УДК 004.41

МЕТОДИ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ОДНОСТОРІНКОВИХ ЗАСТОСУНКІВ НА ПРИКЛАДІ ФРЕЙМВОРКУ REACT

Олександр Сюх

PERFORMANCE OPTIMIZATION METHODS FOR SINGLE-PAGE APPLICATIONS USING THE REACT FRAMEWORK

Oleksandr Siukh

Волинський національний університет імені Лесі Українки, просп. Волі, 13, Луцьк, 43025, Україна

***Abstract**. The paper explores performance optimization methods for single-page applications using React. It focuses on lazy loading, code splitting, memoization, and dynamic imports. The impact of these techniques on client-side speed and efficiency is analyzed.*

У сучасній веброзробці дедалі більшого поширення набувають односторінкові застосунки (SPA), які забезпечують динамічну взаємодію з користувачем без необхідності повного перезавантаження сторінки. Такий підхід значно покращує користувацький досвід, роблячи застосунок динамічним. Зі зростанням обсягу функціональності та ускладненням інтерфейсів виникає необхідність у застосуванні ефективних механізмів оптимізації.

Фреймворк React надає можливість зменшити час початкового завантаження, підвищити швидкість взаємодії з інтерфейсом і знизити навантаження на ресурси браузера. Йдеться про створення швидких, зручних та масштабованих вебзастосунків в умовах високих очікувань з боку кінцевих користувачів. Крім того, продуктивність SPA-застосунків є критичною для забезпечення стабільної роботи масштабованих систем, здатних обробляти великі обсяги даних у режимі реального часу.

Одним із ефективних підходів до оптимізації продуктивності односторінкових вебзастосунків є використання механізму відкладеного завантаження (lazy loading). Ця техніка передбачає динамічне підключення окремих компонентів інтерфейсу або модулів лише в момент їхньої фактичної необхідності, а не під час початкового завантаження застосунку [1]. Застосування відкладеного завантаження дозволяє суттєво зменшити обсяг початкового пакета ресурсів, що завантажуються у браузер користувача. У результаті реалізації даного підходу підвищується швидкість початкового рендерингу сторінки, зменшується час до взаємодії з користувачем, а також оптимізується використання оперативної пам’яті, що особливо важливо для ресурсно-обмежених пристроїв.

Ще одним важливим інструментом у процесі підвищення ефективності вебзастосунків виступає мінімізація. Вона полягає у видаленні з вихідного коду зайвих символів — таких як пробіли, коментарі, переноси рядків та незначущі елементи, — без зміни функціональності програми. В результаті зменшується об’єм файлів, які передаються клієнту, що, своєю чергою, скорочує час завантаження сторінки. У процесі збірки застосунку мінімізація часто виконується автоматично за допомогою спеціалізованих інструментів або вбудованих механізмів у збирачах [2]. В контексті React-застосунків такі оптимізації є ключовими при розгортанні продуктивної версії, адже вони дозволяють зменшити обсяг JavaScript-збірок та покращити загальний час рендерингу.

Серед технік оптимізації, орієнтованих на зменшення обчислювальних витрат під час рендерингу, особливе місце посідає мемоізація [3]. Загалом це підхід, що передбачає кешування результатів обчислень або рендерингу компонентів з метою уникнення повторного

виконання тих самих операцій за незмінних вхідних даних. Це дозволяє запобігти повторним відтворенням функціональних компонентів, що може бути критично важливим у випадках складної логіки або великого обсягу даних. Застосування мемоізації також сприяє зниженню навантаження на інтерпретатор JavaScript, підвищенню чутливості інтерфейсу до дій користувача.

Варто також зазначити, що використання технології SPA (Single Page Application) вже на етапі архітектурного проєктування сприяє оптимізації функціонування вебзастосунку. Основна перевага полягає у тому, що такий застосунок завантажує єдину HTML-сторінку, яка надалі оновлюється динамічно за допомогою JavaScript без потреби повторного завантаження сторінки з сервера. Це дозволяє значно зменшити кількість мережевих запитів, мінімізувати затримки між діями користувача і відповіддю інтерфейсу, а також знизити загальне навантаження на серверну інфраструктуру [4]. У цьому контексті фреймворк React забезпечує додатковий рівень оптимізації завдяки використанню віртуального DOM — проміжного представлення структури інтерфейсу у пам’яті, що дає змогу ефективно виявляти і застосовувати лише ті зміни, які справді необхідні для оновлення сторінки. Крім того, React підтримує повторне використання компонентів, що зменшує надлишкові обчислення, а також дозволяє впроваджувати умовний рендеринг, завдяки чому компоненти створюються лише тоді, коли вони потрібні в поточному контексті взаємодії.

Ефективне управління продуктивністю односторінкових застосунків є критичним чинником у розробці сучасних вебінтерфейсів. Використання React як фреймворку надає розробникам не лише зручний інструментарій для побудови компонентної архітектури, а й вбудовані можливості оптимізації на рівні візуального рендерингу та завантаження ресурсоємних частин застосунку. Застосування таких технік, як відкладене завантаження, мемоізація та мінімізація забезпечує баланс між швидкодією, масштабованістю та стабільністю клієнтської частини застосунку. У поєднанні з перевагами архітектури SPA це створює фундамент для побудови продуктивних і чутливих до дій користувача систем, що відповідають сучасним вимогам до веброзробки.

Бібліографія

1. *React.dev*. *lazy*. URL: https://react.dev/reference/react/lazy#lazy (дата звернення: 17.05.2025).
2. Оптимізація продуктивності. *React Dev*. URL: https://uk.legacy.reactjs.org/docs/optimizing-performance.html (дата звернення: 17.05.2025).
3. Подобєд О. Оптимізація продуктивності в React-застосунку. *DOU*. 26.09.2023. URL: https://dou.ua/forums/topic/45406/ (дата звернення: 17.05.2025).
4. Побережний О. Що таке SPA додаток?. *Asabix*. 06.03.2024. URL: https://asabix.com.ua/what-is-a-single-page-application/ (дата звернення: 17.05.2025).