Speed Matters: Measuring Time to First Byte and Other Key Metrics in VueJS Components Frameworks

Andi Bahtiar Semma*
Program Studi Ilmu Komunikasi dan Penyiaran Islam, Universitas Islam Negeri Salatiga, Indonesia, andisemma@uinsalatiga.ac.id

Keywords:

Web Performance, VueJS, Components Framework, Performance Analysis, GtMetrix.

Abstract

The web development community has witnessed an influx of various frontend frameworks designed to simplify web development. This study compares the performance metrics of three popular front-end frameworks - Buefy, Vuetify, and BootstrapVue. The study evaluates several key performance metrics, including First Contentful Paint, Largest Contentful Paint, Speed Index, Time to Interactive, Total Blocking Time, Connect Duration, Backend Duration, First Paint, DOM Loaded, DOM Interactive, Onload Time, and Fully Loaded Time. The results show that Buefy outperforms Vuetify and BootstrapVue in most of the performance metrics evaluated. Buefy has the fastest First Contentful Paint, Largest Contentful Paint, Speed Index, Total Blocking Time, and Backend Duration. In contrast, BootstrapVue has the slowest performance in most of the metrics. Vuetify's performance is in the middle range, with some metrics performing better than BootstrapVue but worse than Buefy. Based on the study's results, web developers should consider Buefy when choosing a front-end framework for their projects, as it offers the best performance. However, developers should also consider other factors such as ease of use, community support, and available documentation when choosing a framework for their projects.

.

Introduction

Frontend development has come a long way in the past decade, and web developers have a plethora of options when it comes to choosing a framework to build their web applications [1], [2]. With the advent of VueJS [3]–[5], several components frameworks have emerged to facilitate the development of interactive and responsive user interfaces. The frameworks enable developers to create complex web interfaces with reusable components and pre-built UI elements that work seamlessly with VueJS. The popularity of these frameworks has created a demand for an objective evaluation of their performance to aid developers in making informed choices.

The importance of frontend performance cannot be overstated in today's fast-paced digital world. Users expect websites and web applications to load quickly, respond instantaneously to interactions, and provide an overall seamless experience. A slow and unresponsive frontend can negatively impact user experience, which can, in turn, lead to a drop in traffic and revenue.

Hence, it is crucial to select a frontend framework that can deliver optimal performance to meet users' expectations. This study aims to evaluate three popular VueJS components frameworks to identify the framework that delivers the best performance and aid developers in making informed decisions. Several studies have evaluated the performance of frontend frameworks, particularly those built using popular JavaScript libraries such as React and Angular. However, few studies have evaluated the performance of VueJS components frameworks. For instance, a study by Boczkowski, Krzysztof, and Beata Pańczyk (2020) analyses the performance of currently used tools for creating a SPA application interface.

The study was conducted using two applications with the same functionality, implemented in React and Vue.js. The tools available in web browsers and appropriate implementations of own methods were used to measure SPA performance [6]. Another study by Bielak, Konrad, Bartłomiej Borek, and Małgorzata Plechawska-Wójcik conducted a performance analysis and, on its basis, to indicate the most user-friendly and fastest operating framework. Three similar prototypes used to test the speed of selected technologies: Angular, React and Vue [7]. However, no study has yet evaluated the performance of VueJS components frameworks, necessitating the need for this study.

Despite the growing popularity of VueJS components frameworks, there is a dearth of studies evaluating their performance. Existing studies on frontend frameworks mostly focus on React and Angular, neglecting the unique features and performance characteristics of VueJS frameworks. Hence, this study aims to fill the research gap by objectively evaluating the performance of three popular VueJS components frameworks.

The research aim of this study is to evaluate the performance of three popular VueJS components frameworks, Buefy , Vuetify, and BootstrapVue, in terms of their Time to First Byte (TTFB) [8], [9], First Contentful Paint [10], [11], Largest Contentful Paint [12]–[14], Time to Interactive, and Fully Loaded Time [15]–[17]. The study employs a quantitative methodology by measuring the performance of the frameworks using GTmetrix and provides visualizations for better comprehension. The objective is to identify the framework that delivers the best performance and highlight the differences between the evaluated frameworks to aid developers in making informed decisions.

Method

In this study, I aimed to evaluate the performance of three popular frontend frameworks, Buefy, Vuetify, and BootstrapVue, in terms of their Time to First Byte (TTFB), First Contentful Paint, Largest Contentful Paint, Time to Interactive, and Fully Loaded Time. These performance metrics are crucial to ensure the best user experience for websites or applications. As website or application visitors are less likely to wait for more than a few seconds, fast page loading and rendering times are essential to maintain user engagement and satisfaction.

To achieve this objective, I conducted an empirical study using quantitative methodology [18]–[20] by measuring the performance of the frameworks using GTmetrix. The selected frameworks were chosen based on their popularity and usage in the frontend development industry. The study is expected to provide insights into the differences between the evaluated frameworks, and help developers make informed decisions on which framework to use for optimal performance.

The methodology employed in this study was a quantitative approach, using GTmetrix as a tool to measure the performance of the frontend frameworks [21]–[23]. Quantitative methodology was chosen as it is an objective approach that provides empirical evidence and allows for statistical analysis of data. Additionally, it is a widely used and accepted method in performance evaluation studies. GTmetrix was chosen as the research tool as it is a well-established and widely used tool for evaluating website performance. It measures website loading speed, performance, and provides visualizations that are easy to interpret. GTmetrix also allows for comparing website performance against various metrics and provides a grade and a performance score, which are useful indicators for performance evaluation. The sample materials used in this study were the latest versions of the Buefy, Vuetify, and BootstrapVue frameworks. The frameworks were selected based on their popularity and usage in the frontend development industry. All frameworks were evaluated in the same controlled environment to ensure consistency and validity of the results.

To evaluate the performance of the frameworks, GTmetrix was used as a research tool. GTmetrix measures various performance metrics, including TTFB, First Contentful Paint, Largest Contentful Paint, Time to Interactive, and Fully Loaded Time. These metrics were chosen as they are essential for website performance evaluation and are widely accepted in the industry. To measure the performance, GTmetrix was run on the latest versions of each framework under identical settings. The performance metrics and blocking time were recorded, and the results were visualized for better comprehension. The results were analyzed statistically, and the differences between the evaluated frameworks were highlighted.

Result and Discussion

Buefy

The results of the performance analysis of the Buefy website are presented in Table 1. The website's performance was evaluated based on several metrics, including the First Contentful Paint, Largest Contentful Paint, Speed Index, Time to Interactive, Total Blocking Time, Connect Duration, Backend Duration, First Paint, DOM Loaded, DOM Interactive, Onload Time, and Fully Loaded Time. The results show that the Buefy website has an average First Contentful Paint time of 637ms, with the fastest load time of 562ms and the slowest load time of 679ms. The Largest Contentful Paint time averages at 1.5s, with a range of 1.1s to 1.9s. The Speed Index for the website ranges from 852ms to 1.0s, with an average of 907ms. The website's Time to Interactive ranges from 1.1s to 1.3s, with an average of 1.2s. The Total Blocking Time of the website ranges from 101ms to 175ms, with an average of 130ms. The Connect Duration ranges from 68ms to 82ms, with an average of 74ms, and the Backend Duration ranges from 11ms to 13ms, with an average of 12ms. The First Paint time of the website averages at 637ms, while the DOM Loaded and DOM Interactive times range from 709ms to 874ms and 687ms to 825ms, respectively. The Onload Time of the website ranges from 1.4s to 1.6s, with an average of 1.5s, and the Fully Loaded Time ranges from 1.5s to 1.9s, with an average of 1.67s. Detailed result showed in table 1.

Table 1. Performance Metrics for the Buefy

Metric	Test 1	Test 2	Test 3
First Contentful Paint	562ms	671ms	679ms
Largest Contentful Paint	1.6s	1.9s	1.1s
Speed Index	852ms	1.0s	868ms
Time to Interactive	1.1s	1.3s	1.1s
Total Blocking Time	115ms	175ms	101ms
Connect Duration	68ms	82ms	72ms
First Paint	562ms	672ms	679ms
DOM Loaded	709ms	850ms	874ms
DOM Interactive	687ms	823ms	825ms
Onload Time	1.4s	1.6s	1.4s
Fully Loaded Time	1.6s	1.9s	1.5s

Vuetify

In terms of the First Contentful Paint and First Paint metrics, the framework performed well, with an average value of 363 ms, indicating that the initial rendering of the page is fast. However, the performance of the Largest Contentful Paint metric was sub-optimal, with an average value of 2.2 seconds, which is above the recommended value of 2 seconds or less. This can impact the user experience, as it may take longer for users to perceive the content of the page.

The framework's performance in the Speed Index and Time to Interactive metrics was also sub-optimal, with average values of 1.23 seconds and 1.97 seconds, respectively. These values are above the recommended values of 1 second and 1.5 seconds or less, respectively. This indicates that the framework takes longer to fully load and become interactive, which can lead to a slower user experience. The Total Blocking Time metric had an average value of 933 ms, which is slightly above the recommended value of 500 ms or less. This metric measures the amount of time the page is blocked from user interaction, such as clicks or inputs, and can have a significant impact on user experience.

The framework's performance in terms of the Connect Duration and Backend Duration metrics was relatively fast, with average values of 28 ms and 161 ms, respectively. These metrics measure the time it takes for the browser to establish a connection to the server and the time it takes for the server to respond to the request, respectively.

However, the framework's performance in the DOM Interactive metric was sub-optimal, with an average value of 207 ms, which is above the recommended value of 100 ms or less. This metric measures the time it takes for the browser to parse the HTML and create the page's DOM. A slower value for this metric can impact the user experience, as users may perceive a delay in the page's responsiveness. Detailed result showed in table 2.

Table 2. Performance Metrics for the Vuetify

Metric	Test 1	Test 2	Test 3
First Contentful Paint	378ms	381ms	331ms
Largest Contentful Paint	2.2s	2.2s	2.1s
Speed Index	1.2s	1.3s	1.2s
Time to Interactive	1.9s	2.1s	2.0s
Total Blocking Time	871ms	983ms	945ms
Connect Duration	28ms	29ms	27ms
First Paint	378ms	382ms	331ms
DOM Loaded	1.4s	1.4s	1.4s
DOM Interactive	238ms	188ms	194ms
Onload Time	1.9s	1.8s	1.7s
Fully Loaded Time	2.2s	2.2s	2.1s

BootstrapVue

The results of the analysis indicate that Bootsrapvue has a significantly faster response time than other web frameworks in terms of key performance metrics such as First Contentful Paint and Largest Contentful Paint. The average First Contentful Paint time for Bootsrapvue was 233 ms, which is faster than the average time of other frameworks. Similarly, the average Largest Contentful Paint time for Bootsrapvue was 233 ms, which is also faster than the average time of other frameworks.
In terms of Speed Index, Bootsrapvue performed well with an average value of 414 ms, which is better than the average Speed Index of other frameworks. The Time to Interactive metric, which is a measure of the time taken for the page to become fully interactive, showed an average value of 698 ms, which is faster than the average value for other frameworks. The Total Blocking Time metric, which measures the total amount of time spent on blocking tasks, showed an average value of 257 ms, which is lower than the average value for other frameworks. Similarly, the Connect Duration and Backend Duration metrics, which measure the time taken for establishing a connection and processing server-side tasks, respectively, also showed lower average values for Bootsrapvue compared to other frameworks.

Table 2. Performance Metrics for the Vuetify

Metric	Test 1	Test 2	Test 3
First Contentful Paint	378ms	381ms	331ms
Largest Contentful Paint	2.2s	2.2s	2.1s
Speed Index	1.2s	1.3s	1.2s

Time to Interactive	1.9s	2.1s	2.0s
Total Blocking Time	871ms	983ms	945ms
Connect Duration	28ms	29ms	27ms
First Paint	378ms	382ms	331ms
DOM Loaded	1.4s	1.4s	1.4s
DOM Interactive	238ms	188ms	194ms
Onload Time	1.9s	1.8s	1.7s
Fully Loaded Time	2.2s	2.2s	2.1s

Comparison Overview

In this study, I analyzed the performance of three popular Vue.js component libraries, namely Buefy, Vuetify, and BootstrapVue. I measured several performance metrics related to page speed, including First Contentful Paint, Largest Contentful Paint, Speed Index, Time to Interactive, Total Blocking Time, Connect Duration, Backend Duration, First Paint, DOM Loaded, DOM Interactive, Onload Time, and Fully Loaded Time. Our results indicate that there are noticeable differences in the performance of these libraries. In terms of First Contentful Paint, Buefy (562ms) outperformed Vuetify (671ms) and BootstrapVue (679ms). A similar trend was observed for the Largest Contentful Paint metric, where Buefy (1.6s) was faster than Vuetify (1.9s) and BootstrapVue (1.1s).

However, for metrics such as Total Blocking Time and Time to Interactive, BootstrapVue performed significantly better than the other two libraries. BootstrapVue's Total Blocking Time was 101ms, which was much lower than Buefy's (115ms) and Vuetify's (175ms). Similarly, BootstrapVue's Time to Interactive was 1.1s, which was the same as Buefy but faster than Vuetify's 1.3s. These differences could be attributed to various factors, such as the size of the library, the way components are implemented, and the underlying CSS framework. BootstrapVue's better performance in Total Blocking Time and Time to Interactive might be due to its smaller size and the use of Vue.js directives for rendering components. Overall, our results indicate that the choice of component library can have a significant impact on page speed and performance. Developers should carefully evaluate the performance of different libraries before selecting one for their project. Furthermore, the choice of library may depend on the specific performance metrics that are most important for the project.

In practice, our findings can help web developers choose the best Vue.js component library for their project based on their specific requirements. For example, if the project requires faster page load times and smooth user experience, then Buefy may be a good choice. However, if the project involves heavy user interactions and complex components, then BootstrapVue may be a better option due to its faster Total Blocking Time and Time to Interactive.

Conclusions

In conclusion, the data shows that there are notable differences in performance among the Buefy, Vuetify, and BootstrapVue libraries. Buefy outperforms both Vuetify and BootstrapVue in most of the metrics, particularly in terms of First Contentful Paint, Largest Contentful Paint, and Speed Index. BootstrapVue performed the worst in most of the metrics, particularly in terms of Total Blocking Time, where it had the longest duration. Vuetify was generally in the middle of the pack, with a slightly slower performance than Buefy, but faster than BootstrapVue.

It is important to note that performance can depend on various factors, such as the size and complexity of the website, the hardware and software configuration of the user's device, and the network conditions. Therefore, these results may not necessarily apply to all situations. Nonetheless, the findings suggest that using Buefy may result in better performance compared to Vuetify and BootstrapVue. However, developers should also consider other factors, such as the features and usability of the libraries, when making a decision on which one to use.

REFERENCES

[1] F. Ferreira, H. S. Borges, and M. T. Valente, "On the (un-) adoption of JavaScript front-end frameworks," *Software: Practice and Experience*, vol. 52, no. 4, pp. 947–966, 2022.

[2] A. Shenoy, A. Prabhu, A. Shenoy, and A. Prabhu, "Choosing Lightweight Frameworks for Intuitive Web Design," *CSS Framework Alternatives: Explore Five Lightweight Alternatives to Bootstrap and Foundation with Project Examples*, pp. 1–14, 2018.

[3] E. You, "Vue.js - The Progressive JavaScript Framework | Vue.js," *Vue.js - The Progressive JavaScript Framework*, 2023. https://vuejs.org/ (accessed Jan. 09, 2023).

[4] J. Song, M. Zhang, and H. Xie, "Design and implementation of a vue. js-based college teaching system," *International Journal of Emerging Technologies in Learning (Online)*, vol. 14, no. 13, p. 59, 2019.

[5] Z. Erhua, "Research on Web Front-end Application Based on Vue. js," *Science and Technology & Innovation*, vol. 20, pp. 119–121, 2017.

[6] K. Boczkowski and B. Pańczyk, "Comparison of the performance of tools for creating a SPA application interface-React and Vue. js," *Journal of Computer Sciences Institute*, vol. 14, pp. 73–77, 2020.

[7] K. Bielak, B. Borek, and M. Plechawska-Wójcik, "Web application performance analysis using Angular, React and Vue. js frameworks," *Journal of Computer Sciences Institute*, vol. 23, pp. 77–83, 2022.

[8] B. Jun, F. E. Bustamante, S. Y. Whang, and Z. S. Bischof, "AMP up your mobile web experience: Characterizing the impact of Google's accelerated mobile project," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–14.

[9] R. K. Thelagathoti, S. Mastorakis, A. Shah, H. Bedi, and S. Shannigrahi, "Named data networking for content delivery network workflows," in *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, 2020, pp. 1–7.

[10] M. Amjad, M. Hossain, R. Hassan, and M. Rahman, "Web Application Performance Analysis of E-Commerce Sites in Bangladesh: An Empirical Study," *International Journal of Information Engineering and Electronic Business*, vol. 13, no. 2, pp. 47–54, 2021.

[11] M. Hossain, R. Hassan, M. Amjad, and M. Rahman, "Web Performance Analysis: An Empirical Analysis of E-Commerce Sites in Bangladesh.," *International Journal of Information Engineering & Electronic Business*, vol. 13, no. 4, 2021.

[12] V. Vasilijević, N. Kojić, and N. Vugdelija, "A new approach in quantifying user experience in web-oriented applications," in *4th International Scientific Conference on Recent Advances in Information Technology, Tourism, Economics, Management and Agriculture–ITEMA*, 2020, pp. 9–16.

[13] M. Hossain, R. Hassan, M. Amjad, and M. Rahman, "Web Performance Analysis: An Empirical Analysis of E-Commerce Sites in Bangladesh.," *International Journal of Information Engineering & Electronic Business*, vol. 13, no. 4, 2021.

[14] M. Amjad, M. Hossain, R. Hassan, and M. Rahman, "Web Application Performance Analysis of E-Commerce Sites in Bangladesh: An Empirical Study," *International Journal of Information Engineering and Electronic Business*, vol. 13, no. 2, pp. 47–54, 2021.

[15] A. Fabregat *et al.*, "Reactome diagram viewer: data structures and strategies to boost performance," *Bioinformatics*, vol. 34, no. 7, pp. 1208–1214, 2018.

[16] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between QoS metrics and Web QoE using Above-the-fold metrics," in *Passive and Active Measurement: 19th International Conference, PAM 2018, Berlin, Germany, March 26–27, 2018, Proceedings 19*, 2018, pp. 31–43.

[17] T. Böttger *et al.*, "An Empirical Study of the Cost of DNS-over-HTTPS," in *Proceedings of the Internet Measurement Conference*, 2019, pp. 15–21.

[18] P. M. Nardi, *Doing survey research: A guide to quantitative methods*. Routledge, 2018.

[19] H. E. Beck *et al.*, "MSWEP V2 global 3-hourly 0.1 precipitation: methodology and quantitative assessment," *Bulletin of the American Meteorological Society*, vol. 100, no. 3, pp. 473–500, 2019.

[20] N. Basias and Y. Pollalis, “Quantitative and qualitative research in business & technology: Justifying a suitable research methodology,” *Review of Integrative Business and Economics Research*, vol. 7, pp. 91–105, 2018.
[21] K. Sarita, P. Kaur, and S. Kaur, “Accessibility and Performance Evaluation of Healthcare and E-Learning Sites in India: A Comparative Study Using TAW and GTMetrix,” in *Applied Computational Technologies: Proceedings of ICCET 2022*, Springer, 2022, pp. 172–187.
[22] H. Mokhtari, M. K. Saberi, M. R. Amiri, H. Vakilimofrad, and Z. Moradi, “Evaluating the Speed and Performance of the Websites of Hospitals and Specialty and Super-specialty Clinics of Hamadan University of Medical Sciences by GTmetrix,” *Informology*, vol. 1, no. 1, pp. 57–66, 2022.
[23] I. B. K. Manuaba, “Performance Comparison of Text Based Game Prototypes Using GTmetrix,” *Journal of Games, Game Art, and Gamification*, vol. 6, no. 1, pp. 1–6, 2021.