УДК 004.432

МЕТОД ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ФРОНТЕНД-ЗАСТОСУНКІВ З ВИКОРИСТАННЯМ WEBASSEMBLY

Романенко С.О.
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут
Україна, Київ
serhii.romanenko@viti.edu.ua

Анотація. *У даній роботі представлено метод оптимізації фронтенд-застосунків, що передбачає перенесення обчислювальних модулів у C/C++ з подальшою компіляцією до WebAssembly. На прикладі інтерактивного застосунку продемонстровано скорочення часу рендерингу та зниження навантаження на CPU. Результати підтверджують ефективність WebAssembly для задач з високою обчислювальною складністю.*

Ключові слова: *WebAssembly; оптимізація продуктивності; фронтенд-розробка; інтеграція JavaScript та Wasm; веб-застосунки.*

Вступ. У сучасній веб-розробці одним із ключових завдань є підвищення продуктивності фронтенд-застосунків. Користувачі очікують швидкого завантаження сторінок, високої інтерактивності та плавної роботи інтерфейсу навіть за складної графіки, анімацій або великих обчислювальних навантажень. Одним із перспективних рішень цієї проблеми є використання технології WebAssembly (Wasm) — низькорівневого байт-коду, який виконується у браузері майже з нативною швидкістю. WebAssembly компілюється з таких мов, як C або C++, і працює у захищеному середовищі, взаємодіючи з JavaScript [1].

Актуальність теми зумовлена потребою у високопродуктивних веб-рішеннях без збільшення навантаження на сервери. Попри значний прогрес у швидкодії JavaScript, WebAssembly відкриває можливість переносити у браузер обчислювально інтенсивні завдання, зменшуючи час їх виконання та підвищуючи чутливість інтерфейсу.

Мета роботи — дослідити метод інтеграції WebAssembly у фронтенд-архітектури для оптимізації продуктивності та визначити його ефективність у реальних сценаріях..

Основна частина. Запропоновано метод оптимізації фронтенд-застосунків, який передбачає ідентифікацію критичних ділянок коду, їх подальший рефакторинг у вигляді окремих функціональних модулів, реалізованих мовами C/C++, компіляцію таких модулів до формату WebAssembly та інтеграцію з JavaScript через спеціалізовані програмні інтерфейси. Проведено аналіз переваг та обмежень використання WebAssembly, серед яких: висока швидкодія, портативність, але водночас складність у процесі налагодження та обмежений прямий доступ до об’єктної моделі документа (DOM).

Під час реалізації запропонованого методу враховано особливості життєвого циклу веб-застосунків, що охоплює фази ініціалізації, рендерингу та обробки подій користувача. Інтеграція WebAssembly потребує чітко спроєктованої архітектури взаємодії між модулями, створеними різними мовами програмування, що зумовлює необхідність формування ефективних «мостів» (bridge API) між JavaScript та Wasm-компонентами. Ускладнення процесу зумовлюється відсутністю безпосереднього доступу до DOM, що потребує оптимального розподілу функціональних обов’язків між компонентами [2].

Додатково проаналізовано вплив структури та розміру Wasm-модулів на загальну ефективність роботи застосунку. Встановлено, що надмірна фрагментація у вигляді великої кількості дрібних модулів призводить до збільшення часу завантаження та зростання накладних витрат при викликах між середовищами. Оптимальним рішенням визначено об'єднання взаємопов'язаних функцій у більші за обсягом модулі та застосування асинхронного завантаження для зменшення впливу на початковий час відгуку інтерфейсу.

Експериментальне дослідження проведено на прикладі інтерактивного візуалізаційного веб-застосунку, що виконує рендеринг значного обсягу графічних елементів у режимі реального часу. Заміна рендер-двигуна, реалізованого на JavaScript, на WebAssembly-модуль дала змогу скоротити середній час рендерингу на 9% та знизити загальне навантаження на центральний процесор. Результати експерименту підтвердили позитивний вплив запропонованого підходу на чутливість інтерфейсу до дій користувача та плавність відтворення анімацій. Крім того, прискорене виконання обчислень потенційно зменшує енергоспоживання мобільних пристроїв [3].

Отримані результати дозволяють стверджувати, що застосування WebAssembly є доцільним не лише у сфері високопродуктивної графіки, але й для задач, пов'язаних із обробкою мультимедійного контенту, виконанням складних математичних обчислень безпосередньо у браузері. Це створює передумови для розробки кросплатформних клієнтських застосунків із підвищеною продуктивністю та розширеними можливостями.

Висновки. Таким чином, використання WebAssembly у фронтенд-розробці є дієвим методом підвищення продуктивності, особливо в умовах високих вимог до обчислювальних ресурсів. Запропонований підхід не лише покращує продуктивність, але й відкриває нові можливості для реалізації складних обчислювальних задач у браузері без потреби в бекенд-ресурсах. Подальші дослідження передбачають розширення методу для підтримки мультиплатформних рішень та автоматизацію побудови взаємодії між JavaScript та WebAssembly. У перспективі передбачається також розробка візуальних інструментів для генерації Wasm-модулів без глибоких знань C/C++. Це дозволить розробникам швидше адаптувати існуючі проєкти до високопродуктивної архітектури. Інтеграція таких рішень у навчальні програми сприятиме підвищенню цифрової грамотності фахівців у галузі веб-інженерії. У підсумку, WebAssembly постає не лише як інструмент оптимізації, а й як ключовий елемент майбутньої еволюції веб-платформ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Monteiro, F. (2023). Learning Single-page Web Application Development. Packt Publishing – Ohio, 143–165.
[2] P. Smith, & J. Thompson. (2022). High-Performance Frontend Development with WebAssembly. Cambridge University Press– Cambridge, 107–108.
[3] Yun, L., & Bliault, A. (2020). Optimizing Complex Data Processing in Frontend with Rust and WebAssembly.

Romanenko S.O.

Method for optimizing the productivity of front-end applications using Webassembly

Abstract. *This paper presents a method for optimizing front-end applications, which involves porting computational modules to C/C++ with subsequent compilation to WebAssembly. Using the example of an interactive application, the reduction of rendering time and reduction of CPU load are demonstrated. The results confirm the effectiveness of WebAssembly for tasks with high computational complexity.*

Keywords: *WebAssembly; performance optimization; front-end development; JavaScript and Wasm integration; web applications*