УДК 004.774.6 DOI: 10.31673/2412-4338.2023.041320

Бондарчук А.П., Хлиста І.А.
Державний університет інформаційно-комунікаційних технологій, Київ

ВИРІШЕННЯ ПРОБЛЕМИ ЧАСТКОВОГО ОНОВЛЕННЯ ВМІСТУ ВЕБ-СТОРІНКИ ШЛЯХОМ ОПТИМІЗАЦІЇ ПАРАМЕТРІВ SPA

***Анотація**: Досвід останніх років показав, що кількість активних веб-застосунків, розроблених відповідно до основоположних принципів функціонування односторінкових додатків продовжує неухильно зростати попри падіння кількості новостворених. Це свідчить про зміщення фокусу з важливості розробки заново на необхідність підтримання, обслуговування та оптимізації уже наявного кінцевого продукту. Більше того, із розвитком та вдосконаленням технологій веб-розробки зростають потенційні очікування користувачів та виникає потреба забезпечення гідного користувацького досвіду та підвищення конкурентоспроможності додатків. Неабияку роль при цьому відіграє процес часткового оновлення вмісту веб-сторінки, що широко застосовується під час розробки односторінкових застосунків.*

Більшість сучасних JavaScript фреймворків, призначених для клієнтської веб-розробки послуговуються технікою за принципом "out-of-box", надаючи таким чином розробнику цілковиту повноту прийняття рішень щодо підходів її застосування. Однак автоматичне використання механізму оновлення вмісту веб-сторінки не гарантує бажаних результатів, а призводить лише до середньостатистичних, а подекуди незадовільних значень, які можна було б уникнути.

У статті подано аналіз особливостей впливу параметрів продуктивності односторінкового веб-додатку, розглянуто альтернативні методи вирішення поставленої задачі, досліджено потенційні "проблемні" місця клієнтської частини застосунків, проведено порівняння функціонування механізмів оновлення вмісту веб-сторінки найбільш популярних JavaScript фреймворків Angular та React.js та сформовано рекомендації для оптимізації роботи додатку в цілому.

Наприкінці наводяться таблиці показників Core Web Vitals для оцінки ефективності проведеної оптимізації. Насамкінець запропоновані потенційні вектори подальших досліджень.

***Ключові слова:** часткове оновлення вмісту веб-сторінки, односторінковий веб-додаток, параметри оптимізації веб-додатку, клієнт веб-розробка, бібліотека React.js, фреймворк Angular, веб-сторінка, інформаційна технологія*

Bondarchuk A.P., Khlysta I.A.
State university of information and communication technologies, Kyiv

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

***Annotation**: 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, web page, information technology*

1. Постановка проблеми

Оптимізація - один із ключових та найбільш суттєвих елементів розробки та подальшого обслуговування будь-якої програми, а тим паче, коли мова йде про веб-додатки. Продуктивність, оперативність реагування, споживання ресурсів браузера, навантаження, загальний час завантаження, передбачуваність, плавне відображення - ці та багато інших аспектів стосуються процесу поліпшення функціонування веб-сторінки. У сучасному світі такі додатки стали значно більш орієнтованими на користувача ніж будь-коли раніше, а отже значно складнішими та вимогливішими як з точки зору розробки, так і з боку очікувань відвідувачів сайтів.

Одним із аспектів такої складності та водночас потужності, що застосовується в клієнтській розробці, є техніка часткового оновлення змісту сторінки додатку. Ця техніка сьогодні використовується майже в усіх технологіях клієнтської частини веб-розробки, та сприймається користувачами як щось абсолютно стандартне та цілком очікуване. Проте проблема цієї техніки полягає в тому, що вона потребує свідомого управління та постійного коригування, оскільки будучи потужним інструментом і даючи широкі можливості для якісного відображення користувацького інтерфейсу, вона несе з собою ризики для безперешкодного функціонування збірки. Тому просто послуговуватись технікою не означає гарантувати якісний функціонал користувачеві. Якщо очікування користувача не задовольнити, то в подальшому він може відчувати незручності в користуванні та врешті-решт відмовитись від відвідування веб-сайту, який не відповідає його очікуванням. Покращення продуктивності може значною мірою вплинути на користувацький досвід, збагачуючи його та як наслідок сприяючи збільшенню відвідуваності електронного ресурсу.

Тому ефективним рішенням проблеми часткового оновлення вмісту веб-сторінки є аналіз особливостей розробки веб-додатку в залежності від конкретного інструменту та порівняння показників ефективності функціонування такого додатку до проведення оптимізації та після.

2. Аналіз останніх досліджень та публікацій

Для досягнення поставленої мети було проведено ґрунтовний огляд наукових джерел, що стосуються теми дослідження. Найбільш релевантними на нашу думку є наробки:

- Scott E. A. Jr. з Університету Південної Міссісіпі, котрі розкривають потенціал функціонування механізму оновлення сторінки односторінкового веб-додатку та надають загальний аналіз поняття SPA в світлі веб-розробки.
- Beglerović V., Pirija L., Prazina I. and Okanović V з Університету Сараєва обговорюють питання, пов'язані з виявленням змін на веб-сторінках та важливості порівняння схожості веб-сторінок. Крім того, розглядається архітектура системи виявлення змін на веб-сайтах. Наводяться різні види змін, які можуть відбуватися на веб-сторінках, описується алгоритм для виявлення змін вмісту та структури сторінки. Автори наводять порівняльний аналіз різних параметрів схожості. При обробці частих змін на веб-сайтах важливо мати ефективну архітектуру для системи виявлення змін.
- M. Selakovic та M. Pradel з Технічного університету Дармштадту, де пояснюється загальна концепція WPO (з англ. - Оптимізація продуктивності веб-сторінок) та основні методи покращення продуктивності веб-сторінки. Згідно з працею, WPO включає в себе

оптимізацію продуктивності таких складових сторінки, як вміст HTML, веб-компоненти, елементи сторінки та ресурси сторінки і т.і.

- Van Riet J., Malavolta I. з Вільного університету Амстердама аналізують вплив окремих параметрів клієнтської частини веб-додатку на його продуктивність. Дослідники об'єднують на перший погляд зовсім непорівнювані показники, серед яких мова програмування, веб-протокол, швидкість завантаження та доступності веб-сторінки, показники її відгуку.
- Vesper з Гарвардського університету описує алгоритм Google Page Experience та його роль у вимірюванні трьох найважливіших аспектів користувацького досвіду веб-сайту, що складають Core Web Vitals. В роботі пояснюється, як правильно оцінити показники продуктивності сторінки та підвищити їх на основі принципів роботи Web Vitals.

3. Виклад основного матеріалу

Під техніками часткового оновлення вмісту веб-сторінки науковці розуміють певний набір засобів реагування на зміни виду (view) статичної веб-сторінки під час перегляду та взаємодії із нею користувача. Процес впровадження засобів, які б могли постійно підтримувати стандартне функціонування сторінки так, щоб з одного боку максимально ефективно відповідати на запити користувача завдяки кращій продуктивності, а з іншого боку дозволяти полегшити технічне обслуговування коду та зберігати відповідний стандарт якості, ніколи не досягне кінця.

Тільки згідно із дослідженням Британського Інституту Фізики (IOP), проведеним наприкінці 2021 року, відомо, що складність та багатошаровість логіки веб-сайтів та інформаційних платформ постійно зростають. Саме через це звичайні статичні веб-сайти чи їх альтернативні динамічні рішення, зокрема ті, що базуються на CMS, не в змозі більше задовольнити потреби користувачів. Ключову роль для сучасного користувача тепер відіграють час, якість та швидкість взаємодії з клієнтом, які в світі цифрових технологій дедалі більше наближаються до мілі-, якщо не мікро показників [3, ст. 2].

Значною мірою це стало можливо завдяки прориву у сфері клієнтської веб-розробки через застосування підходу Single Page Application (з англ. - односторінкового додатку). Під SPA розуміють повноцінну клієнтську частину веб-додатку, яка має лише одну окрему сторінку, що використовується ніби оболонка для всіх інших компонентів (складових) цього веб-додатку на основі JavaScript, CSS та HTML та складається із окремих самостійних компонентів. В основу SPA покладена технологія AJAX, яка служить асинхронній передачі даних від клієнта до сервера без необхідності повністю оновлювати всю сторінку заново щоразу та базується на принципах специфікації асинхронного визначення модулів (AMD). [4]

Оскільки більшість веб-сайтів сьогодні містить низку повторюваного матеріалу, будь то заголовки, описи, правові положення чи (в залежності від тематики) безпосередньо товари, здебільшого вони постійно повторюються або в одному конкретному розділі, або в цілому по всьому додатку. Уникнути цього контенту неможливо, але односторінкові додатки використовують це повторення завдяки асинхронності та частковому оновленню сторінки на свою користь.

До того ж передача даних здійснюється у форматі JSON, що сприяє прискоренню клієнт-серверної комунікації та підтримується багатьма мовами програмування. Як тільки дані надходять від серверу до клієнту, останній відображає зміни ніби динамічно перезаписуючи вид сторінки.

Однак для того, щоб повною мірою відчути переваги підходу SPA, варто більш детально проаналізувати те, від чого врешті-решт залежить його продуктивність. В науці параметри продуктивності часто розглядають в контексті їх оптимізації. В сукупності вони разом формують Framework WPO (з англ. - рамка оптимізації продуктивності веб-додатку). Здебільшого Framework WPO розглядається або в розрізі етапів життєвого циклу проекту, або на різних архітектурних рівнях. З огляду на це відрізняють такі етапи життєвого циклу фреймворку включно із залежними параметрами [5].

Таблиця 1

Параметри оптимізації SPA

Архітектура та дизайн	Імплементація	Валідація	Моніторинг
Розмір та потужність інфраструктури: ● апаратне забезпечення ● мережева інфраструктура ● системне програмне забезпечення Наприклад - наявні фізичні ресурси процесора або ж використання CDN чи характеристик веб-серверу	Вихідний код програмного забезпечення: ● загальні ● спеціальні Наприклад - компресія, кешування, масштабованість, управління процесом вичислення змін, збереження стану чи клієнт-серверної комунікації	Аналіз навантаження на фізичні ресурси	Метрики Core Web Vitals: Largest Contentful Paint, First Input Delay, Cumulative Layout Shift
Змістове наповнення сторінки у веб-браузері: ● HTML-контент, зображення/ресурси, сценарії, стильові таблиці	Управління та збереження статичних файлів та мініфікація збірки: ● Медіа ресурси	Тестування можливих “вразливостей”	Аналіз Google PageSpeed Insights
Моделювання стратегії продуктивності: ● Визначення параметрів продуктивності та механізму впливу на них	Адаптація “не-кодових” значень: ● Стратегії завантаження ● Безпека автентифікації Наприклад Lazy loading		Застосування інструменту Dom Size Analyzer
Визначення бажаних показників метрик продуктивності ● Погодження допустимих метрик Core Web Vitals	Застосування обраних технік оптимізації продуктивності		

Джерело: розроблено автором на основі джерел [2,4,7]

Як можна прослідкувати з таблиці 1, параметри, які впливають на продуктивність і разом з цим на оновлення вмісту веб-сторінки дуже відрізняються, однак для кращого їх порівняння об’єднаємо їх в дві групи:

1. такі, що стосуються загальної оптимізації;
2. такі, що залежать від обраного механізму розробки (фреймворк, бібліотека, CMS), тобто - спеціальні.

Серед загальних параметрів найсуттєвішу роль відіграють HTML-контент, медіа елементи, збірки, налаштування. Тільки частота та формат зображень, відео та аудіо матеріалів, що використовуються на веб-сайтах, свідчить про неефективність використання ресурсів, що потенційно дорівнює незадовільним показникам Largest Contentful Paint. Статистика показує, що більшість веб-сайтів використовують статичні фотоматеріали в наступних форматах: JPEG (54,9%), PNG (28,2%), WEBP (10,7%), GIF (2,6%), SVG (2,4%), ICO (0,9%), AVIF (0,3%). Відео та аудіо-матеріали - WAV, MP3, MP4, WEBM та OGG використовуються як правило порівну з точки зору доступності та принципів сталого розвитку.

Більшість з цих форматів є застарілими на сьогодні. Дійсно GIF, PNG і JPEG є найбільш популярними форматами, але WEBP і AVIF пропонують більш ефективну компресію та інші переваги. Рекомендується конвертувати існуючі JPEG-зображення у менш витратний формат WEBP для зменшення передачі обсягу даних. [2, ст. 82, 83]

Схожою є ситуація з відео або аудіо. Для їхнього відображення краще просто зберігати локальне посилання на них, пересвідчуючись, що вони завантажуються лише при активному кліку на нього. Насамкінець, слід використовувати "ліниве завантаження" для відео та аудіо, що потребує JavaScript.

Що стосується шрифтів, то зараз майже половина усіх веб-сайтів використовує формат WOFF2, а чверть - формат WOFF. Таким чином, використання економних за обсягом даних форматів на веб-сайтах є помітно поширеним, що свідчить про високий рівень оптимізації шрифтів. Найголовніше при виборі шрифтів пам’ятати про те, що вони мають зберігатись локально і не завантажуватись щоразу, коли додаток завантажується. [2]

Хоча сам код HTML сторінки коригується на синтаксичні помилки та частково оптимізується браузерами, це не виключає необхідність та важливість написання чистого та синтаксично правильного коду. По-перше, важливо пам’ятати про обмеження браузера при створенні DOM-структури сторінки. Не варто генерувати більше ніж 1500 об’єктів глибиною в 32 вузли, адже це провокує підвищення часу завантаження, а також збільшення обсягу пам'яті та енергоспоживання локального браузера. По-друге, для HTML-коду кожен окремий символ важливий, а тому важливо слідкувати за рівнями вкладення, порожніми тегами, відступами, коментарями. [6]

До загальних параметрів також належить оптимізація, пов’язана із фізичною інфраструктурою. І якщо розробник, приміром, не уповноважений на прийняття рішення про вибір хостинг-середовища, то попри це можна принаймні повпливати на особливості серверного кешування, використання стандартів передачі даних клієнт-серверної комунікації, споживання даних та їх стиснення.

Наприклад, якщо сервер підтримує стиснення Brotli або стандарт HTTP/2, це полегшує застосування ефективних інструментів оптимізації по принципу “out-of-box”, просто шляхом активації налаштувань хосту. З іншого боку, деякі хости навіть не надають можливості визначити обсяг даних на веб-сторінці, не говорячи про більш нові стандарти. [2, ст. 171-176]

Емпіричної сторони дослідження більше стосуються спеціальні параметри оптимізації, які групуються навколо програмного коду додатку та залежать пропорційно від обраних інструментів розробки. Для клієнтської частини веб-додатку це бібліотека React.js та фреймворк Angular, покликані показати відмінності в підходах вирішення проблеми часткового оновлення вмісту веб-сторінки.

Основним пунктом дотику між технікою та бібліотеками є те, як останні послуговуються нею. Для цього в Angular існує вмонтований механізм Change Detection, відповідальний за порівняння змін виду сторінки внаслідок взаємодії з нею користувача. Ми пропонуємо відмовлятись від дефолтного Change Detection на користь Zone-Less Change Detection. В крайньому разі, необхідно принаймні застосовувати різні стратегії визначення змін. Якщо вхідні дані компонента залишаються незмінні, то немає сенсу проводити їх повторну перевірку, а робити це лише тоді, коли в компоненті присутня асинхронна логіка.

Стратегія виявлення змін OnPush є стратегією оптимізації в Angular. За цією стратегією виявлення змін запускається лише тоді, коли виконуються певні умови, такі як зміна вхідного властивості компонента або спрацювання події з цього компонента.

В React існує схожий механізм,- Virtual DOM, який проте не контролюється розробником, а є повністю protected (від англ.- захищеним) і повністю автоматизованим. Для сприяння порівняння виду сторінки, додаємо key-атрибут при обробці великих об'ємів даних, мемоізуємо вхідні дані компоненти після першого їх надходження, використовуючи HOC React.memo та Hooks useCallback та useMemo.

Наступним найбільш поширеним аспектом оптимізації є аналіз використовуваних ресурсів. В цьому контексті пропонуємо в першу чергу застосування DevTools для Profiling продуктивності. Крім цього, треба запроваджувати профілактику витоку пам'яті через використання Observables в зв'язці з @UntilDestroy декоратором для Angular та завдяки обробці підписок в useEffect для React. Особливе значення мають математичні розрахунки із використанням вкладених структур даних, які слід мутувати, щоб дотримуватись принципу чистих функцій та не змінювати вхідні дані.

Що стосується архітектури, то тут нашою рекомендацією було б обрати Feature-Slice та Modular підходи для забезпечення розширюваності, ясності та доступності. Важливо розбивати програмний код на частини за принципом Code Splitting та завантажувати в чанках/пакетах за потреби. Ідеально підходить вже згадуваний механізм lazy loading та suspense. Пам'ятаючи про обмеження DOM-у, ретельно розробляємо ієрархію компонентів, розбиваючи їх на підкомпоненти.

4. Результати дослідження

Для того, щоб порівняти те, наскільки ефективною стала робота додатку, скористаємось метриками Core Web Vitals. Вони вимірюють важливі аспекти взаємодії користувача з веб-сторінкою, та базуються на трьох найважливіших показниках:

1. Час відгуку додатку на першу взаємодію (First Input Delay)
2. Час затримки ініціалізації додатку (Largest Contentful Paint)
3. Якість відображення макету веб сторінки (Cumulative Layout Shift)

Таблиця 2

Стартові значення метрик 100 елементів

Метрика	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

Джерело: розроблено автором на основі власних тестувань [1-7]

До цих показників додамо також загальний час блокування (Total Blocking Time) та індекс швидкості (Speed Index).

Для цілей порівняння було створено декілька додатків на основі React та Angular. Під час першої ітерації було створено додаток із наповненням в 100 елементів із підключеними зображеннями, відео, шрифтом та стандартною архітектурою.

Відповідно до перших замірів, вхідні дані мали вигляд, представлений в таблиці 2.

Відповідно, оптимізовані показники для першої ітерації виглядають наступним чином (представлено в таблиці 3).

Таблиця 3

Оптимізовані значення метрик для 100 елементів

Метрика	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

Джерело: розроблено автором на основі власних тестувань [1-7]

Під час другої ітерації було створено додаток з більш складною архітектурою та з наповненням в 1000 елементів. При цьому вхідні показники мали вигляд, представлений в таблиці 4.

Таблиця 4

Стартові значення метрик для 1000 елементів

Метрика	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

Джерело: розроблено автором на основі власних тестувань [1-7]

Оптимізовані показники виглядають наступним чином (представлено в таблиці 5)

Таблиця 5

Оптимізовані значення метрик для 1000 елементів

Метрика	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

Джерело: розроблено автором на основі власних тестувань [1-7]

5. Висновок

Механізм часткового оновлення вмісту веб-сторінки є потужним інструментом для користувацького досвіду клієнтської частини веб-додатку. Проте, для того, щоб повною мірою скористатись перевагами, які дає ця техніка, та запропонувати користувачеві максимально якісний досвід, необхідно постійно оптимізувати проект. В дослідженні розглянуті

основоположні поняття, що стосуються функціонування оновлення вмісту веб-сторінки в односторінковому веб-додатку, а також проаналізовано параметри, що відповідають за високий рівень продуктивності та методи впливу на них із ціллю покращення кінцевих метрик Core Web Vitals. За результатами проведеного дослідження встановлено, що адаптація проаналізованих параметрів згідно із розглянутими методами, становить левову частку для збалансування додатку в цілому та є відмінним інструментом для покращення роботи механізму оновлення контенту з ціллю запропонувати більш якісний користувацький інтерфейс.

Список використаних джерел

1. Beglerović V., Pirija L., Prazina I. and Okanović V., "Detection and Logging Changes in Web Pages," 2022 21st International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 2022, pp. 1-5, doi: 10.1109/INFOTEH53737.2022.9751305.
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” et al 2021 J. Phys.: 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. and Prade M..Performance Issues and Optimizations in JavaScript: An Empirical Study. IEEE/ACM 38th International Conference on Software Engineering (ICSE), 2016, pp. 61-72, doi: 10.1145/2884781.2884829, 2016.
6. Van Riet J., Malavolta I. "Client-side Performance of Web-based Applications: the State of the Art." 2019. [Online; accessed 12. Oct. 2023]. Available at: https://jaspervanriet.nl/assets/literature_study.pdf.
7. Vesper. Measuring time-to-interactivity for modern web pages. In: Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation, NSDI. USENIX Association. [Online; accessed 14. Nov. 2023] 2018 Available at: https://www.cs.princeton.edu/~ravian/publications/vesper_nsdi18.pdf
8. Moshenchenko M., Zhurakovskyi B., Poltorak V., Bondarchuk A., Korshun N. Optimization Algorithms of Smart City Wireless Sensor Network Control / CEUR Workshop Proceedings, 2021, 3188, p. 32–42
9. Zhebka V., Gertsiuk M., Sokolov V., Malinov V., Sablina M. Optimization of Machine Learning Method to Improve the Management Efficiency of Heterogeneous Telecommunication Network CEUR Workshop Proceedings, 2022, 3288, p. 149–155