UDC: 004.4+004.774

DOI: 10.31673/2412-9070.2023.060812

I. A. KHLYSTA, Master's student;
O. O. SHEVCHENKO, Portgraduate student;
O. V. SENKOV, Candidate of Technical Sciences (Ph.D),
State University of Information and Communication Technology, Kyiv

MITIGATING PARTIAL CONTENT UPDATE ISSUES ON WEB PAGES THROUGH SPA PARAMETER OPTIMIZATION

The experience of recent years has shown that the number of active web applications developed according to the fundamental principles of single-page applications continues to grow steadily despite a decrease in the number of newly created ones. This indicates a shift in focus from the importance of developing anew to the necessity of maintaining, servicing, and optimizing the existing end product. Furthermore, with the advancement and improvement of web development technologies, the potential expectations of users increase, necessitating the provision of a worthy user experience and the enhancement of application competitiveness. A significant role in this process is played by the partial content update process on web pages, widely used in the development of single-page applications. Most modern JavaScript frameworks designed for client-side web development employ the «out-of-box» technique, thereby providing the developer with complete discretion in deciding on approaches to its application. However, the automatic use of the web page content update mechanism does not guarantee desired results and often leads to average or sometimes unsatisfactory values that could have been avoided. This article provides an analysis of the characteristics of the performance parameters of a single-page web application, explores alternative methods to solve the stated problem, investigates potential «problematic» areas of the client-side of applications, compares the functioning of content update mechanisms of the most popular JavaScript frameworks Angular and React.js, and formulates recommendations for optimizing the overall application performance. The article concludes with tables of Core Web Vitals metrics for evaluating the effectiveness of the conducted optimization. Finally, potential vectors for further research are proposed.

Keywords: partial content update; single-page web application; web application optimization parameters; client-side web development; React.js library; Angular framework.

Introduction

Problem Statement. Optimization is one of the key and most essential elements in the development and maintenance of any software, especially when it comes to web applications. Productivity, responsiveness, browser resource consumption, load, overall loading time, predictability, smooth rendering — these and many other aspects are relevant to the process of improving the functionality of a web page. In the modern world, such applications have become significantly more user-oriented than ever before, making them more complex and demanding both in terms of development and user expectations.

One aspect of this complexity and power applied in client development is the technique of partial content update of the application page. This technique is widely used today in almost all client-side technologies in web development and is perceived by users as something absolutely standard and entirely expected.

However, the problem with this technique lies in the fact that it requires conscious management and constant adjustment. While being a powerful tool with broad possibilities for displaying the user interface qualitatively, it carries risks for the smooth functioning of the build. Therefore, simply employing the technique does not guarantee quality functionality to the user.

If user expectations are not met, it can lead to discomfort in usage, and ultimately, the user may decide to stop visiting a website that does not meet their expectations. Improving performance can significantly impact the user experience, enriching it and consequently contributing to increased traffic to the electronic resource. Therefore, an effective solution to the problem of partial content update on a web page is to analyze the characteristics of web application development depending on the specific tool and compare the performance indicators of such an application before and after the optimization.

Analysis of Recent Research and Publications. To achieve the set goal, a thorough review of scientific sources related to the research topic was conducted. The most relevant works, in our opinion, include:

- Scott E. A. Jr. from the University of Southern Mississippi, who explores the functioning potential of the single-page web application update mechanism and provides a general analysis of the SPA concept in the context of web development.
- Beglerović V., Pirija L., Prazina I., and Okanović V. from the University of Sarajevo discuss issues related to detecting changes on web pages and the importance of comparing the similarity of web pages. They also examine the architecture of the system for detecting changes on websites, detailing various

types of changes that can occur on web pages and describing an algorithm for detecting changes in content and page structure. The authors provide a comparative analysis of various similarity parameters, emphasizing the need for an efficient architecture to change detection systems.

• M. Selakovic and M. Pradel from the Technical University of Darmstadt explain the general concept of WPO (Web Performance Optimization) and the fundamental methods of improving web page performance. According to their work, WPO involves optimizing the performance of page components such as HTML content, web components, page elements, and page resources.

• Van Riet J. and Malavolta I. from the Vrije Universiteit Amsterdam analyze the impact of individual parameters of the client-side of a web application on its performance. The researchers combine seemingly unrelated metrics, including programming language, web protocol, loading speed, and web page availability, to assess its responsiveness.

• Vesper from Harvard University describes the Google Page Experience algorithm and its role in measuring three crucial aspects of the user experience of a website, known as Core Web Vitals. The work explains how to accurately evaluate page performance indicators and improve them based on the principles of Web Vitals.

The Main Part

Under the techniques of partial content update of a web page, researchers understand a specific set of tools to respond to changes in the view of a static web page during user viewing and interaction. The process of implementing tools that can consistently support the standard functioning of the page, maximizing responsiveness to user requests through improved performance, while facilitating the technical maintenance of code and maintaining a corresponding quality standard, is never-ending in its scope.

According to a study by the Institute of Physics (IOP) in the UK conducted at the end of 2021, it is known that the complexity and multi-layered logic of websites and information platforms are constantly increasing. This is why ordinary static websites or their alternative dynamic solutions, especially those based on CMS, are no longer able to meet user needs. Time, quality, and speed of interaction with the client now play a key role for the modern user, approaching milli- if not micro-level indicators in the world of digital technologies [3].

This has become possible largely due to a breakthrough in client-side web development through the application of the Single Page Application (SPA) approach. SPA refers to a full-fledged client part of a web application that has only one separate page, serving as a shell for all other components of this web application based on JavaScript, CSS, and HTML, and consisting of separate standalone components. SPA is built on AJAX technology, which assists in the asynchronous transmission of data from the client to the server without the need to completely reload the entire page each time and is based on the principles of the asynchronous module definition (AMD) specification [4].

Since the majority of websites today contain a significant amount of repetitive content, whether it be headlines, descriptions, legal disclaimers, or (depending on the theme) actual products, they are mostly repeated either within a specific section or throughout the entire application. Avoiding this content duplication is impossible, but single-page applications leverage this repetition to their advantage through asynchrony and partial page updates.

Moreover, data transmission occurs in JSON format, which accelerates client-server communication and is supported by many programming languages. Once the data arrives from the server to the client, the latter displays changes as if dynamically rewriting the page view.

However, to fully experience the benefits of the SPA approach, it is worth analyzing in more detail what ultimately determines its performance. In science, performance parameters are often considered in the context of their optimization. Together, they collectively form the Web Performance Optimization (WPO) Framework. The WPO Framework is usually examined either in the context of project life cycle stages or at different architectural levels. Considering this, the following are the life cycle stages of the framework, including dependent parameters (table 1) [5].

The text discusses how to track parameters influencing performance and content updates on a webpage using a table, categorizing them into two groups:

1) those related to general optimization;

2) those dependent on the chosen development mechanism (framework, library, CMS), i.e., special parameters.

Among the general parameters, HTML content, media elements, assemblies, and configurations play a crucial role. The frequency and format of images, videos, and audio materials on websites indicate potential inefficiencies, correlating with unsatisfactory Largest Contentful Paint indicators. Statistics reveal that most websites use static photo materials in the following formats: JPEG (54,9%), PNG (28,2%), WEBP (10,7%), GIF (2,6%), SVG (2,4%), ICO (0,9%), AVIF (0,3%). Video and audio materials — WAV, MP3, MP4, WEBM, and OGG are usually used equally in terms of accessibility and sustainable development principles.

Table 1

SPA Optimization Parameters Source*

Architecture and Design	Implementation	Validation	Monitoring
Size and Power of Infrastructure: • Hardware Network • infrastructure • System software, Physical processor resources or the use of CDN and web server characteristics	Software Source Code: • General • Specialized For example — compression, caching, scalability, managing computation change processes, state persistence, or client-server communication	Analysis of the Load on Physical Resources	Core Web Vitals Metrics: Largest Contentful Paint, First Input Delay, Cumulative Layout Shift
Content Rendering in a Web Browser: • HTML content, images/resources, scripts, style sheets	Management and Storage of Static Files and Build Minification: • Media resources	Testing potential vulnerabilities	Analysis of Google PageSpeed Insights
Modeling Performance Strategy: • Defining performance parameters and the mechanism to influence them	Adaptation of «No-Code» Values: • Loading strategies • Authentication security, e.g. Lazy Loading		Utilization of the Dom Size Analyzer tool
Defining desired performance metric indicators: • Alignment with acceptable Core Web Vitals metrics	Application of selected performance optimization techniques		

* Developed by the author based on sources [2; 4; 7]

Most of these formats are outdated today. Indeed, GIF, PNG, and JPEG are the most popular formats, but WEBP and AVIF offer more efficient compression and other advantages. It is recommended to convert existing JPEG images to a less data-intensive WEBP format to reduce data transfer volume [2].

A similar situation exists with video and audio. For their display, it's better to simply store a local link to them, ensuring that they are loaded only when actively clicked.

Finally, «lazy loading» should be used for videos and audio that require JavaScript. As for fonts, currently, almost half of all websites use the WOFF2 format, and a quarter use the WOFF format. Thus, the use of data-efficient formats on websites is noticeably widespread, indicating a high level of font optimization. The most important thing when choosing fonts is to remember that they are to be stored locally and not loaded during the application load [2].

Although the HTML page code is automatically corrected for syntax errors and partially optimized by browsers, this does not negate the necessity and importance of writing clean and syntactically correct code. Firstly, it's important to remember the browser limitations when creating the DOM structure of the page. Generating more than 1500 objects with a depth of 32 nodes should be avoided, as it leads to increased loading time and an increase in the memory and energy consumption of the local browser. Secondly, for HTML code, every individual character is important, so it's crucial to pay attention to nesting levels, empty tags, indentation, and comments [6].

Optimization related to physical infrastructure is also among the general parameters. If a developer, for example, is not authorized to make decisions regarding the choice of hosting environment, they can still influence server caching features, the use of client-server data transfer standards, data consumption, and compression.

For instance, if the server supports Brotli compression or the HTTP/2 standard, it facilitates the application of effective optimization tools «out-of-the-box» simply by activating host settings. On the other hand, some hosts may not even provide the option to determine the data size on a web page, let alone support newer standards [2].

The empirical aspect of the research is more related to the special optimization parameters, which revolve around the application's source code and depend proportionally on the chosen development tools. For the client-side of the web application, this involves the React.js library and Angular framework, aimed at showcasing differences in approaches to solving the issue of partial content updates on web pages.

The main point of intersection between technology and libraries is how the latter leverage it. For this purpose, Angular has a built-in mechanism called Change Detection, responsible for comparing changes in the view of a page due to user interaction. We propose rejecting the default Change Detection in favor of Zone-Less Change Detection. At the very least, different change detection strategies should be applied. If the input data of a component remains unchanged, there is no sense in rechecking them unless there is asynchronous logic in the component.

The OnPush change detection strategy is an optimization strategy in Angular. With this strategy, change detection is triggered only when certain conditions are met, such as a change in the input property of the component or the occurrence of an event from that component.

In React, a similar mechanism exists called Virtual DOM, which, however, is not controlled by the developer and is fully protected and automated. To facilitate the comparison of page views, we add a key attribute when processing large data volumes, memoize the input data of components after their initial arrival using React.memo Higher Order Component (HOC) and Hooks useCallback and useMemo.

The next widely used aspect of optimization is the analysis of utilized resources. In this context, we suggest prioritizing the application of DevTools for profiling performance. Additionally, memory leak prevention is crucial through the use of Observables in conjunction with the @UntilDestroy decorator for Angular and handling subscriptions in useEffect for React. Mathematical calculations using nested data structures, which should be mutated to adhere to the principles of pure functions and not modify input data, hold particular significance.

Regarding architecture, our recommendation is to choose Feature-Slice and Modular approaches to ensure scalability, clarity, and accessibility. It's essential to break down the code into parts based on the Code Splitting principle and load them in chunks/packages as needed. The lazy loading and suspense mechanisms are ideal for this purpose. Keeping in mind the limitations of the DOM, carefully design the hierarchy of components, breaking them down into subcomponents.

Research Results. To assess how effective the application has become, we will use Core Web Vitals metrics. These metrics measure important aspects of user interaction with the web page and are based on three key indicators:

1. First Input Delay (FID): The time it takes for the application to respond to the first user interaction.

2. Largest Contentful Paint (LCP): The time it takes for the application to render the largest element on the page.

3. Cumulative Layout Shift (CLS): The quality of the display layout of the web page.

In addition to these indicators, we will also consider the Total Blocking Time and Speed Index. For comparison purposes, several applications based on React and Angular were created.

During the first iteration, an application with 100 elements containing images, videos, fonts, and a standard architecture was developed.

According to the initial measurements, the input data shown in the table 2.

Accordingly, the optimized indicators for the first iteration shown in the table 3.

During the second iteration, an application with a more complex architecture and content containing 1000 elements was developed. The input metrics appeared as follows (table 4).

Table 2

Initial Metrics Values for 100 Elements Source*

Metric	React.js	Angular
FID[ms]	220±25	240±12
LCP[s]	3,0±0,05	3,3±0,02
CLS[ms]	0,13±0,03	0,15±0,05
TBT[ms]	0,00±0,00	0,02±0,01
SI[s]	3,0±0,05	3,3±0,02

* Developed by the author based on own tests [1-7]

Table 3

Optimized Metrics Values for 100 Elements Source*

Metric	React.js	Angular
FID[ms]	85±10	97±7
LCP[s]	1,4±0,03	1,7±0,02
CLS[ms]	0,05±0,03	0,045±0,02
TBT[ms]	0,00±0,00	0,019±0,03
SI[s]	1,4±0,03	1,7±0,02

* Developed by the author based on own tests [1-7]

Table 4

Initial Metrics Values for 1000 Elements Source*

Metric	React.js	Angular
FID[ms]	260±16	280±10
LCP[s]	3,3±0,1	3,5±0,12
CLS[ms]	0,192±007,18	0,174±002,40
TBT[ms]	0,15±0,00	0,20±0,03
SI[s]	3,3±0,1	3,5±0,12

* Developed by the author based on own tests [1-7]

The optimized indicators appear as follows (table 5).

Table 5

Optimized Metrics Values for 1000 Elements Source*

Metric	React.js	Angular
FID[ms]	120±7	130±9
LCP[s]	1,8±0,08	2,0±0,06
CLS[ms]	0,13±0,03	0,15±0,05
TBT[ms]	0,09±0,05	0,1±0,09
SI[s]	1,8±0,08	2,0±0,06

* Developed by the author based on own tests [1-7]

Conclusions

The mechanism of partial content updates on a web page is a powerful tool for enhancing the user experience of the client-side of a web application. However, to fully leverage the advantages offered by this technique and provide the user with the highest quality experience, it is necessary to continually optimize the project. The study examined fundamental concepts related to the functioning of content updates on a web page in a single-page web application. It also analyzed parameters responsible for a high level of productivity and methods to influence them with the goal of improving Core Web Vitals metrics. Based on

the results of the conducted research, it has been established that adapting the analyzed parameters according to the discussed methods plays a significant role in balancing the application as a whole and is an excellent tool for enhancing the functionality of the content update mechanism to offer a user interface of a higher quality.

Reference

1. ***Detection*** *and Logging Changes in Web Pages / V. Beglerović, L. Pirija, I. Prazina, V. Okanović // 21st International Symposium INFOTEH-JAHORINA (INFOTEH). East Sarajevo, Bosnia and Herzegovina. 2022. P. 1–5.*

2. ***Beyer T.*** *Nachhaltige Websites. Praktischer Leitfaden zur Prüfung und Optimierung – mit zahlreichen Tool-Tipps und Programmcodes. Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature. 2023.*

3. ***Kornienko D. V.*** *The Single Page Application architecture when developing secure Web services // J. Phys. 2021. Conf. Ser. 2091 012065.*

4. ***Scott E. A. Jr.*** *SPA Design and Architecture: Understanding single-page web applications. Manning Publications Co. NY, 2015. USA.*

5. ***Selakovic M., Prade M.*** *Performance Issues and Optimizations in JavaScript: An Empirical Study // IEEE/ACM 38th International Conference on Software Engineering (ICSE), 2016. P. 61–72.*

6. ***Van Riet J., Malavolta I.*** *Client-side Performance of Web-based Applications: the State of the Art. 2019 [Online; accessed 12. Oct. 2023]. URL:*

https://jaspervanriet.nl/assets/literature_study.pdf.

7. ***Vesper.*** *Measuring time-to-interactivity for modern web pages // Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation, NSDI. USENIX Association. [Online; accessed 14. Nov. 2023] 2018. URL:*

https://www.cs.princeton.edu/~ravian/publications/vesper_nsdi18.pdf

І. А. Хлиста, О. О. Шевченко, О. В. Сеньков

ВИРІШЕННЯ ПРОБЛЕМИ ЧАСТКОВОГО ОНОВЛЕННЯ ВМІСТУ ВЕБСТОРІНКИ ЧЕРЕЗ ОПТИМІЗАЦІЮ ПАРАМЕТРІВ SPA

Досвід останніх років показав, що кількість активних вебзастосунків, розроблених відповідно до засадницьких принципів функціонування односторінкових застосунків, продовжує неухильно зростати попри зменшення кількості новостворених. Це свідчить про зміщення фокусу з важливості розроблення заново на потребу в підтриманні, обслуговуванні та оптимізації вже наявного кінцевого продукту. До того ж, із розвитком та вдосконаленням технологій веброзроблення зростають потенційні очікування користувачів та виникає потреба забезпечення гідного користувацького досвіду та підвищення конкурентоспроможності застосунків. Неабияку роль при цьому відіграє процес часткового оновлення вмісту вебсторінки, що широко застосовується під час розроблення односторінкових застосунків.

Більшість сучасних JavaScript фреймворків, призначених для клієнтської веброзробки послуговуються технікою за принципом «out-of-box», надаючи в такий спосіб розробнику цілковиту повноту прийняття рішень щодо підходів її застосування. Однак автоматичне використання механізму оновлення вмісту вебсторінки не гарантує бажаних результатів, а призводить лише до середньостатистичних, а подекуди незадовільних значень, яких можна було б уникнути.

У статті подано аналіз особливостей впливу параметрів продуктивності односторінкового вебзастосунку, розглянуто альтернативні методи розв'язання поставленої задачі, досліджено потенційні «проблемні» місця клієнтської частини застосунків, проведено порівняння функціонування механізмів оновлення вмісту вебсторінки найбільш популярних JavaScript фреймворків Angular та React.js, а також сформовано рекомендації для оптимізації роботи застосунку загалом.

Наприкінці наведено таблиці показників Core Web Vitals для оцінювання ефективності проведеної оптимізації. Запропоновано потенційні вектори подальших досліджень.

Ключові слова: часткове оновлення вмісту вебсторінки; односторінковий вебзастосунок; параметри оптимізації вебзастосунку; клієнтська веброзробка; бібліотека React.js; фреймворк Angular.