Fachhochschule für Oekonomie und Management

Academic Writing in English

Prof. Paul Acquaro

WS 2023

BWIt_BS WS20 B

Developing a Metric System for Full Stack Javascript Frameworks

Presented by:	Emil Triest Westendallee 97A 14052 Berlin emil@triest.de
Field of Studies:	Bachelor of Science (B.Sc.) - Wirtschaftsinformatik - Business Information Systems - 7. Semester
Matriculation Number:	566224
Reviewed By:	Prof. Paul Acquaro
Place and Date:	Berlin, 12.11.2023

Table of Contents

1. Introduction

1.1 Overview of Full-Stack JavaScript Frameworks

In the world of web development, full-stack JavaScript frameworks have revolutionized how developers create and maintain web applications. These frameworks, integrating both client-side and server-side functionalities, have become indispensable tools in the modern developer's toolkit. Their evolution reflects a trend towards more integrated, efficient, and scalable web application development processes. By consolidating diverse functionalities into cohesive frameworks, they significantly streamline the development process, enabling rapid deployment and iterative improvements in web applications.

1.2 Need for a Comprehensive Metric System

Despite their widespread adoption, a challenge persists in objectively evaluating and comparing these frameworks to determine the most suitable one for specific project requirements. The current landscape lacks a standardized metric system that holistically assesses these frameworks across various dimensions. Such a system is crucial for evaluating aspects like performance efficiency, scalability, user experience, and developer friendliness. A comprehensive metric system would not only facilitate informed decision-making for developers but also contribute to the optimization of web application development, ensuring the selection of the most appropriate framework for a given task.

1.3 Objectives and Scope of the Paper

This paper aims to address this gap by proposing a robust metric system for full-stack JavaScript frameworks. It seeks to synthesize existing metrics and introduce a unified framework for evaluation, providing a balanced assessment of both user-oriented and developer-oriented attributes. The paper will explore various dimensions of these frameworks, including their technical performance, ease of use, and adaptability to diverse project requirements. The ultimate goal is to equip developers and organizations with a comprehensive tool for making informed choices in the rapidly evolving landscape of web application development.

2. Literature Review

2.1 Evolution of Full-Stack JavaScript Frameworks

The full-stack JavaScript framework landscape has seen significant evolution over recent years. Web.dev's resource on fast load times[i] highlights the increasing importance of performance in web applications, a key factor driving the development of these frameworks. Akamai Technologies' survey on web performance expectations[ii] also underscores the growing user demand for efficient and responsive web applications, further necessitating robust and versatile frameworks.

2.2 Existing Metrics in Framework Evaluation

Current metrics for evaluating these frameworks often emphasize technical performance. Aigner et al.'s work on JavaScript performance[iii] and Jacob, Ng, & Wang's exploration of memory systems[iv] offer insights into critical performance metrics. However, as Robillard[v] and Storey et al.[vi] suggest, there is a need to consider developer experience and ease of implementation when evaluating these frameworks, which is often overlooked in traditional metrics.

2.3 Gaps in Current Evaluation Methods

Despite these insights, existing methods fall short in providing a holistic evaluation of full-stack JavaScript frameworks. This gap is evident in the lack of comprehensive metrics that balance both technical performance and developer experience. As indicated by Weber[vii] and Adhikari[viii], there is a need for a unified metric system that not only assesses the technical prowess of these frameworks but also their practicality, adaptability, and ease of use for developers, ensuring a more balanced and informed selection process for web application development.

3. Methodology

3.1 Criteria for Metric Selection

The selection of metrics for this system was guided by the need to address both the technical performance and the developer experience associated with full-stack JavaScript frameworks. This involved a detailed review of literature and existing evaluation methods, highlighting key performance indicators such as load times, API response times, and resource utilization, alongside developer-centric factors like documentation quality, community support, and modularity

3.2 Framework for Metric Development

The metric system was developed to integrate these diverse criteria into a coherent evaluation framework. Quantitative metrics were derived from technical performance data, while qualitative metrics were gathered to assess aspects of developer experience and framework usability. This dual approach ensures that the system provides a balanced view, catering to both the end-user's and the developer's perspectives in the context of modern web application requirements.

3.3 Data Sources and Selection Criteria

The methodology employed a multi-source approach for data collection, encompassing academic literature, industry reports, and framework documentation. Sources were selected based on their relevance to the evolving field of web development, their academic rigor, and the practical insights they offer into current trends and challenges in full-stack JavaScript framework development, ensuring a comprehensive and up-to-date foundation for the metric system.

4. Analysis

4.1 Evaluation of Existing Metrics

The analysis begins with a thorough review of existing metrics used in evaluating full-stack JavaScript frameworks. This includes technical performance metrics such as Load Time/Page Load Time (PLT) and API Response Time, which are essential for assessing the efficiency and speed of applications (web.dev, n.d.; Akamai Technologies, 2014). Additionally, developer-oriented metrics like Documentation Quality and Modularity (Robillard, 2009;

Bosch, 1999) are examined to understand how they impact the ease of framework implementation and maintenance. This evaluation highlights the strengths and limitations of current metrics, providing a basis for the development of a more comprehensive system.

4.2 Development of the Unified Metric System

Building on the evaluated metrics, a unified metric system is proposed, combining both technical performance and developer experience aspects. The system categorizes metrics into two primary dimensions: 'User-Oriented Performance Metrics' and 'Developer-Oriented Experience Metrics.' User-oriented metrics focus on aspects like PLT, JavaScript Execution Time, and Memory Usage, which directly affect user experience (Jacob, Ng, & Wang, 2010). Developer-oriented metrics include factors such as Documentation Quality, Community Support, and Modularity, which influence the ease of framework usage and adaptability (Storey et al., 2006). This unified approach ensures a balanced assessment, catering to the comprehensive needs of web application development.

5. Results

5.1 User-Oriented Performance Metrics

1. Load Time/Page Load Time (PLT)
    - Definition: The time taken for a webpage to fully load and render in a browser.
    - Importance: Directly affects user experience and engagement, with faster PLT leading to increased user satisfaction.
2. Time to First Byte (TTFB)
    - Definition: The time it takes for a user's browser to receive the first byte of data from the server.
    - Importance: Provides insight into server response times and the initial stages of data retrieval.
3. First Contentful Paint (FCP) and Largest Contentful Paint (LCP)
    - Definition: FCP measures the time it takes for the first content to appear on the screen, while LCP measures when the largest content element is rendered.
    - Importance: Indicate how quickly users see meaningful content on the screen.

4. JavaScript Execution Time
    - Definition: The time taken for the JavaScript code to execute.
    - Importance: Slow or blocking scripts can significantly hinder application performance.
5. API Response Time
    - Definition: The time taken for an API to process a request and return a response.
    - Importance: Critical in full-stack applications where front-end components rely on back-end APIs. Slow API responses can negatively impact user experience.
6. Memory Usage
    - Definition: The amount of memory consumed by the application.
    - Importance: Excessive memory consumption can lead to slowdowns, crashes, and reduced server capacity.

5.2 Developer-Oriented Experience Metrics

1. Documentation Quality
    - Definition: Clarity, comprehensiveness, and accuracy of official documentation, tutorials, and guides.
    - Importance: Essential for onboarding new developers and assisting experienced ones.
2. Community Support and Ecosystem
    - Definition: The presence of an active developer community, available plugins/extensions, and third-party resources.
    - Importance: A vibrant community can provide solutions to common issues, share best practices, and develop tools or plugins that extend the framework's capabilities.
3. Consistency and Predictability
    - Definition: Uniformity in naming conventions, patterns, and behaviors within the framework.
    - Importance: Allows developers to make educated guesses about unfamiliar parts of the API or framework.
4. Modularity and Flexibility

- Definition: The framework's ability to allow modular development and be flexible enough for various use cases.
- Importance: Enables developers to use only the components they need, leading to more efficient and cleaner codebases.

5. Clear Error Messages and Debugging Tools
    - Definition: Informative error messages and powerful debugging tools incorporated within the framework.
    - Importance: Help developers quickly identify and fix issues, reducing development time.
6. Scalability
    - Definition: The framework's capability to adapt to growing application needs without significant modifications.
    - Importance: Ensures that the framework can support the application as it grows in complexity and user base.
7. Integration with Tools and Services
    - Definition: How easily the framework integrates with popular development tools and services.
    - Importance: Ensures that developers can use their preferred tools and services without friction.

5.3 Findings from the Metric System Development

The development of the metric system for evaluating full-stack JavaScript frameworks yielded critical insights into the multifaceted nature of framework assessment. The system successfully integrated both technical performance metrics and developer experience metrics, revealing the complex interplay between these two dimensions.

A key finding was the identification of essential metrics that were previously underemphasized in framework evaluations. These included aspects like modularity, scalability, and community support, which are crucial for long-term framework sustainability and effectiveness. The system's ability to encapsulate these broader aspects, alongside traditional performance metrics such as load times and execution efficiency, provided a more comprehensive framework for evaluation.

Another significant outcome was the establishment of a balanced approach to framework assessment. Rather than leaning solely towards technical performance or developer convenience, the metric system facilitated a dual-perspective evaluation, considering how well frameworks serve both end-user requirements and developer needs. This holistic approach is particularly relevant in scenarios where trade-offs between user experience and developer efficiency are common.

The results underscored the potential of the proposed metric system to reshape the criteria and methods used in selecting and utilizing full-stack JavaScript frameworks, suggesting a shift towards more informed, balanced, and context-sensitive decision-making in web application development.

6. Discussion

6.1 Implications of the Proposed Metric System

The introduction of the proposed metric system for full-stack JavaScript frameworks represents a paradigm shift in the evaluation process. By incorporating both user-oriented and developer-oriented metrics, the system provides a more nuanced and comprehensive understanding of a framework's capabilities. This approach addresses the need for a balance between technical performance and usability, encouraging developers to consider a broader range of factors when choosing a framework. The system's emphasis on modularity and community support, in particular, highlights the importance of not just immediate performance but also long-term viability and adaptability of a framework.

6.2 Relevance to Current and Future Web Development Trends

The proposed metric system aligns with current trends in web development, which increasingly prioritize both efficiency and user experience. As web technologies continue to evolve, the flexibility of this metric system allows it to adapt to new frameworks and technologies, maintaining its relevance. The inclusion of scalability and community support metrics anticipates future trends in web development, such as the growing importance of open-source communities and the need for frameworks to support increasingly complex and data-intensive applications. This forward-looking approach ensures that the metric system remains a valuable tool for framework evaluation in the ever-evolving landscape of web development.

6.3 Potential Impact on Development Practices

The implementation of this metric system could significantly influence web development practices. By providing a structured and comprehensive framework for evaluation, it encourages developers to take a more holistic view of their technology choices, beyond just immediate technical requirements. This could lead to more informed decisions, potentially affecting the popularity and usage of certain frameworks. Moreover, the system's emphasis on developer experience and community aspects may prompt framework creators to focus more on these areas, leading to overall improvements in the quality and user-friendliness of full-stack JavaScript frameworks. In turn, this can enhance the efficiency and effectiveness of web application development.

7. Conclusion

7.1 Synthesis of Research Findings

The development and implementation of the proposed metric system represent a significant advancement in the evaluation of full-stack JavaScript frameworks. This research successfully identified and integrated key metrics that encompass both technical performance and developer experience, offering a more holistic approach to framework assessment. The findings reveal the importance of considering a range of factors — from load times and memory usage to documentation quality and community support — in selecting a framework. This comprehensive evaluation method addresses the complexities and varying needs inherent in modern web application development.

7.2 Practical Recommendations

For developers and organizations, this metric system provides a valuable tool for making informed decisions about framework selection. It encourages considering both immediate technical requirements and long-term sustainability, ensuring the chosen framework aligns with the specific needs and goals of their web applications.

7.3 Directions for Future Research

Future research should focus on refining and expanding this metric system to accommodate emerging technologies and evolving web development practices. Further studies could explore the application of this system in different contexts and its adaptability to new frameworks, ensuring its continued relevance and utility in the field of web development.

Declaration in lieu of oath

I hereby declare that I produced the submitted paper with no assistance from any other party and without the use of any unauthorized aids and, in particular, that I have marked as quotations all passages, which are reproduced verbatim or nearby verbatim from publications. Also, I declare that the submitted print version of this thesis is identical to its digital version. Further, I declare that this thesis has never been submitted before to any examination board in either its present form or in any other similar version. I herewith agree that this thesis may be published. I herewith consent that this thesis may be uploaded to the server of external contractors for the purpose of submitting it to the contractors' plagiarism detection systems. Uploading this thesis for the purpose of submitting it to plagiarism detection systems is not a form of publication.

E Triest

Berlin, 12.11.2023

Signed, Emil Triest

Bibliography

[i] web.dev. (n.d.). Fast load times. Retrieved from https://web.dev/explore/fast.

[ii] Akamai Technologies. (2014). Consumer Web Performance Expectations Survey.

[iii] Aigner, M., Hütter, T., Kirsch, C., Miller, A., Payer, H., & Preishuber, M. (2014). ACDC-JS. Proceedings of the 10th ACM Symposium on Dynamic Languages, 67–78. https://doi.org/10.1145/2661088.2661089

[iv] Jacob, B., Ng, S., & Wang, D. (2010). Memory Systems: Cache DRAM Disk. Morgan Kaufmann.

[v] Robillard, M. P. (2009). What makes APIs hard to learn? Answers from developers. Software, IEEE, 26(6), 27-34.

[vi] Storey, M. A. D., Cheng, L. T., Bull, I., & Rigby, P. C. (2006). Shared waypoints and social tagging to support collaboration in software development. In Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (pp. 181-190).

[vii] Weber, N. (2022). Evaluation and Comparison of Full-Stack JavaScript Technologies.

[viii] Adhikari, A. (2016). Full Stack JavaScript: Web Application Development with MEAN.