Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

JOURNAL OF INFORMATION SYSTEM AND TECHNOLOGY MANAGEMENT (JISTM)
www.jistm.com

QUANTITATIVE STUDY ON VIRTUAL DOM DIFF PERFORMANCE WITH DIFFERENT KEY ATTRIBUTE USAGE: A CASE OF VUE.JS

Yao Lei[1], Khairul Anwar Sedek[2*], Azlan Ismail[3]

[1] Faculty of Computer and Mathematical Sciences, Universiti Teknologi Mara (UiTM),40450, Shah Alam, Malaysia
Email: Yaolei202506@163.com
[2] Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM), Perlis Branch, Arau Campus, 02600 Arau, Perlis, Malaysia
Email: khairulanwarsedek@uitm.edu.my
[3] Faculty of Computer and Mathematical Sciences, Universiti Teknologi Mara (UiTM),40450, Shah Alam, Malaysia; Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Universiti Teknologi Mara (UiTM), 40450, Shah Alam, Malaysia
Email: azlanismail@uitm.edu.my
[*] Corresponding Author

Article Info:

Article history:
Received date: 30.06.2025
Revised date: 28.07.2025
Accepted date: 15.08.2025
Published date: 19.09.2025

To cite this document:

Yao, L., Sedek, K. A., & Ismail, A. (2025). Quantitative Study on Virtual Dom Diff Performance with Different Key Attribute Usage: A Case of Vue.JS. *Journal of Information System and Technology Management, 10* (40), 286-308.

DOI: 10.35631/JISTM.1040020

Abstract:

As the complexity of modern web applications continues to grow, frontend performance has become a critical factor influencing user experience. Vue.js, as a mainstream framework, has gained widespread adoption thanks to its MVVM framework model and virtual DOM technology. Its Diff algorithm minimizes operations to update the real DOM, significantly improving page response efficiency. As the unique identifier for virtual nodes, the key attribute directly influences the Diff algorithm's node matching and reuse strategies, thereby impacting rendering performance. While extensive research has focused on performance comparisons of frontend frameworks, there remains a lack of systematic empirical analysis on the performance implications of Vue.js's internal key attribute usage (unique ID, index, or none) across different scenarios. This paper conducts a quantitative study using Vue 2.0 as the platform, constructs three typical pages (simple page, high-frequency large-data list page, and nested structure page), and builds a minimum-maximum inverse normalization and linear weighted scoring model to analyze overall performance. Additionally, by combining analysis of variance (ANOVA) and Tukey post-hoc tests, the performance differences under different settings are statistically validated. The research results indicate that in complex high-data-volume pages and nested page scenarios, selecting a unique ID as the key attribute can effectively improve page rendering performance, providing theoretical and practical support for front-end performance optimization.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Keywords:

Attribute Optimization, Diff Algorithm, Virtual DOM, Web Frontend Performance

Introduction

In the current technical context of the pursuit of excellent user experience in Web applications, front-end rendering performance optimization has become a core engineering challenge in modern Web development (Ekpobimi et al., 2024; Simões et al., 2024). To cope with the performance bottleneck of complex pages, the Vue.js framework significantly improves page response efficiency by introducing the Virtual DOM mechanism (Virtual DOM) and the efficient Diff algorithm (Ma, 2024; Vyas, 2022; Ревенчук & Стешко, 2022), which is responsible for comparing the differences between the two virtual DOM trees before and after and updating the real DOM with minimal operations, thus reducing unnecessary DOM operation overhead (Lu et al., 2021; Schwab et al., 2021). The key attribute usage plays a crucial role in identifying and updating virtual nodes as node identifiers. Different key attribute usage (unique ID, index, none) will directly affect the way the Diff algorithm recognizes the nodes, which leads to a significant impact on the rendering performance.

Although most of the current mainstream literature on key attribute usage stays on official recommendations and textual descriptions (Barth et al., 2022; Kanwal et al., 2021; Li et al., 2023), they still remain at the level of empirical summarization or conceptual exposition, lacking quantitative empirical analysis based on systematic experiments. Especially in the scenarios of high-frequency updating of big data pages and nested structure pages, whether the key policy can have a significant impact on the page performance is not sufficiently researched and still needs to be systematically analyzed and explored.

In this paper, we aim to construct three typical page scenarios (static small list page, high-frequency big data list page, and nested structure page) based on the Vue2.0 framework from a practical perspective by using the key policy as the independent variable, keeping other non-critical variables all the time, collecting the core performance indexes such as DOM rendering time, FCP, LCP, and TBT through the batch testing tool, and constructing the Min-Max inverse normalization and the Min-Max inverse normalization. The core performance indicators, such as DOM rendering time, FCP, LCP, TBT, etc., are collected by batch testing tools, and the min-max inverse normalization and linear weighted scoring models are constructed to quantitatively analyze overall performance. Meanwhile, the performance differences of the strategies in different scenarios are statistically validated by combining but-factor analysis of variance (ANOVA) and Tukey's post-hoc test. The study aims to provide front-end developers with a comparable, quantifiable, and generalizable basis for optimizing key attribute usage and to promote front-end performance engineering towards more refined and data-driven conversations.

The structure of this paper is organized into five main sections. Section 1, Introduction, outlines the motivation, research objectives, and significance of front-end performance optimization. Section 2, Background, reviews the concepts of virtual DOM, Diff algorithms, and key attribute usage in modern front-end frameworks. Section 3: This section reviews and compares existing studies on key attribute usage in virtual DOM performance optimization.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Section 4: Research Methodology and Experimental Scenarios describes the experimental design, including the key variables, test scenarios, and evaluation metrics, as well as the tools and procedures employed. Section 5: Experimental Results and Performance Analysis, presents and interprets the experimental findings, highlighting the impact of key attribute usage under different scenarios. Finally, Section 6, Summary and Future Work summarizes the main contributions of the study and suggests directions for further research and improvement.

Background

Vue.js Framework

Vue.js is a progressive JavaScript framework for building user interfaces. It uses virtual DOM and responsive data binding technology to ensure efficient performance while providing an intuitive and declarative development experience. With its lightweight, easy-to-use, and superior performance, Vue.js has become one of the mainstream front-end frameworks.

The framework used in this study, Vue.js, also uses virtual DOM and Diff mechanisms and provides developers with flexible support for multiple key strategies (Li et al., 2023). Developers can choose to omit keys, use element indexes, or assign unique identifiers. Different strategies may have different effects on rendering efficiency in complex or high-frequency update scenarios (Kanwal et al., 2021). Theoretically understanding these effects is important for subsequent empirical testing and guiding front-end development best practices (Barth et al., 2022).

Virtual DOM and Diff Algorithm

In the context of the continuous development of the front end, improving the performance of web applications has become one of the important research areas in computer engineering. Virtual DOM is one of the key innovations to meet this challenge and is widely used in modern JavaScript frameworks such as Vue, React, and Angular (Ekpobimi et al., 2024b). Virtual DOM is a lightweight description of the real DOM. By comparing the differences between the two virtual trees, the real DOM is updated as little as possible to improve rendering efficiency. When the state changes, the update happens in the virtual DOM first, and then it checks against the last version using the Diff algorithm to find the smallest number of changes needed, which are then applied to the real DOM, greatly boosting the performance of dynamic interfaces (Ekpobimi et al., 2024b). The specific figure is as follows:

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

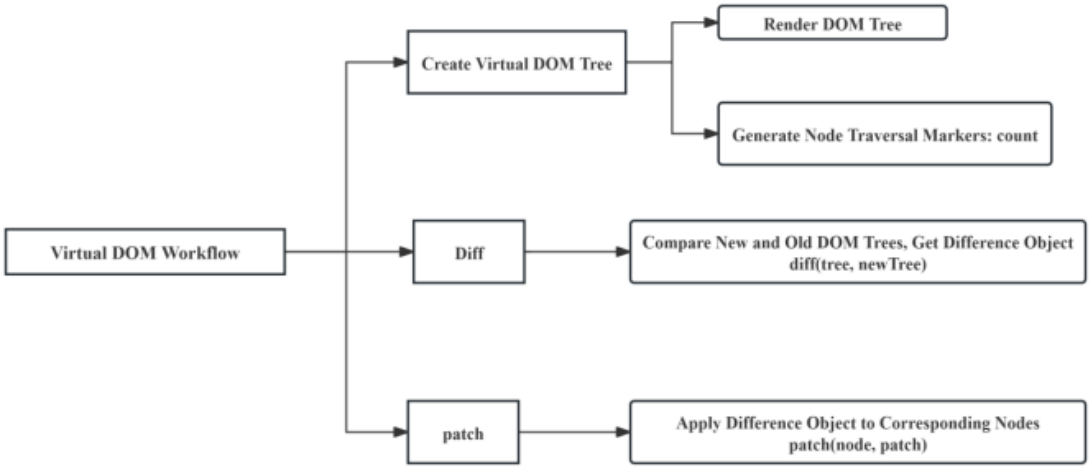

Figure 1: Virtual DOM working process

Key Attribute Usage

In modern frameworks, the Diff algorithm not only focuses on the addition, deletion, and modification of nodes but also relies on developers to provide unique identifiers to optimize node matching. This is enough to illustrate the importance of the key attribute in performance optimization. In Vue.js, the key attribute is used to uniquely identify each node in the list. The Diff algorithm determines whether the node needs to be reused or destroyed and rebuilt by comparing the key value. Reasonable setting of the key attribute helps avoid unnecessary DOM operations, improve rendering performance, and prevent problems such as state confusion.

Common key configuration strategies include:

Table 1: Key Attribute Usage in Vue.js

Strategy	Description
Unique ID	Each element uses a unique and independent ID as the key, ensuring the most accurate node matching.
Index	Uses the element's position in the list as the key, suitable for simple scenarios without order changes.
None	By default, Vue attempts to reuse elements based on their order, which may lead to state inconsistency.

The common way of writing is as follows:

```
<!-- unique ID -->
<li v-for="item in list" :key="item.id">{{ item.name }}</li>

<!-- index -->
<li v-for="(item, index) in list" :key="index">{{ item }}</li>

<!-- none -->
<li v-for="item in list">{{ item }}</li>
```

Figure 2: Examples of Key Attribute Usage in Vue.js List Rendering

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Each strategy will show different performance characteristics under different UI complexity and data update rules. This study conducts a systematic evaluation around these strategies.

Literature Review

In order to better understand the current state of research on the use of key attributes usage in virtual DOM performance optimization, this paper compares existing research from multiple dimensions, including the frameworks and technologies used, the diff strategies analyzed, the key attribute configurations, the performance metrics evaluated, and the evaluation methods adopted. This comparison highlights the strengths and limitations of existing research and points out the deficiencies that this study aims to address.

Table 2: Comparison of Existing Studies

Study	*Framework /Tech*	*Diff Strategy Analyzed*	*Key Attribute Usage*	*Performance Metrics Evaluated*	*Evaluation Method*
1	Vue.js\react\angular\svelte	Default	No / Index	DOM rendering time	Use preset scripts and automation tools to simulate user operation processes
2	Vue.js\React	Default	index\unique ID	FCP\DOM operations	Lighthouse + Chrome Dectools + manual test
3	Million.js (comparison between Vue and React)	Compilation optimization	unique ID	DOM rendering time	Synthetic rendering test
4	Vue.js\react\angular	default	Not specified	Memory usage, rendering time, table operation speed	Synthetic table operation test (chrome 104)
5	Vue.js\react\vanilla	default	Not specified	DOM operations	Synthetic script evaluation
6	Vue.js\react	default	index\unique ID	FCP\TBT\rendering time	Lighthouse + chrome DecTools

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

7	Vue.js\react\angular	default	No\index	Rendering time, LCP	Lighthouse + Chrome Dectools+multi-scene test
8	Vue.js\react\angular	default	index\unique ID	DOM operation time, CPU usage	Manually perform a series of operations to test performance + experimental group
9	Vue.js	default	Unique ID	Conceptual discussion	Literature review, best practice summary
10	React	default	Index	Conceptual discussion	Theoretical exposition,
11	Vue.js\react	default	Unique ID	Conceptual discussion, empirical summary	Literature review, empirical observations
This study	*Vue.js*	*default*	*Unique ID*、*index*、*none*	*DOM rendering time*、*FCP*、*LCP*、*TBT*	*Batch testing* + *Lighthouse* + Chrome Dectools

Framework and Technologies

Most studies focus on mainstream frameworks such as Vue.js, React, Angular, and occasionally Svelte or Vanilla.js. Bai (2022) uniquely investigates Million.js, but its findings have limited generalizability to widely adopted frameworks.

Diff Strategies Analyzed

Almost all studies adopt the default diffing mechanism without explicitly controlling or experimenting with alternative strategies. Only Bai (2022) explores compilation-level optimization in Million.js, indicating a lack of systematic exploration of diffing strategies in the context of key attribute usage

Key Attribute Configurations

Significant limitations exist regarding key attribute configurations. Many studies do not explicitly specify key settings or limit their analysis to simple index vs. unique ID comparisons. Although some studies include the “none” strategy, they fail to examine its interaction with UI complexity or data volume.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Performance Metrics and UI Complexity

Earlier works measure only DOM rendering time (Levlin, 2020; Bai, 2022), while later studies expand to include FCP, LCP, TBT, and CPU usage. However, these metrics are typically collected in controlled, synthetic environments, failing to fully capture realistic UI complexity and dynamic behavior.

Evaluation Methods

Existing studies employ diverse evaluation methods: automated scripts, Lighthouse audits with manual validation, and manual experimental groups. Literature reviews and conceptual discussions (Barth et al., 2022; Kanwal et al., 2021; Li et al., 2023) further highlight the lack of empirical, controlled experiments.

Research Framework And Experimental Design

Research Framework and General Idea

This study centres on the Vue.js framework as the experimental platform and focuses on the quantitative analysis of the impact of key attribute usage on rendering performance in the virtual DOM diff process. Three typical key attribute usages (unique ID, index, and none) are selected for the experiment, and the experiment is comparable and effective by constructing real front-end pages and keeping all non-critical variables consistent, with only the key attribute usage as the independent variable. In terms of performance evaluation, this paper introduces a linear weighted scoring model to comprehensively analyze the core metrics, including DOM rendering time (40%), FCP (20%), LCP (20%), and TBT (20%). All the metrics are Min-Max inverse normalized and assigned scores to quantify the overall performance of different strategies under multiple page complexity levels. The aim is to explain the actual impact of key attribute usage on virtual DOM performance and provide front-end developers with empirical evidence and replicable strategy suggestions for component rendering optimization.

This study uses quantitative research methods to evaluate the performance of different virtual DOM diff strategies in the Vue framework through experimental performance data and statistical analysis. Quantitative research can objectively compare the performance differences of different strategies in multiple scenarios and verify their significance through data-supported statistical methods (such as ANOVA and Tukey HSD tests). This method ensures the objectivity and repeatability of the research results and provides reliable data for front-end performance optimization.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

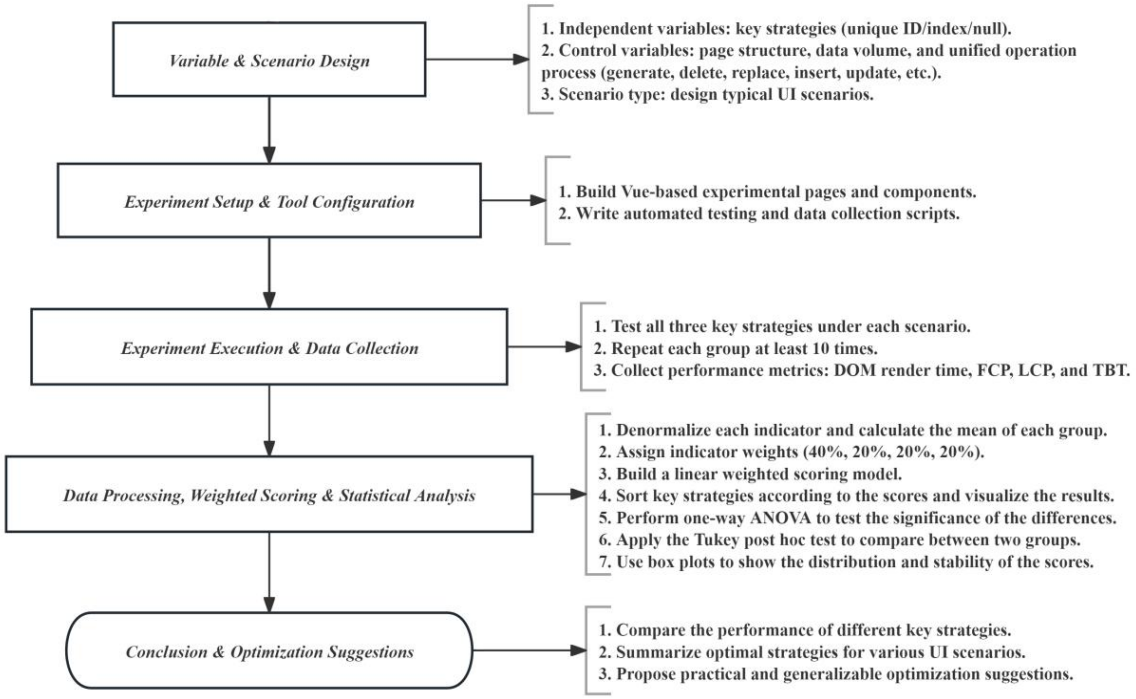

Figure 3: Research Workflow For Evaluating The Performance Impact of Vue's Virtual DOM Key Attribute Usage

The overall research workflow for evaluating the performance impact of Vue's virtual DOM key attribute strategies is illustrated in Figure 3. This workflow consists of six sequential steps, each addressing a specific aspect of the experimental design and analysis.

The first step, Variable & Scenario Design, defines the independent variable as the key attribute strategy (unique ID, index, or null), while controlling for factors such as page structure, data volume, and operation type (e.g., insert, delete, update). Three representative UI scenarios—simple, dynamic, and nested—are designed to reflect typical web application patterns, as described in Experimental Variables and Controlled Environment.

The second step, Experiment Setup & Tool Configuration, involves building the Vue-based experimental pages and components and preparing automated testing and data collection scripts. The detailed configuration of the testing environment, including hardware, software, and performance measurement tools, is presented in Experimental Page Scenarios.

The third step, Experiment Execution & Data Collection, executes the experiments by applying each of the three key strategies under all scenarios, with each group of tests repeated at least 10 times to ensure consistency. Performance metrics, such as DOM render time, First Contentful Paint (FCP), Largest Contentful Paint (LCP), and Total Blocking Time (TBT), are recorded during this stage, as explained in Experimental platform and tool configuration.

The fourth step, data processing, weighted scoring & statistical analysis, standardizes the collected metrics, assigns weights to indicators, and builds a weighted scoring model. Statistical methods, including one-way ANOVA and Tukey post hoc tests, are applied to assess

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

the significance of differences among strategies. Box plots are used to visualize the distribution and stability of data collection and processing methods.

Finally, the fifth step, Conclusion & Optimization Suggestions, summarizes the findings by comparing the performance of the key strategies, recommending optimal strategies for different UI scenarios, and providing practical and generalizable guidance for front-end performance optimization.is presented in Indicator System Construction and Weight Setting.

Experimental Variables and Controlled Environment (Variable & Scenario Design)

This section elaborates on the variable and scenario design step illustrated in Figure 3. In the experiment, all variables except the key attribute usage are controlled to ensure that the differences only originate from the way the key attribute usage is set. The experimental variable division and content are shown in Table 3 below:

Table 3: Experimental Variable Classification And Scenario Types

Type	**Description**
Independent Variable	key attribute usage: unique ID, index, none
Dependent Variables	DOM rendering duration, FCP (First Contentful Paint), LCP (Largest Contentful Paint), TBT (Total Blocking Time)
Control Variables	Page structure, data scale, unified operation process (generate, delete, update, etc.)
Scenario Types	Simple page、HF big data page, nested structure page

Experimental Page Scenarios (Experiment Setup & Tool Configuration)

In order to be close to the diversity of actual Web applications, this paper constructs three representative front-end page types.

Table 4: Page complexity and scene division

Page Type	**Functional Characteristics**	**Structural Depth**	**List Length**
Simple Page	Static display-type page	≥2 levels	≤10
HF-Data Page	Contains high-frequency updating data lists with periodic refresh	≥2 levels	≥2000
Nester Structure Page	Multi-level nested structure with parent-child components and complex nested lists	≥3 levels	≥1500

Simple Page Scenario simulates a static information display page, such as book catalog management, with a small number of list items (less than or equal to 10 items) and a stable structure, which is used for fixing the content or refreshing the content periodically and is used to test the performance of different key attribute usage in the basic static rendering. This scenario is used to test the performance of different key attribute usage in the basic static rendering and to provide a performance baseline for the subsequent comparison of highly dynamic scenarios.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

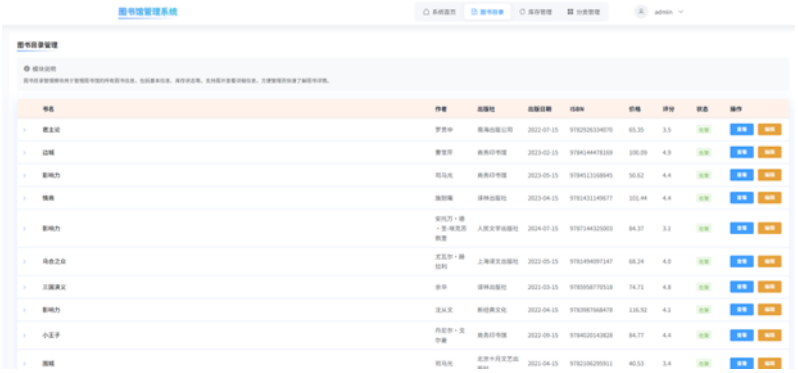

Figure 4: Book Management Interface For Performance Testing Scenarios

In order to evaluate the practicality of the key attribute usage in the case of large-scale data and highly concurrent updates, it is necessary to simulate a high-frequency dynamic big list scenario, which simulates such pages as displaying the business of inventory management, and the data reaches 3,577 items. In this context, the performance bottleneck of the virtual DOM Diff algorithm and the impact of the key attribute usage are most significant.

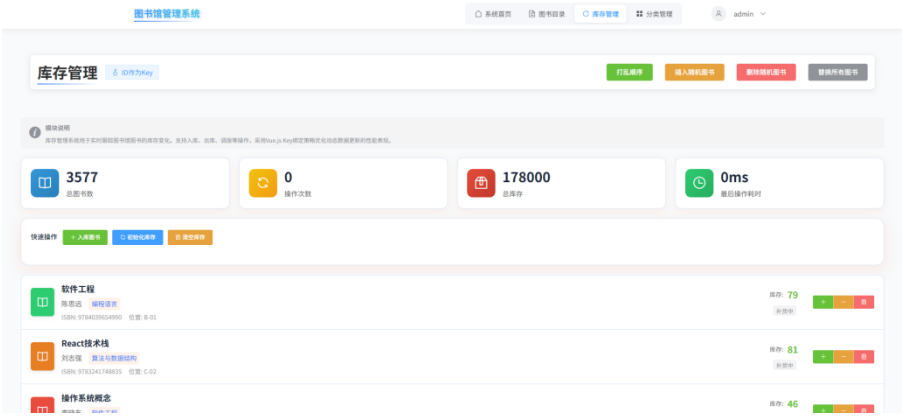

Figure 5: Inventory Management Interface for HF Frequency Large Data Scenario

Library management systems are often designed with hierarchical institutional data, such as book classification-secondary classification-multi-level nested display of specific books and other multi-level data relationships. This scenario can evaluate the impact of key attribute usage on DOM rendering time and other performance under complex nesting conditions.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

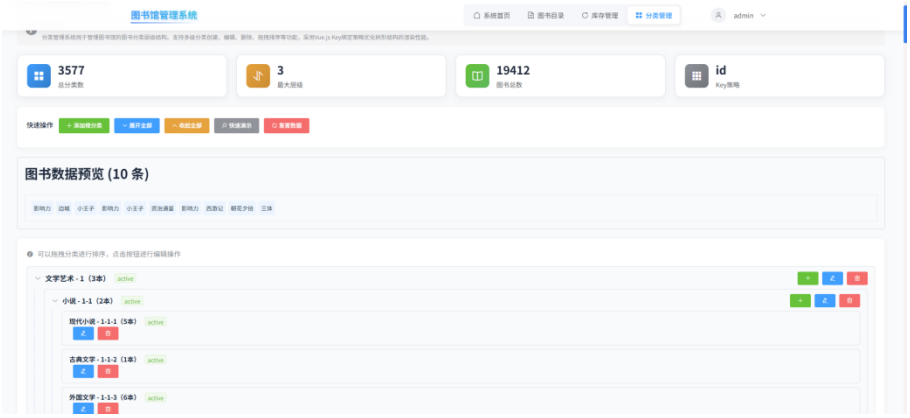

Figure 6: Hierarchical Book Category Interface For Nested Structure Scenario

Experimental Platform and Tool Configuration (Experiment Execution & Data Collection.)
The configuration of the experimental environment is shown in Table 5 to ensure that the hardware and software platform is stable and consistent, with high-precision performance acquisition capabilities.

Table 5: Experimental Environment Configuration

Category	Configuration Details
Operating System	Windows 10 Home Edition, 64-bit
Processor	AMD Ryzen 5 4600U @2.10GHz
Memory	16GB DR4
Browser	Google Chrome, Version 137.0.7151.69 (Official Build)
Frontend Framework	Vue 2.17.16、Vue CLI 5.0.0
Scripting Tool	Node.js 22.16.0
Performance Tools	Lighthouse v12.6.1、Performance API

This study selected Vue.js 2.0 as the experimental framework, primarily based on its extensive engineering application foundation, stable virtual DOM mechanism, and robust toolchain support. On the one hand, Vue 2.0 is still widely adopted in numerous projects, particularly in Asia and in small- to medium-sized web systems, where it enjoys a high adoption rate. On the other hand, its virtual DOM architecture is relatively stable, with a clear and logically rigorous implementation of the diff algorithm, making it suitable for comparative experiments. Additionally, mainstream performance analysis tools such as Vue CLI, Lighthouse, and Performance API integrate well with Vue 2.0, facilitating automated testing and data collection. In summary, Vue 2.0 ensures technical compatibility while providing exceptional runtime flexibility and reproducibility for experimental design. Before each test, clear the browser cache or use no-trace browsing to ensure that the initial state of the page is the same for each test, and close other unnecessary processes and applications to minimize the interference of external factors during the test.

Data Collection And Processing Methods
All pages perform the following standardized operational procedures in the test to ensure consistency and comparability.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

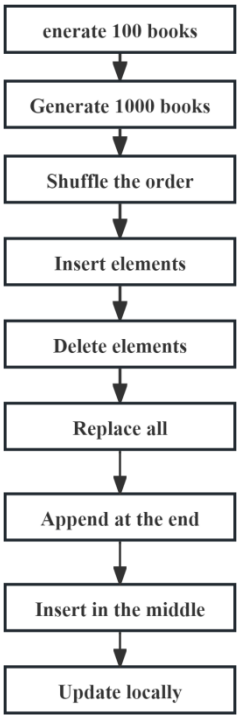

Figure 7: Unified Operation Sequence For DOM Performance Evaluation

Scripts are run in bulk and sampled in multiple rounds via Vue + JS + Performance API, and each combination of strategy and scenario is repeated 10 times to ensure data stability. Definition of performance indicators and collection methods are shown in Table 6.

Table 6: Performance Metrics And Data Collection Methods

Metric	Definition	Collection Method
DOM Rendering Time	Time taken to re-render the page after a state change	Performance API (performance.mark,performance. measure,performance.now）
FCP (First Contentful Paint)	Time when the first text or image is painted on screen	Lighthouse
LCP (Largest Contentful Paint)	Time when the largest visible content element is fully rendered	Lighthouse
TBT (Total Blocking Time)	Measures the total blocking time impacting responsiveness	Lighthouse

Since some performance metrics are “smaller is better” attributes (e.g., DOM rendering time), the values of these metrics are converted to values between 0 and 1 when using the Min-Max normalization method. However, if min-max normalization is used directly, performance metrics with smaller values receive lower scores. To solve this problem, we use inverse normalization to ensure that “smaller is better” metrics reflect their actual performance benefits. To solve this problem, we use inverse normalization to ensure that the “smaller is better” metric reflects its actual performance advantage. Perform weight setting is displayed in Table 7.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Table 7: Performance Metrics and Data Collection Methods

Metric	Weight
DOM Rendering Time	40%
FCP	20%
LCP	20%
TBT	20%

In this paper, when evaluating the performance of the virtual DOM Diff in the Vue.js framework, based on the current front-end performance research literature and practical experience, we assign weights to each performance indicator, which reflects the relative importance of different indicators in the evaluation of front-end performance, especially the impact on user experience. DOM rendering time is given the highest weight due to its direct impact on the initial loading experience of the user, while FCP, LCP, and TBT are evaluated in terms of page loading visibility, main content presentation, and interactive performance, respectively, which rationally assigns the weights of each metric to provide a more scientific and accurate performance evaluation. Constructing a linear weighting model for performance scoring.

$$Total_n = 0.4 * DOM_n + 0.2 * FCP_n + 0.2 * LCP_n + 0.2 * TBT_n$$

Where Totaln is the total performance score of the nth test, a weighted scoring model is used to comprehensively evaluate each performance metric, where each metric is assigned, a different weight based on its impact on the user experience. For example, DOM rendering time directly affects the initial loading speed of a page, so it is given a weight of 40%. FCP, LCP, and TBT represent the first rendering of the page, the loading time of the main content, and the interactive performance, respectively, so their weights are 20% each.

Comparison Of Performance Scoring Results For Each Strategy

Figure 8 presents a comparative analysis of the total performance scores for the three key attribute usages—unique ID, index, and none—across ten iterations in a simple page scenario. The visualization distinctly reveals stratified performance levels, with the unique ID strategy consistently occupying the top tier, index showing moderate stability, and none markedly underperforming, thereby highlighting clear trends in relative effectiveness. As can be seen, the unique ID attribute offers the best stability and highest score in this scenario.

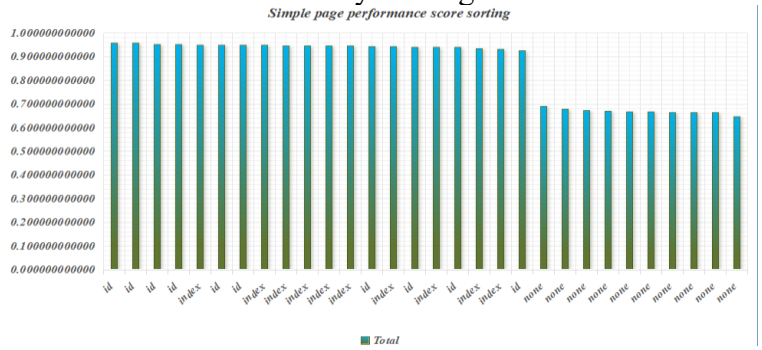

Figure 8: Total Performance Score Comparison On Simple Page Across key Attribute Usage

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Figure 9 compares the overall performance of three key attribute usage—unique ID, index, and none—based on ten iterations in a high-volume data scenario. The visual distribution reveals statistically distinguishable trends in strategy effectiveness, with the unique ID strategy maintaining superior stability and the none strategy lagging significantly, a unique ID is significantly better than index and none.

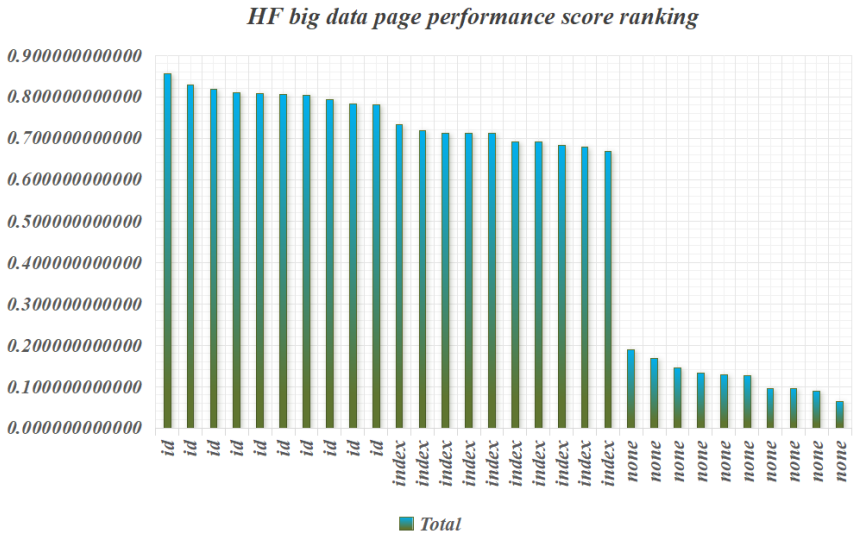

Figure 9: Total Performance Score Comparison On HF Big Data Page Across Key Attribute Usage

Figure 10 illustrates a clear performance hierarchy among the three key attribute usages tested across ten iterations on a nested component page. The consistently superior scores of the unique ID strategy, the moderate but stable results of the index strategy, and the marked underperformance of the none strategy reflect statistically meaningful stratification in DOM diff efficiency under deep structural complexity, unique IDs still have a slight advantage.

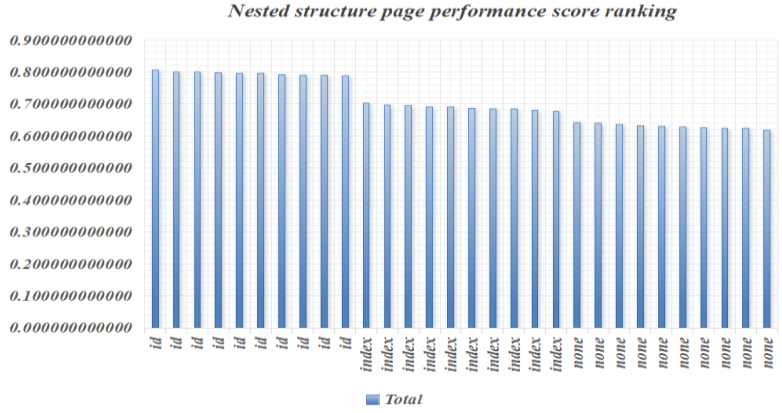

Figure 10: Total Performance Score Comparison On HF Big Data Page Across Key Attribute Usage

Comparison Of Scores And Results By Strategy

By comparing the performance scores of each strategy across multiple test scenarios, we can visualize how each strategy performs under key metrics such as DOM rendering, FCP, LCP, TBT, and so on in the war room. From figure 9, you can see the comparison of the mean values

Journal of Information System and Technology Management
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

of the three strategies (unique ID, index, and none) for different performance metrics, including Total, Dom_norm, FCP_norm, and TBT_norm.

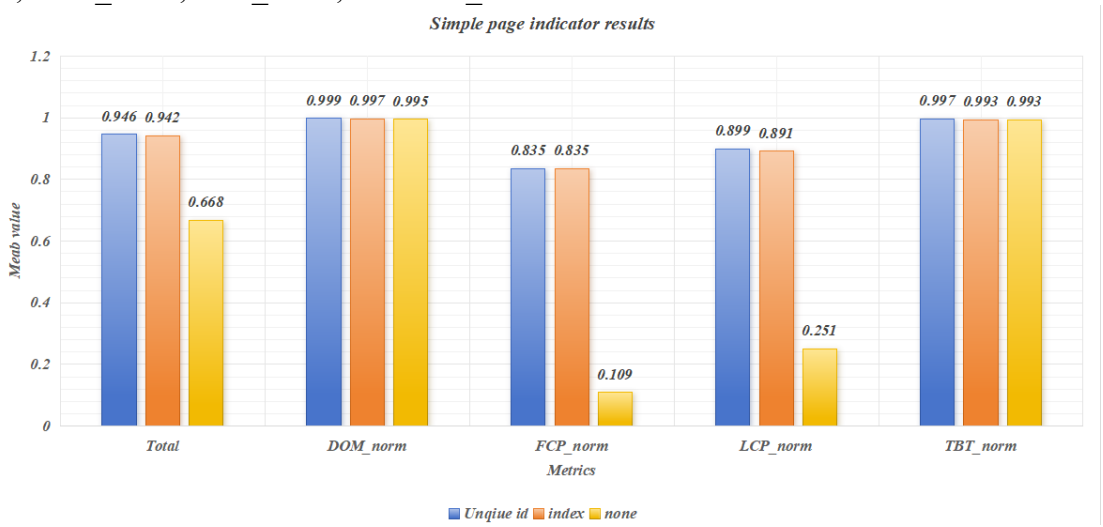

Figure 11: Strategy Performance Interpretation On Simple Page By Metric Dimensions

From figure 11, we can see the performance comparison of the three strategies (unique ID, index, and none) on HF big data pages. The icon shows the comparison of the mean values under different performance metrics (including Total, Dom_norm, FCP_norm, and TBT_norm)

Figure 12: Normalized Performance Metric Comparison of Key Attribute Usage On HF Big Data Page

From figure 13, we can see the performance comparison of the three strategies (unique ID, index, and none) on nested structure pages. The icon shows the comparison of mean values with different performance metrics (including Total, Dom_norm, FCP_norm, and TBT_norm)

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

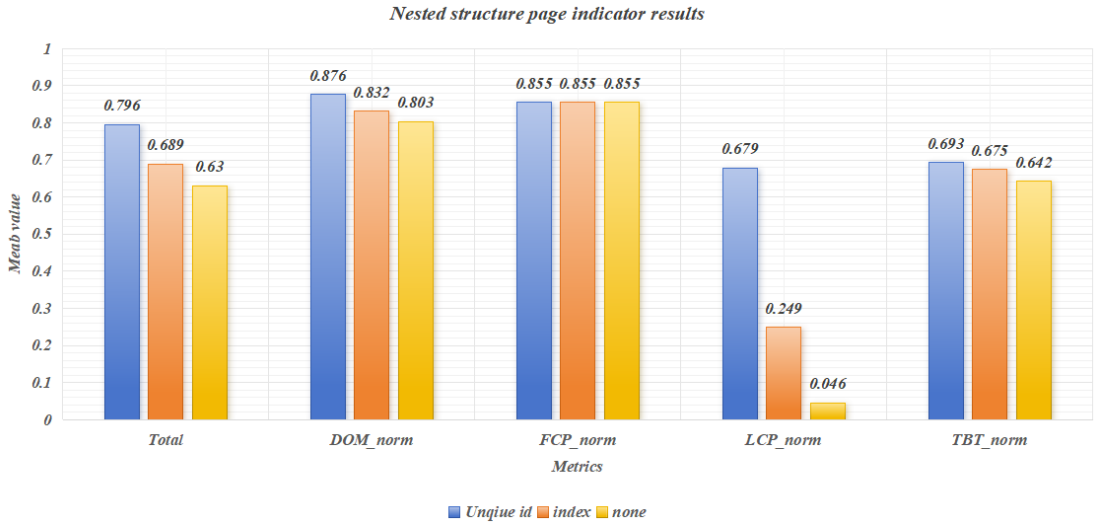

Figure 13: Normalized Performance Metric Comparison Of Key Attribute Usage On Nested Structure Page

Comparison of normalized scores for each metric (DOM rendering time, FCP, LCP, TBT) in three-page scenarios, further demonstrating the comprehensive performance of the unique ID strategy in key performance dimensions. Performance Optimization Recommendations: For nested pages, it is recommended to use the unique ID policy, especially in applications with high requirements for page loading efficiency and interactive performance. Index policy is a suboptimal choice for scenarios that need to balance performance and rendering efficiency. None policy should be avoided in complex pages or nested scenarios. Under three typical test scenarios of simple page, HF big data page, and nested structure page, the performance of the three strategies (unique ID, index, and none) in multiple core metrics (including total score, DOM rendering time, FCP norm, LCP_norm, and TBT_norm) shows a systematic pattern and hierarchical distribution. Comprehensive icon and index data show that a unique ID strategy always maintains the highest performance score and stability among the three types of pages, especially in DOM rendering and TBT performance, which is always better than the other two strategies, suitable for high-response, low-latency, frequent user interaction web page environments, with good cross-scene adaptation capabilities.

Although the overall performance of Index is slightly lower than that of the unique ID policy, it still maintains a high level in most of the indicators, especially in simple pages and nested pages, which is practical to a certain extent. In contrast, the none strategy generally has the lowest performance scores, especially in key indicators such as FCP and TBT, which are significantly lower than the previous two strategies, especially in high-frequency big data pages and nested structure pages, reflecting its poor applicability in modern page rendering and loading optimization.

Based on the above analysis, it is recommended to use the unique ID strategy to optimize page performance, especially in scenarios that require efficient page rendering and fast user interaction, and the index strategy is also a feasible alternative for scenarios with moderate performance requirements, while the None strategy is only considered to be used in resource-constrained scenarios or those with special compatibility requirements. This conclusion lays

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

the data foundation for subsequent multi-factor significance analysis and optimization recommendations

Analysis Of Variance (ANOVA)

To verify whether the performance differences of the three strategies are significant under different page scenarios, this paper uses single-factor analysis of variance (ANOVA) for statistical testing, with the following applicability notes: Applicable for comparisons of three or more means; the experimental groups are independent, and the variance of the data within each group is relatively consistent; the data satisfies the conditions of normal distribution or large sample size. The experimental results show that the F-value is as high as 1932.846, and the p-value is far less than 0.001, indicating that the performance differences between strategies are highly statistically significant.

Table 8 shows that he significance of the component is .000, which is much less than the level of significance commonly used (0.05, 0.001), indicating that there is a highly significant difference in the performance scores among the three groups of strategy and page combinations, and the results are highly statistically significant. F=1932.846 is very high, indicating that the mean difference of components is much larger than the fluctuation difference within the group, which means that there is a great difference in the performance under different combinations of strategies and pages, which also indirectly reflects that the “strategy design” has a very significant impact on the performance of the page, and it is a key influence factor. The within-group sum of squares is only 0.025, indicating that the sample scores within each group are stable, which means that the experimental data have good internal consistency and reproducibility.

Table 8: One-Way ANOVA Results For Total Performance Scores Across Key Attribute Usage

	Sum of squares	df	Mean Square	F	Significance
Between Groups	4.788	8	.598	1932.846	.000
Within Groups	.025	81	.000		
Total	4.813	89			

In this study, the strong significant effect of different key attribute usage on the page performance metric Total is verified by one-way ANOVA with very high F-value and low P-value, which proves that the difference in performance among the three types of strategies under the three-page complexity conditions is statistically significant and has practical engineering significance. It should be noted that the overall score of the "none" strategy is significantly lower than that of the "unique ID" and "index" strategies among the three types of pages, while the "unique ID" strategy maintains a significant advantage in all pages, a finding that is highly consistent with the previous icon analysis and further strengthens the credibility from a statistical perspective. The present results provide a prerequisite for further multiple comparisons and provide data support and a theoretical basis for strategy selection in page structure optimization.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Significance Analysis of Page Performance Differences For Multiple Strategy Groups

To verify whether there is a statistically significant difference in the performance of different strategy combinations in various types of scenarios, this study systematically analyzes the page performance scores (the Total variable) by using one-way ANOVA and combining it with the multiple comparison's method at the Tukey time.

With the total page performance score (Total) as the dependent variable and strategy combinations as the independent variables, the analysis results show the following: (1) Component difference is significant, F=1932.486, p<0.001,indicating that there is at least one set of statistically significant differences in page performance scores between strategy combinations; (2) The component sum of squares (SS_between) is 4.788, which accounts for the majority of the total sum of squares (SS_total=4.813), indicating that strategy combinations are the key factor influencing page performance scores.

In order to clarify those components that were significantly different, further Tukey HSD post hoc comparisons were performed with the alpha significance level set at 0.05, and the results were shown in Table 8. It shows that the unique ID policy consistently outperforms index and none, and the none policy is significantly worse than the other two. There is a significant hierarchical difference in policy performance scores across page types

Table 9: Pairwise Comparison of Strategy Combinations (Tukey HSD Test, p < 0.05)

Comparison Group (Code)	Mean	Mean Difference	Significance
11 vs 13	Simple page:unique ID vs nonne	0.277	Significance
21 vs 23	HF big data :unique ID vs none	0.685	Significance
31 vs 33	Nested Structure:unique ID vs none	0.506	Significance
22 vs 23	HF big data: index vs none	0.576	Significance
32 vs 33	Nested Structure:index vs none	0.059	Significance

Homogeneous Subset Analysis

In order to demonstrate the component significance relationship more intuitively, the subset analysis approach of Tukey HSD was used (alpha = 0.05). The results show that combinations 11 and 12 constitute a homogeneous subset of the optimal performance and are significantly higher than the other combinations in terms of the total performance average value of the rice pavilion, and the difference is not statistically significant (mean values are 0.9458 and 0.9424, Sig = 1.000, respectively). This indicates that the two combinations have minimal differences in performance and belong to the funny strategy combinations with similar performance, which are suitable for priority use in scenarios that require optimization of page loading and response speed. Combination 13 is individually categorized as a homogeneous subset with intermediate performance, with a mean value of 0.6685, which is significantly lower than the other two combinations, and the mean difference between it and combination 11 is 0.277, which is a significant difference (Sig=0.000), indicating that there is a statistically significant disadvantage in the performance of combination 13 compared to the other two combinations of the strategy combinations, and that it is suitable to be used in scenarios that do not have a high performance requirement but still require a certain amount of corresponding efficiency for use in application scenarios.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Combination 33 is categorized into the subset of low performance with a mean value of 0.6299, which is lower than all the combinations in the other subsets, and although the difference with combination 13 is not significant, the difference with combinations 11 and 12 has reached the level of significance, which indicates that there is a statistically significant disadvantage in the performance of combination 33, which is a non-recommended scenario. The remaining combinations, such as 21 and 31, are distributed in a number of different subsets, indicating inconsistencies in their performance. The difference between combination 21 and combination 12 has a value of 0.8084, Sig = 0.785, which is not significant, indicating that they have similar levels of performance. The difference between combination 22 and combination 23 has a value of 0.6996, Sig = 0.221, which is also not significant. The difference between combination 22 There is a significant difference (Sig < 0.05) between Combination 11 with a difference value of 0.6996, indicating a significant difference in performance. Combination 31 has a significant difference with Combination 33 with a difference value of 0.506 and Sig = 0.000. Combination 32 has a significant difference with Combination 33, with a difference value of 0.059, which has reached the level of significance between the subsets, indicating a significantly different level of performance.

Taken together, the results show that different combinations of page types and strategies have a significant impact on overall performance, with statistically significant differences between combinations and stratified performance. In summary, the results of the Tukey HSD multiple comparison analysis provide the following key findings: (1) There is a highly significant impact of policy combinations on page performance scores; (2) The overall score of the Unique ID policy combination is significantly higher than that of the Index and None policy combinations; (3) The None policy scores significantly lower than the other policies in most combinations, verifying its non-recommendation in practical applications; (4) Homogeneous subset analysis provides optimization suggestions for page structure and strategy adaptation; for example, index strategy is an acceptable performance compromise under simple page structure, while unique ID is the best choice for high-frequency big data pages.

Box Plot Analysis

The box plot in figure 14 presents the distribution of the performance indicator Total in a win-win SU combination, where: X-axis (combined group): is the combination number of each group of tests (11, 12, etc.); each number represents a set of unique combinations of page types and policy configurations. Y-axis (Total): the total performance scores, with the higher value indicating better performance. GROUP_A: represents the page type (High Frequency = High Frequency Big Data page, Simple = Simple page, Nested = Nested Structure page). GROUP_B: represents the three strategies (unique ID, index, none). Data points in the graph show the actual score distribution of each combination group.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

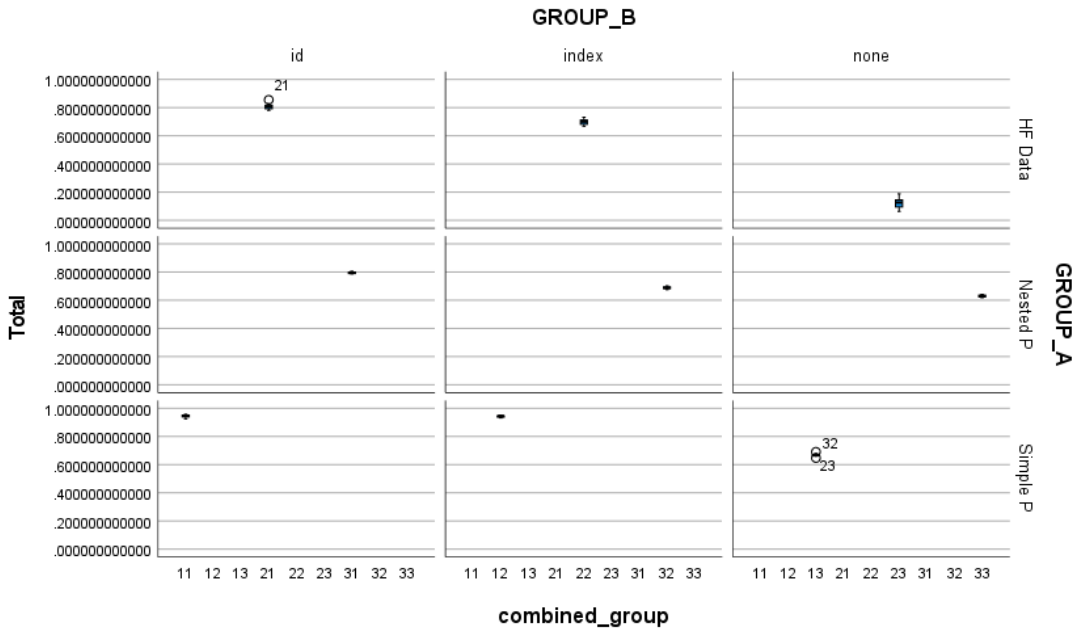

Figure 14: Tukey HSD Subset Grouping For Total Performance Scores

Unique ID strategy: the overall score of the most Karma, especially in high-frequency and simple page types, the overall score is close to 1, the optimal performance, and the distribution of the compact, indicating that the stability is good. Index strategy: the score is slightly lower than Unique ID but still maintains a high level; especially in the high-frequency and simple pages, the performance is good, and the fluctuation range is small. None strategy: the overall score is the lowest, especially in the simple page, which has a large distribution of dispersion, and even appeared less than 0.4 outliers, indicating that its performance in the absence of strategy intervention is unstable and the overall performance is poor. Simple page (middle row): shows strong differentiation, with a score close to 1 when the UNIQUE ID strategy is used, but the performance decreases dramatically under the NONE strategy, and there are multiple outliers, indicating that the page itself is not highly loaded and is sensitive to strategy differences. HF big data page (top row): performs better under all three strategies, with a centralized distribution and high scores for the UNIQUE ID and INDEX strategies, and a drop but no extreme values for none, indicating that it has a more robust impact on the strategies. Nested structure page (bottom row): the scores of the three strategies are generally low, especially under the none strategy; the scores are concentrated in 0.4-0.5, indicating that the nested structure is complex and the strategy has less influence on it, but the overall optimizable space is large.

The above figure shows that there is a significant interaction effect between page type and strategy type as follows: Under simple pages, strategy differences have the largest impact on performance, with unique ID significantly outperforming the other strategies, while the none strategy shows more outliers (e.g., 52, 52,56), with significant differences; Under HF big data pages, the performance gap between different strategies is relatively small, but the unique ID and index strategies are more stable than none; Under nested pages, the scores of all strategies are low, but unique ID is slightly better than index and none, reflecting that the role of the strategies is limited under highly structured and complex pages.

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020

Conclusion

In this paper, a systematic and quantitative experimental study is conducted around the key influencing factor of virtual DOM Diff performance in the Vue framework—key attribute usages The study takes three-level page scenarios (simple page, high-frequency big data page, and nested structure page) as the experimental basis, uses unique ID, index, and none strategies as independent variables, constructs an inverse normalization model to evaluate the overall performance, and verifies the significance of the differences between strategies with the help of single-factor analysis and the Tukey HSD test method.

The main findings of the study are as follows: (1) Key selection has a significant impact on rendering performance. Whether in static page or high-frequency big data page scenarios, different key attribute usages selection methods show significant differences in core metrics such as DOM rendering time, FCP, LCP, TBT, etc., and the ANOVA test shows that $p<0.001$, verifying that the key attribute usages is a key determinant of the virtual DOM's performance; (2) A unique ID strategy performs optimally in all scenarios. In the three types of page scenarios, the total performance score of the Unique ID strategy is better than index and none, and the score fluctuation range is the smallest, which indicates that it has the highest degree of refinement in DOM node matching, and it is suitable for the actual engineering scenarios with high performance requirements; (3) Index strategy performance is sub-optimal, with a certain adaptability. In the simple page and part of the nested structure of the page, scenarios will be strong stability, which is an alternative to Unique ID, especially in the rendering of structural rules; data fluctuations in the controllable page can be used as a compromise optimization scheme; (4) None strategy is significantly worse than the other two strategies in performance. In all scenarios, none performance appears to have lower performance scores, especially in high-frequency big data pages and nested pages. There is a serious performance bottleneck. The general cut method of the unique ID matching mechanism is not recommended to be used in the actual project. (5) Page complexity and update frequency have a significant moderating effect on key performance. In simple pages, the performance difference between different key attribute usages is obvious. However, in nested structure pages, the gap between attributes converges, indicating that the complexity of page structure weakens the dominant role of key attribute usage on performance to a certain extent.

Despite the milestones achieved in this research, the following deficiencies and limitations still exist, which need to be further improved in the subsequent work: (1) The amount of framework selection is limited, focusing only on the Vue 2.0 framework and not covering other popular frameworks such as Vue 3.0 and react; (2) The experimental environment is not desktop browser testing; for the introduction of mobile and other test scenarios, there are limitations in the scope of application; (3) The sample size is limited; the number of repetitions in each group is 10. Although it meets the basic statistical validation requirements, there is still a need to further expand the space of the sample size; (4The performance evaluation index dimension is slightly single.

Based on the above deficiencies, subsequent research can be expanded and deepened in the following aspects: (1) Cross-framework comparative study, expand this evaluation scheme to different virtual DOM implementation mechanisms such as Vue3.0, React, etc., and verify the performance consistency of key attribute usages in different architectures; (2) Introducing real business scenarios: structuring performance logs and user behavior data in enterprise-level Web systems, verifying the effect of the strategy through A/B testing, etc., to enhance the

Journal of Information System and Technology Management
JISTM
EISSN: 0128-1666
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020
practicality of the research. (3) Enrich the evaluation index system: introduce subjective experience evaluation indexes (e.g., TTI satisfaction, etc.) and system resource utilization indexes (e.g., CPU utilization) to achieve multi-dimensional comprehensive evaluation; (4) Build a reusable evaluation tool. This study provides a systematic paradigm for subsequent front-end performance engineering research on the basis of clarifying the performance law of the key attribute usages, and also provides a theoretical basis and data support for the formulation of real-world component rendering optimization strategies.

Acknowledgements
The authors would like to express their sincere gratitude to Universiti Teknologi Mara (UiTM) for their invaluable support and resources that made this research possible.

References
Barth, S., Ionita, D., & Hartel, P. (2022). Understanding online privacy—a systematic review of privacy visualizations and privacy by design guidelines. ACM Computing Surveys (CSUR), 55(3), 1-37.
Cai, D., Li, R., Hu, Z., Lu, J., Li, S., & Zhao, Y. (2024). A comprehensive overview of core modules in visual SLAM framework. Neurocomputing, 10(5), 127760.
Cen, H. D., & Nusantara, P. D. (2024). Enhancing User Interface Comprehensive Evaluation: Front‐End Development Frameworks and Best Practices. Asian J. Inf. Technol, 22, 1-10.
Ekpobimi, H. O., Kandekere, R. C., & Fasanmade, A. A. (2024a). Conceptual framework for enhancing front-end web performance: Strategies and best practices. Global Journal of Advanced Research and Reviews, 2(1), 099-107.
Ekpobimi, H. O., Kandekere, R. C., & Fasanmade, A. A. (2024b). The future of software development: Integrating AI and machine learning into front-end technologies. Global Journal of Advanced Research and Reviews, 2(1), 069-077.
He, L., Chen, Q., Pang, Y., Wang, M., Wu, Y., Liu, L., & Qiang, Z. (2024). An improved face attributes editing method based on DDIM. Scientific Reports, 14(1), 27154.
Jessup, E., Motter, P., Norris, B., & Sood, K. (2016). Performance-based numerical solver selection in the Lighthouse framework. SIAM Journal on Scientific Computing, 38(5), S750-S771.
Kanwal, S., Nawaz, S., Malik, M. K., & Nawaz, Z. (2021). A review of text-based recommendation systems. IEEE Access, 9, 31638-31661.
Li, R., Deng, W., Cheng, Y., Yuan, Z., Zhang, J., & Yuan, F. (2023). Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. arXiv preprint arXiv:2305.11700.
Li, X., Gao, J., Jia, P., Zhao, X., Wang, Y., Wang, W., Wang, Y., Wang, Y., Guo, H., & Tang, R. (2024). Scenario-wise rec: A multi-scenario recommendation benchmark. arXiv preprint arXiv:2412.17374.
Lu, N., Wang, B., Zhang, Y., Shi, W., & Esposito, C. (2021). NeuCheck: A more practical Ethereum smart contract security analysis tool. Software: Practice and Experience, 51(10), 2065-2084.
Ma, H. (2024, February). Research and Design of a B/S Platform Information Management System Based on Django and Vue Frameworks. In 2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS) (pp. 103-107). IEEE.

Journal of Information System and Technology Management
EISSN: 0128-1666
JISTM
Volume 10 Issue 40 (September 2025) PP. 286-308
DOI: 10.35631/JISTM.1040020
Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., & Payne, B. D. (2015). Evaluating computer intrusion detection systems: A survey of common practices. ACM Computing Surveys (CSUR), 48(1), 1-41.

Ollila, R., Mäkitalo, N., & Mikkonen, T. (2022). Modern web frameworks: A comparison of rendering performance. Journal of Web Engineering, 21(3), 789-813.

Rathinam, S. (2022, December). Analysis and Comparison of Different Frontend Frameworks. In International Conference on Applications and Techniques in Information Security (pp. 243-257). Singapore: Springer Nature Singapore.

Schwab, M., Saffo, D., Bond, N., Sinha, S., Dunne, C., Huang, J., Tompkin, J., & Borkin, M. A. (2021). Scalable scalable vector graphics: Automatic translation of interactive svgs to a multithread vdom for fast rendering. IEEE transactions on Visualization and Computer Graphics, 28(9), 3219-3234.

Simões, B., del Puy Carretero, M., Martínez, J., Muñoz, S., & Alcain, N. (2024). Implementing Digital Twins via micro-frontends, micro-services, and web 3D. Computers & Graphics, 121, 103946.