

МЕТОДИ ОПТИМІЗАЦІЇ REACT RECONCILIATION АЛГОРИТМУ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-ДОДАТКІВ

Вінницький національний технічний університет

Анотація

У роботі проведено дослідження методів оптимізації reconciliation алгоритму бібліотеки React.js, що дозволяють підвищити ефективність клієнтської частини веб-додатків. Розглянуто механізм роботи алгоритму, визначено його ключові особливості та вплив на продуктивність веб-інтерфейсу.

Ключові слова: React.js, reconciliation алгоритм, оптимізація, Fiber архітектура, продуктивність, мемоізація, віртуалізовані списки, веб-додатки.

Abstract

The paper investigates methods for optimizing the reconciliation algorithm of the React.js library aimed at improving the efficiency of web application front-ends. The study examines the algorithm's operational mechanisms, identifies its essential features, and evaluates their impact on web interface performance.

Keywords: React.js, reconciliation algorithm, optimization, Fiber Architecture, performance, memoization, virtualized lists, web applications.

Вступ

Інтенсивний розвиток інформаційних технологій та веб-сервісів призводить до зростання вимог до швидкодії та ефективності сучасних веб-додатків. Особливо важливим аспектом є клієнтська частина додатків, яка безпосередньо впливає на взаємодію з кінцевими користувачами. Бібліотека React.js є одним із найпоширеніших інструментів для створення інтерфейсів завдяки своїй зручності та високій продуктивності.

У процесі роботи з великими обсягами даних або при частих змінах стану додатків reconciliation алгоритм може ставати перешкодою, яке призводить до сповільнення роботи інтерфейсів. Це викликає потребу в детальному дослідженні та впровадженні методів оптимізації, які дозволять уникнути проблем зі швидкістю, адже ключовим фактором, який забезпечує швидку та ефективну роботу React-додатків, є даний алгоритм, що відповідає за оновлення структури DOM.

Метою цієї роботи є аналіз та визначення найбільш ефективних методів оптимізації reconciliation алгоритму React.js для підвищення загальної продуктивності веб-додатків.

Результати дослідження

У ході проведеного дослідження було детально проаналізовано механізми роботи reconciliation алгоритму React.js та практично випробувано низку методів його оптимізації. Особливу увагу було приділено аналізу ключових припущень алгоритму, зокрема диференціації елементів за типами та ролі ключів (keys), які допомагають ефективно відслідковувати зміни у структурі DOM [1].

Було з'ясовано, що reconciliation алгоритм React.js, який спирається на алгоритм диференціювання (diffing), дозволяє з високою точністю визначити, які саме частини DOM-дерева необхідно оновити після змін стану додатку. Водночас, саме використання Fiber-архітектури дозволило розподілити процес reconciliation на окремі частини (fibers), що забезпечило можливість асинхронного виконання та пріоритизації задач. Це дозволяє React відкладати менш важливі оновлення інтерфейсу на користь критично важливих задач, таких як введення користувачем інформації або анімація. Таким чином, Fiber суттєво покращив користувацький досвід за рахунок зменшення часу відгуку інтерфейсу, особливо в умовах інтенсивної взаємодії користувача з додатком [2].

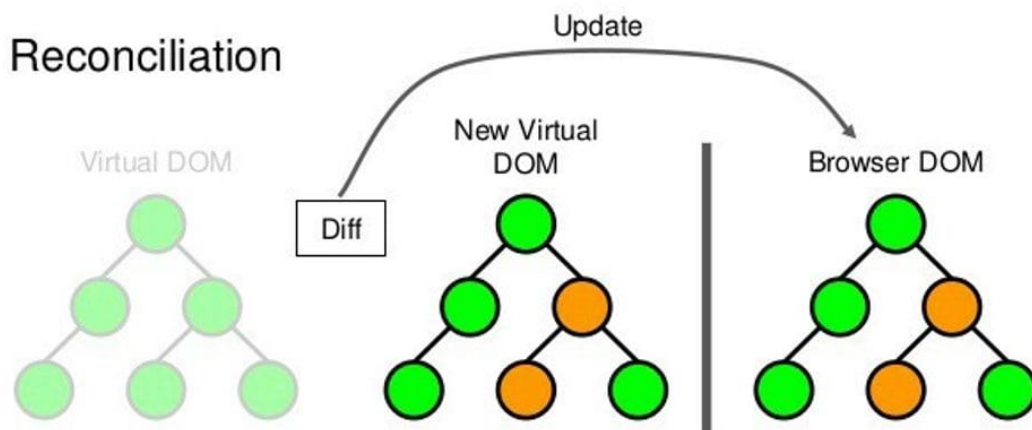


Рисунок 1 – Схематичне представлення роботи reconciliation алгоритму

Один із найважливіших напрямів оптимізації reconciliation алгоритму, виявлений у ході дослідження – це використання memoization компонентів. Практичні експерименти показали, що React.memo та хуки useMemo і useCallback дозволяють суттєво скоротити кількість зайвих рендерів. Згідно з результатами тестування, при правильному використанні memoization, вдається досягти зниження часу на рендеринг компонентів. Однак, було виявлено, що надмірне застосування memoization може збільшити використання оперативної пам'яті, що вимагає додаткового аналізу та балансування між продуктивністю і ресурсами.

Ключовим підходом оптимізації є правильне використання ключів (keys) у списках та інших групах динамічних компонентів. Ключі дозволяють алгоритму React точно ідентифікувати елементи, що були змінені, додані чи видалені. Ознакою успішного застосування keys є зменшення кількості DOM-операцій під час оновлення списків, що особливо помітно у великих, динамічних веб-додатках. Використання стабільних і унікальних ключів сприяє не лише швидшому оновленню інтерфейсу, але й підвищує стабільність роботи додатку [3].

Крім того, результати дослідження показали, що lazy loading (відкладене завантаження компонентів) є важливим інструментом для оптимізації початкового завантаження додатку. Використання цього методу дозволяє скоротити час початкового завантаження сторінок, що значно покращує взаємодію користувача з інтерфейсом. Це особливо корисно для веб-додатків з великою кількістю складних компонентів, які не обов'язково потрібні користувачам одразу при першому завантаженні сторінки.

Окремим способом покращення ефективності reconciliation алгоритму є використання virtualized lists — техніки, яка дозволяє завантажувати та відображати лише ті елементи списку, які перебувають у видимій області екрана користувача. Ознаками ефективного впровадження віртуалізації є значне скорочення навантаження на DOM та більш плавна взаємодія користувача із додатком, що особливо помітно при роботі з великими списками або табличними даними. Віртуалізація успішно вирішує проблему перевантаження інтерфейсу та забезпечує високу швидкість роботи навіть з великими обсягами інформації.

У рамках дослідження також було розглянуто ефективність нововведень Fiber-архітектури, зокрема можливість призупинення і відновлення reconciliation процесів. Виявлено, що це дозволяє React ефективно управляти великими задачами без втрати плавності інтерфейсу. Реалізація пріоритетів reconciliation в рамках Fiber-архітектури дозволила забезпечити значно кращий контроль за ресурсами браузера і уникати «заморожування» інтерфейсу навіть під час виконання складних завдань [4].

Висновок

У роботі було проведено детальний аналіз методів оптимізації reconciliation алгоритму бібліотеки React.js, які дають можливість суттєво підвищити продуктивність клієнтської частини веб-додатків. Було встановлено, що ефективність reconciliation алгоритму безпосередньо залежить від правильної реалізації низки методів оптимізації, серед яких ключовими є memoization компонентів,

використання унікальних ключів для елементів списків, lazy loading компонентів та віртуалізація великих списків (virtualized lists). Комплексне впровадження зазначених підходів дозволяє не тільки оптимізувати reconciliation процес, але й створити більш ефективні, стабільні та комфортні для користувачів веб-інтерфейси.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A Deep Dive into React Reconciliation Algorithm: Everything You Need to Know [Електронний ресурс] – Режим доступу: <https://www.dhiwise.com/post/a-deep-dive-into-react-reconciliation-algorithm> (дата звернення: 23.03.2025).
2. An Introduction to React Fiber - The Algorithm Behind React [Електронний ресурс] – Режим доступу: <https://www.velotio.com/engineering-blog/react-fiber-algorithm> (дата звернення: 23.03.2025).
3. Reconciliation [Електронний ресурс] – Режим доступу: <https://legacy.reactjs.org/docs/reconciliation.html> (дата звернення: 23.03.2025).
4. How does React's Reconciliation Algorithm work? [Електронний ресурс] – Режим доступу: <https://upmostlly.com/tutorials/how-does-reconciliation-in-react-work> (дата звернення: 23.03.2025).

Фоучек Володимир Олексійович – студент групи ІАКІТР-24м, кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: vinvova555333@gmail.com.

Богач Ілона Віталіївна – к.т.н., доцент кафедри автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: ilona.bogach@gmail.com.

Fouchek Volodymyr Oleksiiovych – student of ІАCІTR-24M group, Department of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: vinvova555333@gmail.com.

Bogach Ilona Vitaliivna – Associate Professor of Automation and Intelligent Information Technologies Department, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: ilona.bogach@gmail.com.