ВЕБ-БРАУЗЕРНІ РОЗШИРЕННЯ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ WEBASSEMBLY: ПОТЕНЦІАЛ ТА ВИКЛИКИ

Андрієвич Дмитро Юрійович
студент групи ІАСм -1-23-1.4д,
Київського столичного університету імені Бориса Грінченка,
науковий керівник – д.т.н., проф. Бушма О.В.

У сучасному світі веб-технологій спостерігається стрімкий розвиток, що відкриває нові можливості для створення потужних та ефективних веб-застосунків. Особливу увагу привертає технологія WebAssembly (WASM), яка дозволяє виконувати код у браузері зі швидкістю близькою до нативної [1]. Це відкриває нові горизонти для розробки веб-браузерних розширень, які можуть значно розширити функціональний потенціал браузерів та практичні можливості користувачів.

Мета роботи – визначити можливості та виклики, пов'язані з використанням WebAssembly для розробки веб-розширень. В ході розвідки не лише теоретично обґрунтовуються переваги цієї технології, але й практично визначається доцільність її застосування на прикладі розробки реального веб-розширення.

WebAssembly є бінарним форматом інструкцій, розроблених для ефективного виконання коду у веб-середовищі [1]. Для дослідження проблем та викликів, пов’язаних з використанням WebAssembly, проведено практичне дослідження, у ході якого розроблено застосунок у вигляді веб-розширення, що дозволяє планувати задачі. При цьому використовувався Rust та фреймворк Leptos для створення користувацького інтерфейсу, а також WebAssembly для компіляції та виконання коду в браузері. З функціональної точки зору розширення включає базовий набір функцій для управління задачами, зокрема, створення, редагування, видалення та сортування задач. Особливу увагу було приділено безпеці, а саме: було впроваджено систему автентифікації та шифрування даних для забезпечення конфіденційності користувацької інформації.

Дослідження показало, що використання WASM має ряд суттєвих переваг над розробкою з використанням лише JavaScript:

1. **Висока продуктивність** – WASM-код виконується значно швидше, ніж JavaScript, що особливо важливо для розширень, які містять складні обчислення або обробку даних.
2. **Безпека** – виконання коду в ізольованому середовищі [2] підвищує загальний рівень безпеки розширень, що є критичним фактором для користувачів.
3. **Оптимізація ресурсів** – WASM-модулі зазвичай мають менший розмір порівняно з еквівалентним JavaScript-кодом, що сприяє більш швидкому завантаженню та покращеному досвіду користувачів.

Незважаючи на значний потенціал, було виявлено ряд викликів, пов'язаних з використанням WebAssembly для розробки веб-розширень:

1. **Обмежений доступ до API браузера** – WASM-модулі мають обмежений прямий доступ до функцій браузера, що вимагає додаткової інтеграції з JavaScript. Це, в свою чергу, вимагає більшої залученості з боку розробників та збільшує витрати часу на розробку.
2. **Труднощі з налагодженням** – інструменти для налагодження WASM-коду все ще перебувають на ранніх стадіях розвитку, що може ускладнювати процес розробки.
3. **Складність розробки** – розробка з використанням WebAssembly вимагає від розробників додаткових зусиль та навчання, особливо якщо вони раніше працювали лише з JavaScript.

Важливим наступним етапом дослідження є порівняльний аналіз швидкодії розробленого WASM-розширення з аналогічними розширеннями, створеними з використанням JavaScript. Це дозволить оцінити реальні переваги використання WebAssembly, особливо при створенні високопродуктивних веб-розширень [3].

Було визначено, що WebAssembly має значний потенціал для розробки високопродуктивних веб-розширень. Проте важливо звернути увагу на певні проблеми, які було виявлено на етапі розробки: недостатня кількість документації та відсутність базового функціоналу, наявного у відповідних JavaScript фреймворках. Ці проблеми потребують додаткового дослідження та покращення відповідних інструментів для полегшення процесу створення WASM-розширень з використанням Rust.

Подальші дослідження технології WebAssembly дозволять розширити її можливості, полегшити розробку веб-розширень та вирішити наразі існуючі проблеми. Поява зручних інструментів для розробників неминуче прискорить загальну інтеграцію WebAssembly в множину технологій створення потужних веб-застосунків [4].

ДЖЕРЕЛА

1. WebAssembly Concepts. URL: https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts - 2024.

2. D. Lehmann, J. Kinder, M. Pradel. Everything old is new again: Binary security of WebAssembly. – USENIX Association, 2020.

3. A. Jangda, B. Powers, E. D. Berger, A. Guha. Not So Fast: Analyzing the Performance of WebAssembly vs. Native Code. – USENIX Association, 2019.

4. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhyltsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. https://doi.org/10.28925/9720213284km