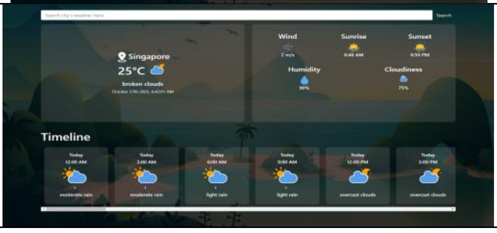
Criteria	Description	Performance Level
First contentful paint (FCP)	The size of the time the website displays the first content in the form of images and text (Gupta & Bartos, 2022).	• fast, indicator is green (0 - 1.8 seconds) • medium, orange indicator (1.8 - 3 seconds) • slow, red indicator (> 3 seconds)
Largest contentful paint (LCP)	Measures the time the website displays the highest resolution content on the viewpoint page when rendering (Yason et al., 2022).	• fast, green indicator (0 - 2.5 seconds) • medium, orange indicator (2.5 - 4 seconds) • slow, red indicator (> 4 seconds)
Time to interactive (TTI)	The website time measure provides interactive feature functions (Armaini et al., 2022).	• fast, green indicator (0 - 3.8 seconds) • medium, orange indicator (3.8 – 7.3 seconds) • slow, red indicator (> 7.3 seconds)
Speed index (SI)	A measure of the speed at which a website displays content visually during page load (Riet, 2020).	• fast, green indicator (0 - 3.4 seconds) • medium, orange indicator (3.4 – 5.8 seconds) • slow, red indicator (>5.8 seconds)
Total blocking time (TBT)	The total time the website blocks all user input responses while the page load until the interactive function runs. In other words, overall duration time it takes between FCP and TTI (Sumedrea et al., 2022) .	• fast, green indicator (0 - 200 milliseconds) • medium, orange indicator (200 – 600 milliseconds) • slow, red indicator (>600 milliseconds)
Cumulative layout shift (CLS)	A score that is calculated based on the cumulative viewpoint shifts across website elements.	• good, indicator is green (0 - 0.1) • need improvement, orange indicator (0.1 - 2.5)

	Measure page load until all lifecycle states disappear (Mokhtari et al., 2022).	• bad, red indicator (> 2.5)

3. RESULTS AND DISCUSSIONS

In this section, the results of system design have already been implemented aster it has been designed. An overview of all websites can be seen in table 2.

Table 2. Weather App Each Frameworks

Framework	Design Web	Link
React		https://react-weather-app-vincent.netlify.app/
Angular		https://angular-weather-app-vincent.netlify.app/
Vue		https://vue-weather-app-vincent.netlify.app/
Svelte		https://weather-app-svelte-vincent.netlify.app/
SolidJS		https://weather-app-solidjs-vincent.netlify.app/

The following performance analysis and testing based on data collected on October 27, 2022, using Lighthouse Tools is listed in table 3. According to table 3, Vue has the highest lighthouse score. The First contentful paint (FCP) is a metric that measures the time a page is first loaded where each part of the page is rendered individually. The FCP value of the five frameworks has a value of less than 1,8 seconds. This means that the load time for the first images and posts takes less time. The Speed index (SI) is a metric that measures the time visual content is visible when the page loads. The SI value of the five frameworks is less than 3.4 seconds, so the entire website load time is very high. The Largest contentful paint (LCP) is the loading time of high-resolution images and blocks of text in the visible viewport. The LCP value of the five frameworks is less than 4 seconds, so the largest content load time is at the best performance. The Time to interactive (TTI) criterion measures the time the interactive features function from the page starting to load until the main content is loaded and can respond quickly to user input. The TTI value of the five frameworks is less than 3.8 seconds, so the time needed for an interactive website is relatively fast. Total blocking time (TBT) is a metric criterion to measure the blocked time so that the user does not input anything. TBT is calculated from the remaining value of TTI minus FCP. Angular's TBT framework has a speed of 10ms, while the other four have no speed because the value obtained is 0.

Table 3 Performance Testing Result

Criteria	**React**	**Angular**	**Vue**	**Svelte**	**SolidJS**
	score	score	score	score	score
FCP	0.3 sec	0.4 sec	0.3 sec	0.4 sec	0.5 sec
LCP	0.7 sec	0.7 sec	0.8 sec	0.7 sec	0.7 sec
TTI	0.3 sec	0.5 sec	0.3 sec	0.4 sec	0.5 sec
SI	2.2 sec	1.7 sec	1.6 sec	2.0 sec	2.0 sec
TBT	0ms	10ms	0ms	0ms	0ms
CLS	0	0	0	0	0
Lighthouse Score	95	97	98	96	96

The graphic comparison with Lighthouse Tools is indicated in figure 3. According to figure 3, React and Vue is the fastest load time, because React and Vue has the fastest FCP and TTI. But, Vue has the slowest LCP than five others. After That, Vue is fastest Speed Index, because only need 1,6 sec.

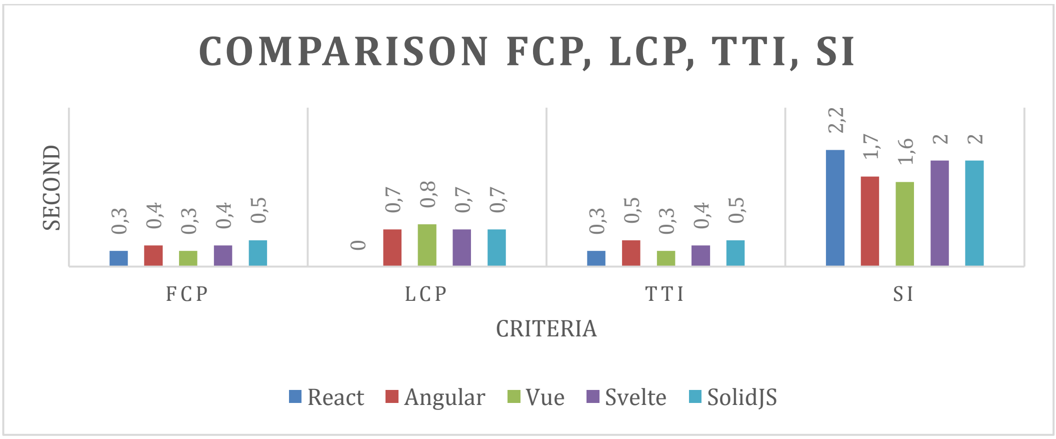


Figure 3 Comparison FCP. LCP, TTI, SI

4. CONCLUSION

By comparing the performance of the Weather App using different front-end Frameworks, this is more realistic way for developers to gain an understanding of performance of these front-end JavaScript frameworks. Based on the results of the analysis and testing conducted in this research using Google Lighthouse, Vue delivering the best performance when implementing the Weather App. Others are not as fast as Vue, but still can deliver good performance when implementing Weather Apps.

REFERENCES

Armaini, I., Dar, M. H., & Bangun, B. (2022). Evaluation of Labuhanbatu Regency Government Website based on Performance Variables. *Sinkron*, *7*(2), 760–766. https://doi.org/10.33395/sinkron.v7i2.11404

Baida, R., Andriienko, M., & Plechawska-Wójcik, M. (2020). *Analiza porównawcza wydajności frameworków Angular oraz Vue.js*. *14*, 59–64.

Białecki, G., & Pańczyk, B. (2021). Performance analysis of Svelte and Angular applications. *Journal of Computer Sciences Institute*, *19*(April), 139–143. https://doi.org/10.35784/jcsi.2633

Carniato, R. (2020). *A Solid RealWorld Demo Comparison of JavaScript Framework Performance*. Level Up Coding. https://levelup.gitconnected.com/a-solid-realworld-demo-comparison-8c3363448fd8

Gupta, A., & Bartos, R. (2022). User Experience Evaluation of HTTP/3 in Real-World Deployment Scenarios. *Proceedings of the 25th Conference on Innovation in Clouds, Internet and Networks, ICIN 2022*, 17–23. https://doi.org/10.1109/ICIN53892.2022.9758130

Levlin, M. (2021). *DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte*.

Malavolta, I., Chinnappan, K., Jasmontas, L., Gupta, S., & Soltany, K. A. K. (2020). Evaluating the impact of caching on the energy consumption and performance of progressive web apps. *Proceedings - 2020 IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2020*, 109–119. https://doi.org/10.1145/3387905.3388593

Mokhtari, H., Saberi, M. K., Amiri, M. R., Vakilimofrad, H., & Moradi, Z. (2022). Evaluating the Speed and Performance of the Websites of Hospitals and Specialty and Super-specialty Clinics of Hamadan University of Medical Sciences by GTmetrix. *E-LIS*.

Nugraha, W., Syarif, M., & Dharmawan, W. S. (2018). Penerapan Metode Sdlc Waterfall Dalam Sistem Informasi Inventori Barang Berbasis Desktop. *JUSIM (Jurnal Sistem Informasi Musirawas)*, *3*(1), 22–28. https://doi.org/10.32767/jusim.v3i1.246

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022). Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*. https://doi.org/10.13052/jwe1540-9589.21311

Patra, T. K. (2022). *Comparison of JavaScript Frameworks: React.js and Vue.js*. *7*(9), 7–11. http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

Persson, M. (2020). *JavaScript DOM Manipulation Performance : Comparing Vanilla JavaScript and Leading JavaScript Front-end Frameworks*. *Independen*, 40. http://bth.diva-portal.org/smash/get/diva2:1436661/FULLTEXT01.pdf%0Ahttp://urn.kb.se/resolve?urn=urn:nbn:se:bth-19531

Pradeep, S., & Sharma, Y. K. (2019). A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications. *Proceedings - 2019 Amity International Conference on Artificial Intelligence, AICAI 2019*, 399–403. https://doi.org/10.1109/AICAI.2019.8701327

Riady, J., Palit, H. N., Andjarwirawan, J., & Petra. (2019). Aplikasi E-Learning Berbasis Progressive Web App Pada Apologetika Indonesia. *Jurnal Infra Petra*, 1–5.

Riet, J. Van. (2020). Study on Mobile Web Performance. *IEEE International, ii.*

Ritonga, R. M., Munthe, I. R., & Purnama, I. (2022). Design an Information System Pendawa Bulucina (RAM) Web-Based Using The Waterfall Method. *Jurnal Mantik*, *6*(36), 383–389. http://www.iocscience.org/ejournal/index.php/mantik/article/view/2211

Sherin, S., Iqbal, M. Z., Khan, M. U., & Jilani, A. A. (2021). Comparing coverage criteria for dynamic web application: An empirical evaluation. *Computer Standards and Interfaces*, *73*(December 2019), 103467. https://doi.org/10.1016/j.csi.2020.103467

Sumedrea, S., Maican, C. I., Chițu, I. B., Nichifor, E., Tecău, A. S., Lixăndroiu, R. C., & Brătucu, G. (2022). Sustainable Digital Communication in Higher Education—A Checklist for Page Loading Speed Optimisation. *Sustainability (Switzerland)*, *14*(16). https://doi.org/10.3390/su141610135

Tjhoernandes, A., Susetyo, Y. A., Kristen, U., & Wacana, S. (2022). Penerapan MAC Address sebagai Autentikasi Aplikasi menggunakan JavascriptBindings Chromium Embedded Framework Python di PT. *Jurnal Inovtek Polbeng*, *7*(1), 26–36.

Yason, S., Sudirman, & Yunus, A. (2022). Analisis Performa Website Sclean Menggunakan Pingdom Tools Dan Page Speed Insights. *KHARISMA Tech*, *17*(1), 113–124. https://doi.org/10.55645/kharismatech.v17i1.213