ДОСЛІДЖЕННЯ ПРОДУКТИВНОСТІ FRONTEND-ФРЕЙМВОРКІВ НА ПРИКЛАДІ РЕАЛЬНОГО ПРОЄКТУ

Довгаль М.І.
Київський столичний університет імені Бориса Грінченка, м. Київ

ВСТУП. Актуальність і постановка проблеми. Вибір frontend-фреймворку є одним із ключових архітектурних рішень при розробці сучасних веб-застосунків. Ринок пропонує широкий спектр рішень, React, Angular, Vue, Svelte, Solid та інші, кожне з яких супроводжується маркетинговими заявами щодо продуктивності та зручності. Проте об'єктивні, відтворювані порівняння різних технологій в однакових умовах залишаються рідкістю. Більшість доступних бенчмарків або вимірюють лише синтетичні сценарії, або не забезпечують контрольованих умов для коректного порівняння [1]. Через це практичні рішення нерідко приймаються на основі суб'єктивних переваг чи популярності технології, а не на основі реальних вимірювань продуктивності.

Мета дослідження. Метою роботи є емпіричне дослідження та порівняльний аналіз продуктивності сучасних frontend-фреймворків на основі реального відкритого benchmark-стенду з метою отримання обґрунтованих практичних рекомендацій щодо вибору технологічного стеку для різних типів веб-проєктів.

Аналіз останніх досліджень і публікацій. Проблематика порівняння frontend-фреймворків активно вивчається у технічній спільноті. Серед відомих підходів: проєкт js-framework-benchmark, який щорічно публікує результати порівняння продуктивності популярних рішень [2]. Окремі дослідники зосереджуються на метриках Web Vitals (FCP, LCP, INP), запроваджених Google як стандарт вимірювання користувацького досвіду [3]. Попри наявність таких досліджень, більшість із них або обмежуються невеликою кількістю фреймворків, або не охоплюють повний цикл від збірки до відображення у браузері, що обумовлює актуальність комплексного підходу.

Серед актуальних досліджень варто виокремити роботи, присвячені прямому вимірюванню DOM-операцій та часових метрик у браузерному середовищі. Зокрема, Bielak et al. [3] дослідили Angular, React та Vue у реальному CRUD-сценарії, а Diniz-Junior et al. [5] провели тестування з акцентом на маніпуляції з DOM та час інтерактивності. Зазначені дослідження підтвердили стійку перевагу легковісних рішень за часовими метриками і водночас виявили, що Lighthouse Performance Score не завжди корелює з DOM-продуктивністю.

Короткий опис дослідження, його методів і засобів.

Дослідження ґрунтується на відкритому benchmark-стенді framework-benchmarks. Методика передбачає реалізацію ідентичного погодного веб-застосунку в кожній із 12 досліджуваних технологій, React, Angular, Vue,

Svelte, Solid, Preact, Qwik, Lit, Alpine.js, jQuery, VanJS та Vanilla JS. Усі реалізації використовують спільні assets, стилі та mocked-дані, що виключає вплив стороннього контенту на результати.

Для автоматизованого вимірювання використовується єдиний набір Playwright-тестів, який запускає кожну реалізацію в ізольованому середовищі та фіксує ключові метрики продуктивності. Вимірювані показники охоплюють час production-збірки, розмір стисненого JavaScript-бандлу, час першої змістовної візуалізації, час відображення найбільшого контентного елемента та показник продуктивності за Lighthouse [4].Такий підхід дозволяє зосередитись на чітко вимірюваних, відтворюваних характеристиках продуктивності без впливу суб'єктивних факторів. Результати вимірювань зберігаються у файлі summary.tsv та є відкрито доступними для перевірки та відтворення.

Середовище тестування було стандартизовано для забезпечення відтворюваності результатів. Усі вимірювання проводились на платформі з процесором Intel Core i7, оперативною пам'ятю 16 GB, операційною системою Ubuntu 22.04 LTS та браузером Chromium 120. Кожен фреймворк запускався у окремому ізольованому процесі Playwright з чистим профілем браузера, що унеможливлює вплив кешу чи стану попереднього запуску. Кожну метрику вимірювано щонайменше тричі, а підсумковий результат обчислювався як медіанне значення для зменшення випадкових відхилень. Такий підхід відповідає рекомендаціям щодо надійного вимірювання продуктивності веб-застосунків, описаним у роботах [3, 4].

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ.

Зведені результати вимірювання продуктивності 12 frontend-підходів наведено в таблиці 1.

Отримані дані демонструють помітне розшарування досліджуваних технологій. Найкращі результати за сукупністю показників отримали легковісні рішення: VanJS (FCP 1214 мс, LCP 1384 мс, показник продуктивності 100), Solid (FCP 1359 мс, показник продуктивності 100), Preact (FCP 1372 мс, стиснений бандл 9,9 КБ) та Alpine.js (показник продуктивності 99,6). Vanilla JS показав найкращий FCP серед усіх досліджуваних рішень, 1214 мс, проте без повноцінного кроку збірки.

Vue.js показав найкращий баланс між продуктивністю та зрілістю екосистеми: FCP 1669 мс, LCP 1839 мс, показник продуктивності 99 при помірному розмірі бандлу 28,7 КБ. React залишається збалансованою платформою з показником продуктивності 97, проте поступився легковісним конкурентам за показниками FCP (1959 мс) та LCP (2264 мс).

Angular виявився найбільш ресурсоємним фреймворком: час збірки 10 414 мс (найгірший результат), стиснений бандл 72,8 КБ, FCP 2108 мс та LCP 2616 мс, найгірші серед усіх досліджуваних технологій. Проте для великих корпоративних систем Angular залишається виправданим вибором

завдяки суворій архітектурі, типізації та довгостроковій підтримуваності [5].

Таблиця 1

Ключові результати вимірювання продуктивності frontend-підходів

Фреймворк	Build time, мс	Gzipped size, КБ	FCP, мс	LCP, мс	Performance
Alpine.js	0	10.1	1363	1618	99.6
Angular	10 414	72.8	2108	2616	95.0
jQuery	727	33.9	1811	1975	92.0
Lit	497	15.4	1528	1844	99.0
Preact	516	9.9	1372	1676	99.4
Qwik	745	67.5	1401	2119	74.0
React	820	49.1	1959	2264	97.0
Solid	765	9.0	1359	1523	100
Svelte	1 917	37.4	1671	2130	98.0
Vanilla JS	0	10.7	1215	1520	94.0
VanJS	456	5.9	1214	1384	100
Vue	822	28.7	1669	1839	99.0

Результати підтверджують, що поняття «найшвидший фреймворк» і «найдоцільніший фреймворк» не є тотожними. Svelte та Solid підтвердили ефективність компіляційного підходу до зменшення витрат під час виконання. Qwik, попри цікавий архітектурний підхід, показав нестабільні результати: мінімальний FCP 1401 мс при водночас слабкому показнику продуктивності 74, що свідчить про незрілість поточної реалізації.

Окремий інтерес становить аналіз залежності між розміром стисненого бандлу та часовими метриками. Фреймворки з найменшим бандлом (VanJS – 5,9 КБ, Solid – 9,0 КБ, Preact – 9,9 КБ) стабільно демонструють найкращий FCP. Це підтверджує відому закономірність: завантаження та парсинг великого JavaScript-файлу є одним із ключових чинників затримки першого відображення сторінки [5]. Водночас Angular (72,8 КБ) і Qwik (67,5 КБ) мають найбільший бандл і водночас гірші часові показники. Варто зазначити, що між розміром бандлу та показником продуктивності Lighthouse немає лінійної залежності: jQuery (33,9 КБ) отримав лише 92 бали, тоді як Lit (15,4 КБ) набрав 99, що свідчить про вагомий вплив архітектурних рішень фреймворку на ефективність виконання, а не лише на розмір коду.

Аналіз показника LCP дозволяє глибше зрозуміти поведінку фреймворків під час відображення основного контенту. Різниця між FCP і

LCP у легковісних рішень (VanJS: 170 мс, Solid: 164 мс) є значно меншою порівняно з Angular (508 мс) та Svelte (459 мс), що вказує на більш рівномірне та предиктивне відображення контенту. Цей показник є критичним для SEO та оцінки користувацького досвіду за методологією Core Web Vitals, де LCP нижче 2500 мс вважається прийнятним [3]. Усі досліджувані фреймворки, крім Angular, укладаються в цей поріг, проте легковісні рішення забезпечують значно більший запас продуктивності.

Практичне значення результатів дослідження. Запропонована методологія benchmark-тестування є відтворюваною і може бути застосована для оцінки нових фреймворків у майбутньому. Отримані дані слугують емпіричною основою для ухвалення технічних рішень і можуть бути включені до архітектурного документу проєкту. Зокрема, для розробки вантажних мобільних застосунків або прогресивних веб-застосунків із суворими вимогами до часу взаємодії, рекомендованими кандидатами є VanJS, Solid та Preact. Для командних проєктів з горизонтом підтримки понад три роки і значним обсягом кодової бази доцільнішим є Vue.js або React, з огляду на зрілість екосистеми, доступність спеціалістів та широку документацію [6]. Angular виправданий у корпоративних системах, де пріоритетом є архітектурна строгість і типізація, а не мінімальний час завантаження.

Важливим аспектом є також відтворюваність benchmark-стенду. Оскільки вихідний код усіх 12 реалізацій і набір Playwright-тестів є відкритими, будь-яка команда розробників може самостійно відтворити вимірювання або розширити стенд новими фреймворками. Це перетворює результати дослідження на живий порівняльний ресурс, а не на статичну таблицю даних.

ВИСНОВКИ. За результатами дослідження продуктивності 12 frontend-фреймворків отримано такі висновки. Легковісні рішення (VanJS, Solid, Preact, Alpine.js) демонструють найкращі значення Web Vitals та показника продуктивності. Vue.js є доцільним варіантом для середніх і великих проєктів, де потрібен баланс між продуктивністю та зрілою екосистемою. React зберігає практичну цінність завдяки розвиненій екосистемі та широкій доступності фахівців. Angular залишається доцільним для великих корпоративних систем з тривалим життєвим циклом. Оптимальний вибір frontend-фреймворку визначається не лише benchmark-рейтингами, а й масштабом системи, складом команди, вимогами до продуктивності та бізнес-цілями проєкту. Результати роботи можуть бути корисним орієнтиром для розробників і архітекторів при виборі технологічного стеку.

Перспективи подальших досліджень полягають у розширенні benchmark-стенду шляхом включення нових frontend-фреймворків і бібліотек, а також у дослідженні їх продуктивності в умовах більш складних і наближених до реальних сценаріїв використання, зокрема

багатосторінкових застосунків, серверного рендерингу (SSR) та гібридних архітектур. Доцільним є також аналіз поведінки фреймворків у мобільних середовищах і при обмежених ресурсах пристроїв.

Важливим напрямом є поглиблене дослідження взаємозв'язку між архітектурними особливостями фреймворків та їх фактичною продуктивністю за метриками Core Web Vitals. Окрему увагу доцільно приділити впливу оптимізацій, таких як code splitting, lazy loading, tree shaking та використання WebAssembly, на кінцеві показники продуктивності.

ДЖЕРЕЛА

1. Shantyr, A., Zinchenko, O., Storchak, K., Bondarchuk, A., & Pepa, Y. (2025). Prediction of quality software quality indicators with applied modifications of integrated gradiates methods. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 15(2), 139–146. https://doi.org/10.35784/iapgos.6892

2. Krause, S. js-framework-benchmark: a widely used open benchmark for JavaScript UI frameworks. GitHub, 2024. URL: https://github.com/krausest/js-framework-benchmark

3. Bielak, K., Borek, B., & Plechawska-Wójcik, M. (2022). Web application performance analysis using Angular, React and Vue.js frameworks. Journal of Computer Sciences Institute, 23, 77–83. https://doi.org/10.35784/jcsi.2827

4. Piastou, M. (2023). Comprehensive performance and scalability assessment of front-end frameworks: React, Angular, and Vue.js. World Journal of Advanced Engineering Technology and Sciences, 09(02), 366–376. https://doi.org/10.30574/wjaets.2023.9.2.0153

5. Diniz-Junior, R.N.V., Figueiredo, C.C.L., Russo, G.S., Bahiense-Junior, M.R.G., Arbex, M.V.L., Santos, L.M., Rocha, R.F., Bezerra, R.R., & Giuntini, F.T. (2022). Evaluating the performance of web rendering technologies based on JavaScript: Angular, React, and Vue. In Proc. 2022 XLVIII Latin American Computer Conference. https://doi.org/10.1109/CLEI56649.2022.9959901

6. Levlin, M. (2020). DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte. Abo Akademi University. URL: https://www.doria.fi/handle/10024/177433

7. Kriskun, I., & Bondarchuk, A. (2026). Оцінювання ефективності управління проєктами в галузі інформаційних технологій. Європейський науковий журнал Економічних та Фінансових інновацій, 1(19), 264-274. https://doi.org/10.32750/2026-0123

8. Comparison of JavaScript Frontend Frameworks — Angular, React and Vue. International Journal of Innovative Science and Research Technology (IJISRT), 2022. URL: https://www.ijisrt.com/comparison-of-javascript-frontend-frameworks-angular-react-and-vue