UDC 004.738.5
DOI https://doi.org/10.36994/2788-5518-2025-01-09-12

INVESTIGATING THE PERFORMANCE IMPACT OF LAZY LOADING IN WEB APPLICATIONS

Borovskova Ye. A., *master's degree, Software Engineer, AppsFlyer Ltd, Kyiv, Ukraine. https://orcid.org/0009-0008-5481-8090*

Abstract. *The study's relevance is due to the growing performance requirements of modern web applications that process significant amounts of data and provide a high user experience. Under these conditions, lazy loading is considered an effective approach to optimise resource utilisation, reduce response times and reduce the load on network and computing resources. At the same time, it has been established that, despite the obvious advantages, there are limitations associated with implementing this technology that require a detailed analysis.*

The study's purpose is to assess the impact of lazy loading on key performance indicators of web applications, identify its advantages and limitations, and develop recommendations for the effective implementation of this technology.

The study applied the methodology of experimental modelling of web applications in different scenarios. Two versions of the application were developed: one without lazy loading and the other with its implementation. The experiments were conducted in networks with different bandwidths (4G, 3G), and performance was evaluated based on indicators such as page load time, RAM consumption, and network resources.

The study found that lazy loading significantly reduces loading time (up to 50% on a 4G network) and reduces the amount of data transferred by almost 3.5 times. This significantly improves the user experience and optimises the use of server resources. At the same time, a number of limitations have been identified, such as dependence on the stability of the network connection, complexity of implementation, and potential problems with SEO indexing of dynamic content.

The conclusions emphasise the need to consider the application's specifics and users' needs when implementing lazy loading. In particular, it is important to ensure compatibility with search engines, adapt to weak network conditions, and consider accessibility requirements for users with disabilities.

Prospects for further research are identified in the development of approaches to optimise SEO indexing of dynamic content, ensure accessibility for all user categories, and integrate lazy loading into systems that handle critical data. Another interesting area is the analysis of device power consumption when using this technology.

Key words: *lazy loading; web applications; performance; optimization; dynamic loading; loading time; memory consumption; network resources; SEO; accessibility.*

ДОСЛІДЖЕННЯ ВПЛИВУ LAZY LOADING НА ПРОДУКТИВНІСТЬ ВЕБ-ЗАСТОСУНКІВ

Євгенія Боровскова, *магістр, інженер програмного забезпечення, AppsFlyer Ltd, Київ, Україна. https://orcid.org/0009-0008-5481-8090*

Анотація. *Актуальність дослідження зумовлена все більшими вимогами до продуктивності сучасних вебзастосунків, які опрацьовують значні обсяги даних і забезпечують високий рівень користувацького досвіду. У цих умовах використання lazy loading розглядається як дієвий підхід до оптимізації завантаження ресурсів, скорочення часу відгуку та зниження навантаження на мережеві й обчислювальні ресурси. Водночас установлено, що, попри очевидні переваги, упровадження цієї технології має низку обмежень, які потребують детального аналізу.*

Метою роботи є оцінювання впливу lazy loading на ключові показники продуктивності вебзастосунків, визначення його переваг та обмежень, а також розроблення рекомендацій щодо ефективного впровадження цієї технології.

Дослідження проводилося із застосуванням методології експериментального моделювання роботи вебзастосунків у різних сценаріях. Створено дві версії застосунку: одну – без використання lazy loading, а іншу – з його впровадженням. Експерименти проводили за умов мереж із різною пропускною здатністю (4G, 3G), а продуктивність оцінювали на основі таких показників, як час завантаження сторінок, використання оперативної пам'яті та споживання мережевих ресурсів.

У результаті дослідження встановлено, що lazy loading забезпечує значне скорочення часу завантаження сторінок (до 50% у мережі 4G) і зменшення обсягу переданих даних майже в 3,5 раза.

Це значно покращує користувацький досвід, оптимізує використання серверних ресурсів. Водночас виявлено низку обмежень, серед яких – залежність від стабільності мережевого з'єднання, складність реалізації та потенційні труднощі із SEO-індексацією динамічного контенту.

У висновках наголошено на важливості врахування специфіки застосунку й потреб користувачів під час упровадження lazy loading. Зокрема, значущим є забезпечення сумісності з пошуковими системами, адаптація до умов слабких мереж і врахування вимог доступності для користувачів з обмеженими можливостями.

Перспективи подальших досліджень включають розроблення підходів для оптимізації SEO-індексації динамічного контенту, забезпечення доступності для всіх категорій користувачів та інтеграції lazy loading у системи, що працюють із критичними даними. Також важливим напрямом є аналіз енергоспоживання пристроїв під час використання цієї технології.

Ключові слова: *lazy loading; вебзастосунки; продуктивність; оптимізація; динамічне завантаження; час завантаження; споживання пам'яті; мережеві ресурси; SEO; доступність.*

Introduction

Web applications that serve many users face the challenge of efficiently managing resources, lowering page loading speeds, and providing a quality user experience. As the amount of data loaded initially increases, difficulties arise due to long response times, network congestion, and low performance of users' devices. One of the most promising approaches to solving these problems is using lazy loading technology, which allows you to optimize the downloading process by delaying its loading.

This problem is of great scientific importance, as it requires research into the impact of different loading strategies on the performance of web applications, considering the specifics of modern frameworks, browsers, and network protocols. The practical significance lies in increasing the efficiency of web applications, reducing the load on servers, and improving the user experience, which is critical for developers, online service providers, and end users.

The topic of researching the impact of lazy loading on web application performance is actively developing. Many scientific works focus on analyzing the principles of this technology, evaluating its effectiveness, and practical approaches to implementation.

The works of R.-M. Bâra, C. A. Boiangiu, and C. Tudose investigate the impact of lazy loading on key performance indicators of web applications, including page load time and memory usage. The authors consider implementation scenarios and offer recommendations for their optimization [1]. Similar aspects are analyzed by S. Turcotte, Gokhale, and F. Tip, who studied how lazy loading can increase the responsiveness of real-time web applications by reducing delays [2].

In their work, E. Mjelde and A. L. Opdahl investigate methods for reducing the loading time of web applications on different devices, focusing on devices with limited resources. Their analysis covers lazy loading as one of the key elements of web application performance [3]. S. K. Shivakumar takes a closer look at the optimization of mobile web applications, particularly the impact of lazy loading on saving network resources and reducing data volume [4].

The study by F. Yan, Z. Xu, and their colleagues aims to optimise front-end development performance. The authors propose a solution to improve the efficiency of dynamic web applications using lazy loading [5]. Wickham M. focuses on the practical aspects of implementing lazy loading for images, which is critical for reducing the loading time of large pages [6].

I. Shvedov investigates the feasibility of using lazy loading in web applications, emphasising its role in improving speed and reducing computing resources [7]. I. Khlysta and A. Bondarchuk consider the problem of partial content updating in SPA applications, using lazy loading as a key approach [8].

A. V. Antonenko and his colleagues describe innovative methods of implementing this technology in their work. The authors analyze the peculiarities of displaying information in web browsers and the role of lazy loading in reducing server load [9].

M. Hajian studies the use of lazy loading in the performance of Angular applications, focusing on improving the performance of large dynamic systems [10]. O. Kostenko and V. Furashev investigate the regulatory aspects of web space development, including technological challenges and their impact on users [11].

K. Chyzhmar and his co-authors analyse information security in the context of web application development, paying attention to the role of optimisation, in particular lazy loading [12]. F. Al-Hawari, in his work, considers design patterns for data management in web systems, highlighting lazy loading as an element of big data management [13].

In his book about progressive web applications, S. Domes demonstrates practical cases of using lazy loading in React applications to achieve loading speed and interactivity [14].

Thus, the analyzed studies demonstrate significant interest in implementing lazy loading as a key tool for optimizing web applications.

Despite studying the principles of lazy loading and its impact on performance, several key aspects still need to be solved. The conditions for using this technology in real-world web applications with complex

architectures require further research, especially in cases with high server loads or poor network quality (for example, in 3G networks). A limited number of studies analyze in detail how lazy loading affects the power consumption of mobile devices or performance in multimodule systems.

Special attention is drawn to the problems of SEO indexing dynamic content. Search engines are often unable to fully index pages with delayed loading, which can negatively affect websites' visibility in search results. Approaches to ensuring the accessibility of dynamic content for users with disabilities are also insufficiently developed, which is critical in terms of meeting modern standards of inclusiveness.

The proposed research will contribute to solving these problems. Experimental analysis will allow us to assess the impact of lazy loading in complex practical scenarios, particularly in conditions of low network speeds. It will also develop recommendations for optimizing SEO indexing of dynamic content and ensuring accessibility, which will help adapt the technology to modern web development needs. This will not only broaden the understanding of lazy loading capabilities but also help eliminate existing limitations for its effective implementation.

The article aims to investigate the impact of lazy loading technology on web application performance, identify its advantages and limitations, and provide practical recommendations for implementation to optimize the user experience and resource efficiency.

Objectives of the article:

1. To investigate the principles of lazy loading and assess its impact on the performance of web applications, including page load time, memory, and network resource consumption.
2. To conduct an experimental study of the implementation of lazy loading in various web application scenarios to identify the main advantages and limitations of its use in practical conditions.
3. To develop recommendations for the effective implementation of lazy loading in web applications based on the analysis of the results obtained and the identified limitations.

Research results

Lazy loading is an important approach in web development that allows you to optimize application performance by loading resources only when needed. The main principle of this technology is to reduce the initial load on the system by deferring the loading of modules, images, videos, or other components that are needed only at a specific moment of the user's work. This approach aims to improve page loading speeds, reduce network resource use, and optimize RAM consumption. In web applications that often contain large data or have a complex structure, lazy loading is a key tool for ensuring high performance (Table 1).

In today's environment, lazy loading is widely used to solve several problems related to the performance of web applications. For example, multi-page websites like e-commerce platforms avoid loading all pages simultaneously. Only those pages that the user accesses are loaded during the session. Also, delayed loading significantly reduces system response time on media platforms with a lot of video content.

Using attributes such as loading='lazy' allows the browser to control the loading process automatically. For example, in static websites, images or iframes are loaded only when they are in the user's visible area. This significantly reduces the consumption of network resources, especially on mobile devices.

Table 1

Basic principles of lazy loading in web applications

How it works	Explanation
Deferred loading of modules	Modules are loaded dynamically only when the user goes to a specific section or page.
Upload images on demand	Images are loaded only when the user scrolls to the appropriate area.
Using special attributes	The loading='lazy' HTML attribute allows the browser to automatically implement delayed image loading.
Dynamically connect scripts and styles	The download is performed using JavaScript depending on the user's actions.

In frameworks such as Angular, the lazy loading principle is implemented through modular connection mechanisms. This means that instead of loading the entire application on the first visit, only those modules that correspond to the functionality selected by the user are loaded. In React, similar functionality is achieved using libraries such as React-lazy and Suspense, which simplify the implementation of deferred component loading. As a result, the user gets faster access to content, load times are reduced, and developers ensure more efficient use of server resources. This approach is ideal for optimizing multifunctional platforms and websites that target a global audience.

As an approach to optimizing web applications, lazy loading affects key performance indicators, including page load time, RAM consumption, and network resource usage. This method reduces the load on the system by delaying the loading of content or modules only when needed. As a result, resources are optimized, the user experience is improved, and load times are reduced. Page load time is a critical metric because it affects the first impression of the user, as well as the overall performance of the application.

Using lazy loading allows you to download only the necessary data, significantly reducing the amount of traffic, especially in conditions of weak network connections (Table 2).

Today, lazy loading is proving to be effective in various industries, such as e-commerce, education, media, and entertainment. For example, in e-commerce, stores that use lazy loading of product images ensure that the first screen is displayed quickly, even if the catalog contains thousands of items. The user sees relevant content without delay, which increases conversion and reduces bounce rates [2]. In educational platforms that include interactive tasks or multimedia content, lazy loading allows you to load only active modules. This provides quick access to learning materials, reducing memory and time consumption. For example, video tutorials in eLearning platforms load only when the corresponding module is opened.

In media applications that focus on streaming video or displaying galleries, delayed loading allows you to optimize your traffic consumption. This is critical for mobile users with limited data. It allows consumers to get quick access to the content they need, avoiding network or device overload.

Table 2

Impact of lazy loading on key web application performance indicators

Performance indicator	**Explanation of the impact without Lazy Loading**	**Explanation of the impact with Lazy Loading**
Page load time	All resources are loaded simultaneously, which leads to longer loading times, especially with large amounts of content and multimedia data.	Only the necessary elements are loaded at the time of use, which significantly reduces loading time, especially for the first screen and on weak devices.
Amount of data transferred	The need to load all resources (images, videos, modules) regardless of their relevance to the current user session.	Only the data that is needed at a particular moment is transmitted, reducing server load and traffic consumption.
RAM usage	High memory requirements due to the simultaneous loading of all modules, which can cause delays or crashes on low performance devices.	Reduced load due to gradual data loading, which optimises memory usage and reduces the risk of overloading devices.
Network performance	Significant delays due to the transfer of large amounts of data in a single session, which is especially problematic for low-speed networks or limited mobile connection resources.	Optimised loading reduces latency, improving the user experience even on low-bandwidth networks.
Power consumption of devices	Increased power consumption due to high processor and memory requirements, which reduces the battery life of mobile devices.	Rational energy consumption by reducing the amount of computing and optimising the use of resources.

Lazy loading is becoming an important tool in improving the performance of web applications, allowing not only to adapt the operation of applications to the needs of users, but also to ensure the efficient use of computing and network resources. Its implementation is strategically important for modern developments focused on interactivity and a high level of user experience [1].

An experimental study of the implementation of lazy loading in web applications was conducted to assess its impact on key performance indicators in various scenarios. The main goal of the experiment was to analyze the loading time, RAM, and network resource efficiency in a web application environment, particularly in the example of multi-page websites built on the Angular framework.

The experimental conditions included simulation of a web application with 10 pages, where the probability of visiting all pages during one session is low. For comparison, two versions of the application were created: without using lazy loading and with its implementation. The metrics used were the initial amount of data downloaded, the time to first page display, and resource consumption under different network conditions (4G and 3G).

The use of lazy loading in this experiment was implemented through the mechanism of dynamic module loading using Angular. The code for routing was as follows: export const routes: Routes = [{ path: 'user', loadChildren: () => import('./pages/user/user.module').then(m => m.UserModule) }];

This approach allows you to download only those modules the user needs at a particular moment, avoiding system overload (Table 3).

The analysis of the results showed that lazy loading significantly reduces the time it takes to load the first screen and optimizes RAM consumption. For example, in a 4G environment, an application with lazy loading achieved a load time of 1 second, while without this approach, the time was about 3 seconds. On a 3G network, the results were even more impressive, showing a reduction in time from 5 to 1.5 seconds. In addition, the amount of downloaded data was reduced by almost 3.5 times, which helped reduce the server and network infrastructure load.

The experiment's practical results prove that lazy loading is an effective tool for optimising multi-page web applications. It provides faster access to content for users with different levels of access to resources, increasing the system's overall efficiency and adaptability. This approach is recommended for use in applications targeting a global audience, especially in a mobile environment with limited capabilities [10].

Table 3

Comparison of web application performance with and without lazy loading

Scenario of use	No Lazy Loading	With Lazy Loading
The initial volume of data	~2.5 MB	~700 KB
Download time (4G)	~3 seconds	~1 second
Download time (3G)	~5 seconds	~1.5 seconds
RAM usage	High due to simultaneous loading	Low due to the step-by-step loading process
User experience	Low due to long waiting times	High due to quick access

Lazy loading is a key web application optimization tool that provides significant performance benefits. However, its implementation is accompanied by several limitations that must be considered when implementing it in practice. This technology helps to improve the efficiency of web applications, especially multi-page platforms, by dynamically loading modules, resources, or content only when needed. This reduces the server load and optimizes the speed of displaying the first screens.

Among the main limitations of lazy loading is the increased complexity of implementation. To integrate the technology, developers must adapt the application architecture, ensuring the correct routing and management of dynamic module loading [6]. This can increase development time and costs, especially in large-scale projects with many components. There may also be difficulties in ensuring technology compatibility with different frameworks or third-party libraries, which requires additional resources for testing and debugging.

Another important limitation is the dependence on the stability of the network connection. Although lazy loading optimizes traffic consumption, real-time resource loading can create delays if the user's connection is slow or unstable. This is especially true for multimedia content, such as videos or large images, where delays can negatively impact the user experience.

Another limitation is the risk of deteriorating SEO (search engine optimization) performance [14]. Since most search engines index the content of pages during the first load, lazy loading can result in some dynamic content remaining unavailable for indexing. This can affect your site's search engine rankings, especially if the content is key to driving traffic.

You should also consider potential accessibility issues for users with disabilities. For example, dynamic loading may not consider specific requirements such as keyboard navigation or screen readers. This can create additional barriers for such users, reducing the accessibility of the web application.

Effective implementation of lazy loading in real-world web projects requires careful planning and adherence to several practical recommendations to ensure maximum performance and user experience. One key aspect is a preliminary analysis of the web application architecture to determine the most resource-intensive modules or components that should be loaded dynamically. This requires testing with performance monitoring tools, such as Lighthouse or WebPageTest, to identify bottlenecks in the content loading process.

An important step is to properly organize the routing. In multi-page applications, modules should be structured in such a way that only the necessary components are loaded according to the route selected by the user. This can be implemented through modern frameworks, such as Angular or React, which support dynamic import mechanisms. In particular, it is useful for Angular to use a modular loading mechanism, where each module is connected only when the corresponding route is activated. This approach minimizes the initial amount of uploaded data and reduces the server load.

To optimize multimedia content, such as images or videos, it is recommended to use built-in browser attributes, such as loading='lazy'` which allow you to automate the process of delayed loading. This is especially effective in cases where a significant part of the page contains large files that can affect the first rendering time. Additionally, for interactive elements such as videos or iframes, you need to take into account their impact on the loading time and ensure that they load only after user interaction.

Particular attention should be paid to optimisation for mobile devices that have limited resources. In this case, it is necessary to test the performance of lazy loading on low-speed networks, such as 3G, to ensure stable and fast loading even under difficult connection conditions. It is also recommended that backup loading mechanisms be implemented in the event of a failure of the main scenario [3].

An additional factor in successful implementation is ensuring compatibility with search engines. To do this, it is necessary to adhere to standards that guarantee the correct indexing of dynamic content. One of these approaches is the use of server-side rendering (SSR) [11], which ensures that the underlying content is loaded on the server, making it available to search engines even before lazy loading is activated.

To summarise, the effective implementation of lazy loading depends on the balance between performance, user experience, and technical capabilities of the application. It requires taking into account the architectural features of the project, testing in real conditions, and integrating modern performance monitoring tools. With the right approach, lazy loading becomes a powerful tool for optimizing web applications, ensuring their adaptability to various use cases.

Conclusions

The study of lazy loading's impact on web application performance has revealed its significant advantages, including optimizing page loading time, reducing the number of resources used, and improving the user experience. This technology is particularly effective for multi-page applications and applications that operate with large amounts of multimedia content, allowing for adaptability to different network environments. Experiments have confirmed that delayed loading reduces the amount of data required for the initial download and reduces the load on servers.

At the same time, several problems have been identified that limit the implementation of lazy loading in practical conditions. The main ones are the increased complexity of the technology implementation, dependence on the stability of the network connection, and potential risks of deterioration of SEO optimization due to dynamic content. In addition, the article identifies accessibility issues for users with disabilities that arise due to dynamic loading of resources.

Recommendations for implementing lazy loading in real projects include a thorough analysis of the application architecture, using modern frameworks for dynamic module connection, and adapting the technology to the conditions of low-speed networks. An important component is to ensure compatibility with search engines using techniques such as server-side rendering. To minimize risks, comprehensive testing at all stages of development is recommended.

Prospects for further research include a deeper analysis of methods for ensuring accessibility for all categories of users, the development of adaptive approaches to SEO indexing of dynamic content, and the study of the cost-effectiveness of implementing lazy loading in projects of different scales and specifics. Particular attention should be paid to integrating this technology into systems that handle critical data and analyzing its impact on the power consumption of modern mobile devices.

Bibliography

1. Bâra R.-M., Boiangiu C. A., Tudose C. Analysing the performance impacts of lazy loading in web applications. *Journal of Information Systems & Operations Management.* 2024. Vol. 18, № 1. P. 1–15. URL: https://web.rau.ro/websites/jisom/Vol.18%20No.1%20-%202024/JISOM%2018.1.pdf#page=9 (date of access: 13.12.2024).

2. Turcotte S., Gokhale, Tip F. Increasing the Responsiveness of Web Applications by Introducing Lazy Loading. *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Luxembourg, Luxembourg, 2023. P. 459–470. DOI: https://doi.org/10.1109/ASE56229.2023.00192 (date of access: 13.12.2024).

3. Mjelde E., Opdahl A. L. Load-Time Reduction Techniques for Device-Agnostic Web Sites. *Journal of Web Engineering.* 2017. Vol. 16, № 3–4. P. 311–346. URL: https://journals.riverpublishers.com/index.php/JWE/article/view/3291 (date of access: 13.12.2024).

4. Shivakumar S. K. Mobile Web Performance Optimization. *Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms.* Apress, Berkeley, CA. 2020. P. 79–103. DOI: https://doi.org/10.1007/978-1-4842-6528-4_4 (date of access: 13.12.2024).

5. Yan F., Xu Z., Zhong Y. S., HaiBei Z., Ge C. Y. Research on Performance Optimization Scheme for Web Front-End and Its Practice. *Liang Q., Wang W., Liu X., Na Z., Li X., Zhang B. (eds) Communications, Signal Processing, and Systems. Lecture Notes in Electrical Engineering.* Springer, Singapore, 2021. Vol. 654. P. 118–127. DOI: https://doi.org/10.1007/978-981-15-8411-4_118 (date of access: 13.12.2024).

6. Wickham M. Lazy Loading Images. Practical Android: 14 Complete Projects on Advanced Techniques and Approaches. Apress, Berkeley, CA, 2018. P. 47–84. DOI: https://doi.org/10.1007/978-1-4842-3333-7_3 (date of access: 13.12.2024).

7. Шведов І. Дослідження доцільності, методів та шляхів оптимізації програмного забезпечення задля пришвидшення роботи веб-застосунків. *Вісник Хмельницького національного університету. Серія «Технічні науки»*. 2024. Т. 337, № 3(2). С. 341–346. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/180 (дата звернення: 13.12.2024).

8. Хлиста І., Бондарчук А. Вирішення проблеми часткового оновлення вмісту веб-сторінки шляхом оптимізації параметрів SPA. *Комп'ютерне моделювання та інформаційні технології.* 2023. С. 286–291. URL: https://conf.nltu.edu.ua/index.php/conf1/article/view/87 (дата звернення: 13.12.2024).

9. Антоненко А. В., Балвак А. А., Цвик О. С., Ємелін Д. М., Гришковець Є. П. Інноваційні методи відображення інформації у веб-браузерах. *Таврійський науковий вісник. Серія «Технічні науки»*. 2024. № 1. С. 3–11. URL: https://journals.ksauniv.ks.ua/index.php/tech/article/view/534 (дата звернення: 13.12.2024).

10. Hajian M. App Shell and Angular Performance. Progressive Web Apps with Angular: Create Responsive, *Fast and Reliable PWAs Using Angular. Apress, Berkeley,* CA. 2019. P. 169–199. URL: https://books.google.com.ua/books?hl=uk&lr=&id=xdyZDwAAQBAJ&oi=fnd&pg=PR5&dq=Hajian,+M.+(2019).+App+Shell+and+Angular+Performance.+Progressive+Web+Apps+with+Angular&ots=0OsppNP85Z&sig=kdvUk8ZkTDu5MXOz5FZq4F1IuvY&redir_esc=y#v=onepage&q=Hajian%2C%20M.%20(2019).%20App%20Shell%20and%20Angular%20Performance.%20Progressive%20Web%20Apps%20with%20Angular&f=false (date of access: 13.12.2024).

11. Kostenko O., Furashev V. Genesis of Legal Regulation Web and the Model of the Electronic Jurisdiction of the Metaverse. *Bratislava Law Review.* 2022. Vol. 6, № 2. P. 21–36. DOI: https://doi.org/10.46282/blr.2022.6.2.316 (date of access: 13.12.2024).

12. State Information Security as a Challenge of Information and Computer Technology Development / K. Chyzhmar et al. *Journal of Security and Sustainability Issues*. 2020. P. 819–828. URL: https://www.researchgate.net/publication/340545387_STATE_INFORMATION_SECURITY_AS_A_CHALLENGE_OF_INFORMATION_AND_COMPUTER_TECHNOLOGY_DEVELOPMENT (date of access: 16.12.2024).

13. Al-Hawari F. Software Design Patterns for Data Management Features in Web-Based Information Systems. *Journal of King Saud University-Computer and Information Sciences.* 2022. Vol. 34, № 10. P. 10028–10043. DOI: https://doi.org/10.1016/j.jksuci.2022.10.003 (date of access: 13.12.2024).
14. Domes S. Progressive Web Apps with React: Create Lightning Fast Web Apps with Native Power Using React and Firebase. Packt Publishing Ltd, 2017. P. 1–15. URL: https://books.google.com.ua/books?hl=uk&lr=&id=8RhKDwAAQBAJ&oi=fnd&pg=PP1&dq=Lazy+Loading+Web+++Applications+book&ots=6nSjjsFCBr&sig=0xM-hTx8bOiXJjf8udSLQsgMp7c&redir_esc=y#v=onepage&q=Lazy%20Loading%20%20Web%20%20%20Applications%20book&f=false (date of access: 13.12.2024).
15. Uluca D. Angular for Enterprise-Ready Web Applications: Build and deliver production-grade and cloud-scale evergreen web apps with Angular 9 and beyond. Packt Publishing Ltd, 2020. URL: https://books.google.com.ua/books?hl=uk&lr=&id=9YLoDwAAQBAJ&oi=fnd&pg=PP1&dq=lazy+loading+web+book&ots=fEb5VAJQYQ&sig=2RI_jPUiCloJFNf7iP2-vD1YPsY&redir_esc=y#v=onepage&q=lazy%20loading%20web%20book&f=false (date of access: 13.12.2024).

References

1. Bâra, R.-M., Boiangiu, C. A., & Tudose, C. (2024). Analysing the performance impacts of lazy loading in web applications. *Journal of Information Systems & Operations* Management, 18 (1), 1–15. Retrieved from: https://web.rau.ro/websites/jisom/Vol.18%20No.1%20-%202024/JISOM%2018.1.pdf#page=9.
2. Turcotte, S., Gokhale, & Tip, F. (2023). Increasing the Responsiveness of Web Applications by Introducing Lazy Loading. *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg,* 459–470. DOI: https://doi.org/10.1109/ASE56229.2023.00192.
3. Mjelde, E., & Opdahl, A. L. (2017). Load-Time Reduction Techniques for Device-Agnostic Web Sites. *Journal of Web Engineering*, 16 (3–4), 311–346. Retrieved from: https://journals.riverpublishers.com/index.php/JWE/article/view/3291.
4. Shivakumar, S. K. (2020). Mobile Web Performance Optimization. *Modern Web Performance Optimization: Methods, Tools, and Patterns to Speed Up Digital Platforms,* Apress Berkeley, CA, 79–103. DOI: https://doi.org/10.1007/978-1-4842-6528-4.
5. Yan, F., Xu, Z., Zhong, Y. S., HaiBei, Z., & Ge, C. Y. (2021). Research on Performance Optimization Scheme for Web Front-End and Its Practice. In Liang, Q., Wang, W., Liu, X., Na, Z., Li, X., & Zhang, B. (Eds.), *Communications, Signal Processing, and Systems. Lecture Notes in Electrical Engineering* (Vol. 654, pp. 118–127). Springer, Singapore. DOI: https://doi.org/10.1007/978-981-15-8411-4_118.
6. Wickham, M. (2018). Lazy Loading Images. *Practical Android: 14 Complete Projects on Advanced Techniques and Approaches*, *Apress, Berkeley, CA*, 47–84. DOI: https://doi.org/10.1007/978-1-4842-3333-7_3.
7. Shvedov, I. (2024). Doslidzhennia dotsilnosti, metodiv ta shliakhiv optymizatsii prohramnoho zabezpechennia zadlia pryskhvydshennia roboty veb-zastosunkiv [Research on the feasibility, methods, and ways of optimizing software to accelerate web applications]. *Visnyk Khmelnytskoho Natsionalnoho Universytetu. Seriia: Tekhnichni Nauky – Bulletin of Khmelnytskyi National University. Series: Technical Sciences*, 337(3), 341–346. Retrieved from: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/180 [in Ukrainian].
8. Khlysta, I., & Bondarchuk, A. (2023). Vyrishennia problemy chastkovoho onovlennia vmistu veb-storinky shliakhom optymizatsii parametriv SPA [Solving the problem of partial web page content updates by optimizing SPA parameters]. *Kompiuterne Modeliuvannia ta Informatsiini Tekhnolohii – Computer Modeling and Information Technologies*, 286–291. Retrieved from: https://conf.nltu.edu.ua/index.php/conf1/article/view/87 [in Ukrainian].
9. Antonenko, A. V., Balvak, A. A., Tsvyk, O. S., Yemelin, D. M., & Hryshkovets, Y. P. (2024). Innovatsiini metody vidobrazhennia informatsii u veb-brauzerakh [Innovative methods of information display in web browsers]. *Tavriiskyi Naukovoho Visnyk. Seriia: Tekhnichni Nauky – Tavrian Scientific Bulletin. Series: Technical Sciences*, (1), 3–11. URL: https://journals.ksauniv.ks.ua/index.php/tech/article/view/534 [in Ukrainian].
10. Hajian, M. (2019). App Shell and Angular Performance. Progressive Web Apps with Angular: Create Responsive, Fast and Reliable PWAs Using Angular, Apress, 169–199. URL: https://books.google.com.ua/books?hl=uk&lr=&id=xdyZDwAAQBAJ&oi=fnd&pg=PR5&dq=Hajian,+M.+(2019).+App+Shell+and+Angular+Performance.+Progressive+Web+Apps+with+Angular&ots=0OsppNP85Z&sig=kdvUk8ZkTDu5MXOz5FZq4F1IuvY&redir_esc=y#v=onepage&q=Hajian%2C%20M.%20(2019).%20App%20Shell%20and%20Angular%20Performance.%20Progressive%20Web%20Apps%20with%20Angular&f=false.
11. Kostenko, O., & Furashev, V. (2022). Genesis of Legal Regulation Web and the Model of the Electronic Jurisdiction of the Metaverse. *Bratislava Law Review*, 6 (2), 21–36. DOI: https://doi.org/10.46282/blr.2022.6.2.316.
12. Chyzhmar, K., Dniprov, O., Korotiuk, O., Shapoval, R. V., & Sydorenko, O. (2020). State Information Security as a Challenge of Information and Computer Technology Development. *Journal of Security and Sustainability Issues*, 819–828. https://www.researchgate.net/publication/340545387_STATE_INFORMATION_SECURITY_AS_A_CHALLENGE_OF_INFORMATION_AND_COMPUTER_TECHNOLOGY_DEVELOPMENT.
13. Al-Hawari, F. (2022). Software Design Patterns for Data Management Features in Web-Based Information Systems. *Journal of King Saud University-Computer and Information Sciences*, *34*(10), 10028–10043. DOI: https://doi.org/10.1016/j.jksuci.2022.10.003.
14. Domes, S. (2017). *Progressive Web Apps with React: Create Lightning Fast Web Apps with Native Power Using React and Firebase*. Packt Publishing Ltd. Retrieved from: https://books.google.com.ua/books?hl=uk&lr=&id=8RhKDwAAQBAJ&oi=fnd&pg=PP1&dq=Lazy+Loading+Web+++Applications+book&ots=6nSjjsFCBr&sig=0xM-hTx8bOiXJjf8udSLQsgMp7c&redir_esc=y#v=onepage&q=Lazy%20Loading%20%20Web%20%20%20Applications%20book&f=false.
15. Uluca, D. (2020). *Angular for Enterprise-Ready Web Applications: Build and deliver production-grade and cloud-scale evergreen web apps with Angular 9 and beyond*. Packt Publishing Ltd. Retrieved from: https://books.google.com.ua/books?hl=uk&lr=&id=9YLoDwAAQBAJ&oi=fnd&pg=PP1&dq=lazy+loading+web+book&ots=fEb5VAJQYQ&sig=2RI_jPUiCloJFNf7iP2-vD1YPsY&redir_esc=y#v=onepage&q=lazy%20loading%20web%20book&f=false.

Дата першого надходження рукопису до видання: 19.05.2025
Дата прийнятого до друку рукопису після рецензування: 18.06.2025
Дата публікації: 25.07.2025