DOI 10.34185/1562-9945-5-124-2019-03
УДК 004.428

Н.Ю. Бессонова, О.С. Волковський

ЕФЕКТИВНІСТЬ РЕАКТИВНОГО ПІДХОДУ ПРИ РЕАЛІЗАЦІЇ ВБУДОВАНИХ БРАУЗЕРНИХ СИСТЕМ

Анотація. Сучасні браузери включають в себе багато можливостей для комфортної роботи в інтернеті, а так само безліч додаткових плагінів, які допомагають розширити їх базовий функціонал. Існують популярні розширення для загального використання, але так само існують специфічні розширення, які затребувані тільки вузькою групою осіб. В ході досліджень було з'ясовано, що розширення, яке здатне відстежувати відвідані посилання і повідомляти про це учасникам однієї мережевої корпоративної групи при повторному перегляді не було реалізовано на сьогоднішній день.

Для виконання цієї мети в роботі було запропоновано алгоритм Render Virtual DOM для платформи React, алгоритм Render Incremental DOM для платформи Angular, визначені переваги і недоліки двох вищевказаних підходів і представлений загальний алгоритм роботи браузерного розширення. Метою даної статті є адаптація запропонованих алгоритмів для розроблюваного розширення і виявлення найбільш ефективного підходу для його реалізації.

В результаті було розроблено розширення для якого була використана платформа React з функцією рендеринга на основі Render Virtual DOM і в якості бази був використаний web-браузер з типовою структурою Google Chrome.

Ключові слова — вбудовані браузерні системи, розширення браузера, перегляд посилань, мережева корпоративна група.

Постановка проблеми. Сучасна концепція розвитку програмного забезпечення часто будується виключно на базових функціях основної програми, а додаткова функціональність реалізується за допомогою спеціалізованих розширень. Браузери є типовими представниками саме такої ідеології.

Розширення браузера – це мікропрограма, за допомогою якої розширюється основний функціонал браузера для загальних чи вузькоспеціалізованих потреб користувачів.

У роботі пропонується використання реактивного підходу для розробки браузерного розширення яке підвищує ефективність пошуку інформації в мережах корпоративної групи.

Аналіз останніх досліджень. В літературі не було знайдено джерел відносно реалізації відстеження переглянутих посилань в межах однієї корпоративній мережі для групи зареєстрованих учасників та повідомлення про це при повторному перегляді іншим членам групи.

Формулювання цілей статті. Виходячи з інформації наданої вище були визначені наступні завдання:

- розробити алгоритм роботи браузерного розширення;
- адаптувати алгоритм Render Virtual DOM для React для розроблюваного додатку;
- адаптувати алгоритм Render Incremental DOM для Angular для розроблюваного додатку.

Основна частина. Розглянемо платформи React та Angular для розробки браузерних розширень, які покращують ефективність пошуку інформації в мережах однієї корпоративної групи. А саме, які дозволяють відстежувати переглянуті посилання кимось з членів обраної групи та повідомляти про це інших її учасників при повторному перегляді цього посилання.

Ми маємо залежність від різноманітних можливостей реалізацій розширення на певній платформі, які можуть покращувати, або погіршувати процес розробки.[1]

React одна з найпоширеніших платформ, саме в якій вперше почали використовувати Virtual DOM (Document Object Model). Головний принцип роботи полягає в тому, що кожний компонент створює нове Virtual DOM-древо кожного разу, коли відбувається рендеринг.[2]Дана платформа вносить набір змінень в DOM браузера тільки після того, як порівняє попереднє древо з новим, для того щоб відобразити нові змінення відповідно з новим Virtual DOM-древом. Алгоритм роботи зображено в блок схемі (рис.1).

Angular також дуже поширена платформа для реалізації різноманітних програм, в тому числі й браузерних розширень. Відносно нещода-

вно в Angular з'явився новий "двигун" рендеринга - Angular Ivy. Він відрізняється тим, що використовує Incremental DOM. Основний принцип його роботи складається в тому, що кожний компонент компілюється в набір інструкцій, які створюють DOM-древа й відповідно оновлюють їх, якщо дані було змінено. У цьому випадку функція template вміщує інструкції для виконання рендеринга та оновлення DOM-древа, але потрібно зауважити, що ці інструкції і є двигуном рендеринга. Роботу алгоритму відображено в блок схемі (рис. 2). [3]

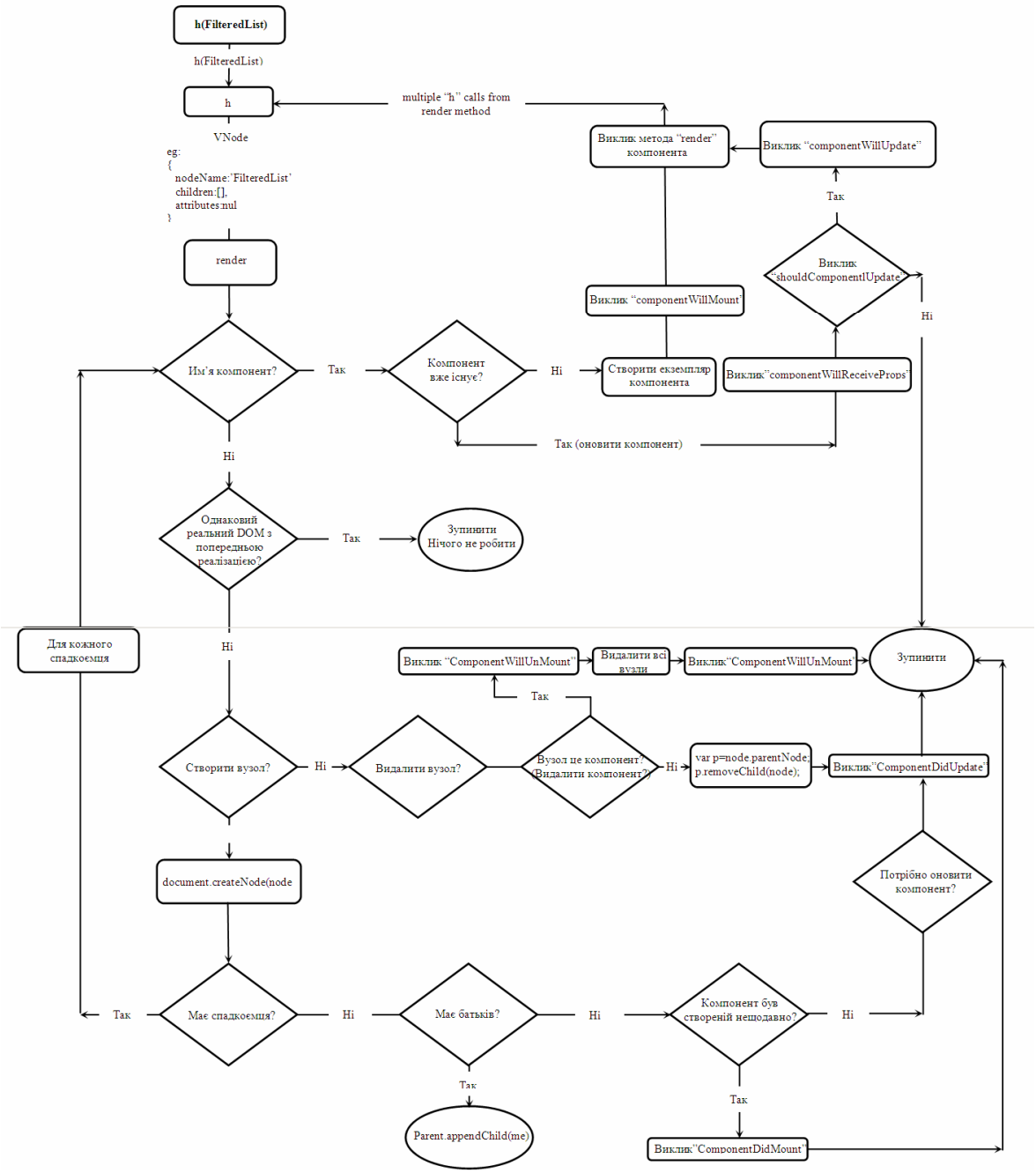

Рисунок 1 – Алгоритм Render Virtual DOM

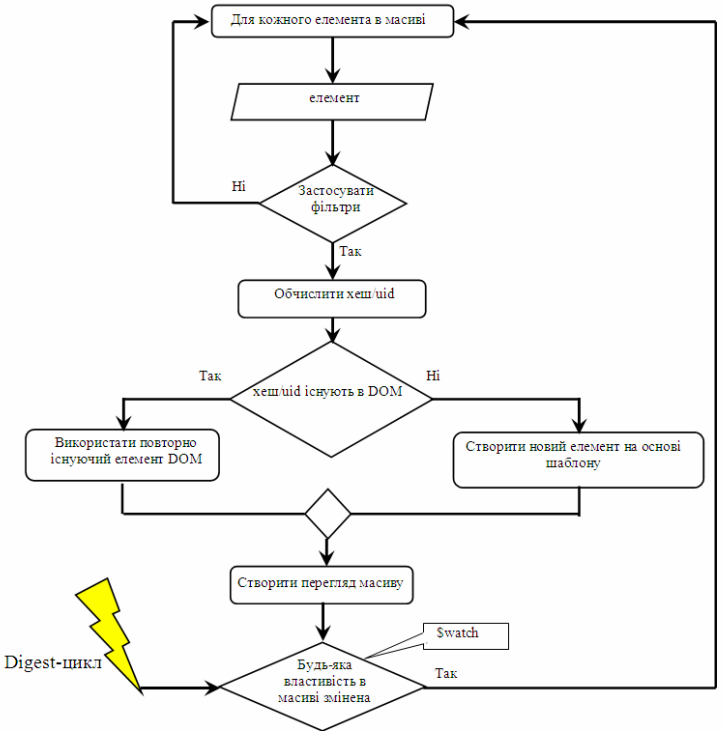

Рисунок 2 – Алгоритм Render Incremental DOM

Однією з переваг Incremental DOM є те, що він не інтерпретує компонент. Замість цього компонент посилається на інструкції. В цьому разі, якщо якась інструкція не була використана, а саме ці дані вже відомі на етапі компіляції, то вона не буде використана. Тобто таким чином, можна виключити такі інструкції з даних з кінцевого побудованого коду додатку (рис. 3).

У випадку з Virtual DOM потрібен інтерпретатор тому, що на етапі компіляції невідомо які з компонент будуть використані, і тому його необхідно цілком завантажувати в браузер. (рис.4)

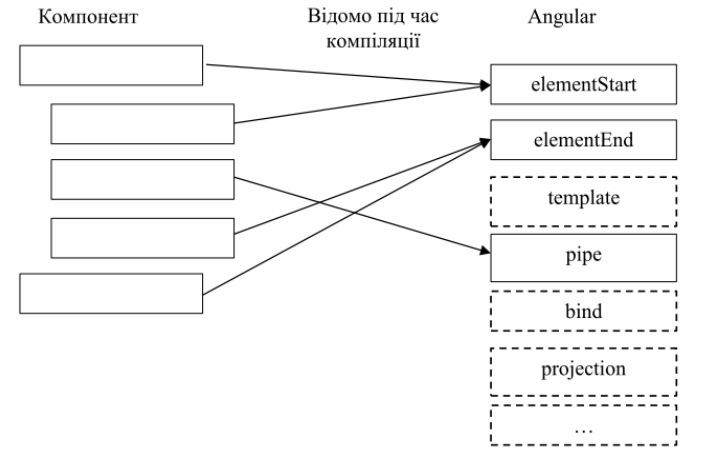

Рисунок 3 – Виключення невикористаних інструкцій

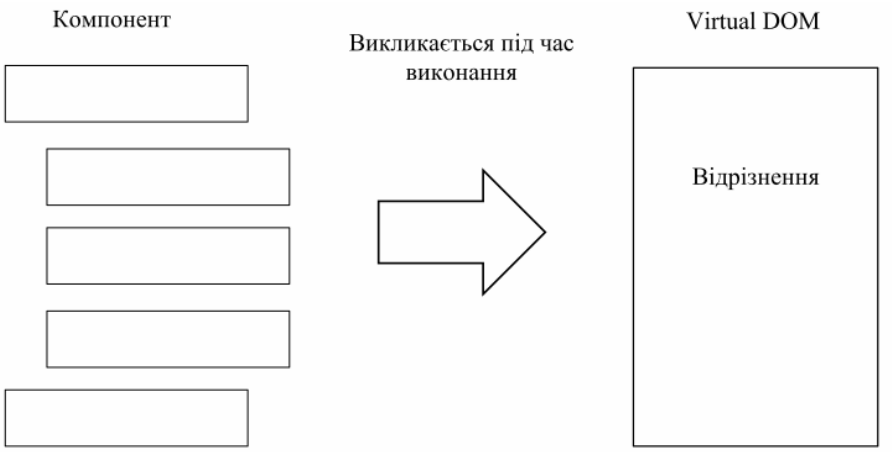

Рисунок 4 – Завантаження всіх компонент

Але не дивлячись на це, головними перевагами Virtual DOM на React є можливість використовувати будь-яку мову програмування для реалізації рендеринга компонент тому, що він не потребує компіляції. Ще однією перевагою є отримання значень як результат виконання рендеринга. Ці данні можуть допомогти при тестуванні або відладці.[4]

На основі вище зазначених міркувань, був розроблений загальний алгоритм роботи браузерного розширення на основі React з використанням алгоритму роботи Render Virtual DOM. (рис. 5)

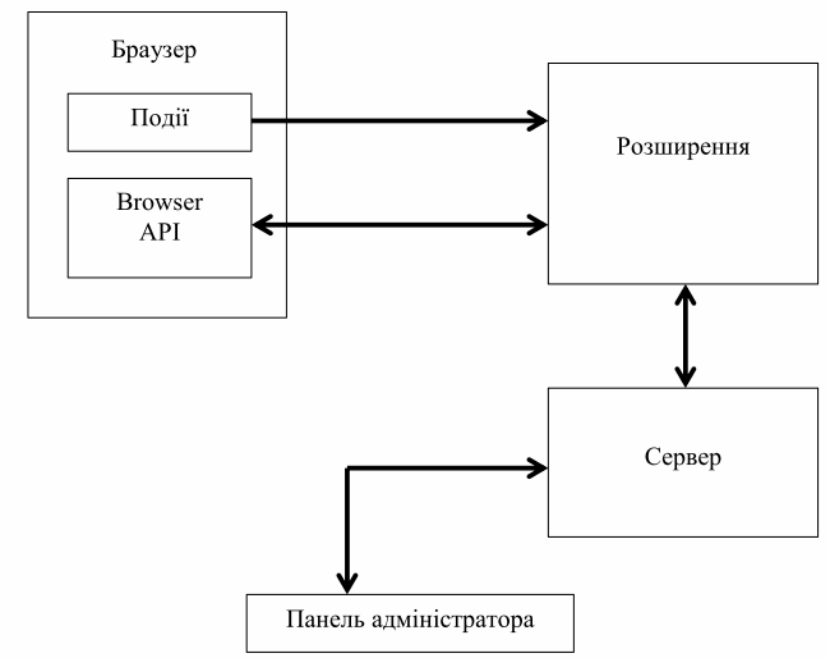

Рисунок 5 – Загальний алгоритм роботи браузерного розширення для відстеження переглянутих посилань в корпоративній мережі

Розширення буде працювати наступним образом. Корпоративна група людей буде зареєстрована у певній мережі. Кожний учасник буде відвідувати посилання. Розширення буде виступати у ролі “слухача”. Воно буде “слухати” події браузеру, а саме відкриття нової вкладки чи вікна. Якщо така подія відбудеться, плагін зробить виклик API браузера для того, щоб отримати посилання, яке відкрив користувач. Потім розширення буде відсилати запит на сервер та отримувати відповідь. Якщо такого посилання не було зареєстровано для іншого користувача, то доповнення фіксуватиме відвідування за поточним користувачем. А якщо буде знайдено збіг, то сервер поверне відповідь розширенню, а додаток, в свою чергу, покаже повідомлення користувачу про те, що відповідний член групи вже переглянув це посилання.

Висновки. Використання реактивного підходу є досить популярним та актуальним, оскільки за допомогою нього можна реалізувати багато програм для різних сфер діяльності людини. Використання алгоритму Render Virtual DOM сприяє розробці браузерного розширення та покращення ефективності його роботи. Алгоритм Render React має багато перевірок на необхідність запуску рендеринга. React тримає в пам’яті весь DOM, який було збудовано якщо необхідно щось відрендерити. Він перевіряє якщо саме з цими даними рендер вже відбувся, то нічого не буде робити для цього компонента. React декілька разів перевірить, перш ніж зробить одну з найвачих операцій. Після цього він побудує змінені частини DOM та змонтує їх в справжній DOM. Розроблений додаток на платформі React з використанням Virtual DOM на основі браузера Google Chrome. Розширення дозволяє повідомляти учасників однієї корпоративної групи про те, що посилання вже було переглянуто іншим учасником.

ЛИТЕРАТУРА / ЛІТЕРАТУРА

1. Olga Filipova. Learning Vue.js 2 Learn how to build amazing and complex reactive web applications easily with Vue.js. — Packt Publishing Ltd, 2016. — 334 c. — ISBN 9781786461131.

2. Reactjs [Електронний ресурс]: Virtual DOM and Internals. Режим доступу: https://reactjs.org/docs/faq-internals.html

3. Angular [Електронний ресурс]: Server-side Rendering (SSR): An intro to Angular Universal. Режим доступу: https://angular.io/guide/universal

4. Pharr, Matt; Humphreys, Greg (2004). Physically based rendering from theory to implementation. Amsterdam: Elsevier/Morgan Kaufmann. ISBN 0-12-553180-X.

REFERENCES

1. Olga Filipova. Learning Vue.js 2 Learn how to build amazing and complex reactive web applications easily with Vue.js. — Packt Publishing Ltd, 2016. — 334 с. — ISBN 9781786461131.

2. Reactjs [Електронний ресурс]: Virtual DOM and Internals. Режим доступу: https://reactjs.org/docs/faq-internals.html

3. Angular [Електронний ресурс]: Server-side Rendering (SSR): An intro to Angular Universal. Режим доступу: https://angular.io/guide/universal

4. Pharr, Matt; Humphreys, Greg (2004). Physically based rendering from theory to implementation. Amsterdam: Elsevier/Morgan Kaufmann. ISBN 0-12-553180-X.

Received 11.10.2019.
Accepted 16.10.2019.

Эфективность реактивного подхода для реализации встроенных браузерных систем

В работе предложен алгоритм Render Virtual DOM для платформи React, Render Incremental DOM для платформи Angular, общий алгоритм работы расширения, с помощью которого можно отслеживать просмотренные ссылки и уведомлять об этом участников одной сетевой корпоративной группы. Для реализации задачи была использована платформа React с функцией рендеринга на основе Render Virtual DOM и в качестве базы был взят web-браузер с типовой структурой Google Chrome. Разработан и программно реализован алгоритм работы расширения.

Efficiency of a reactive approach for implementation of integrated browser systems

Modern browsers include many features for comfortable work on the Internet, as well as many additional plugins that help expand their basic functionality. In the course of the research, it was found that the extension, which is able to track visited links and inform the members of one network corporate group when re-viewing it, was not implemented to date. To accomplish this, the work proposed the Render Virtual DOM algorithm for the React platform, the Render Incremental DOM algorithm for the Angular platform, identified the advantages and disadvantages of the two above approaches and presented a general algorithm for the operation of browser extension. The purpose of this article is to adapt the proposed algorithms for the expansion being developed and to identify the most effective approach for its implementation.

As a result, an extension was developed for which the React platform was used with a rendering function based on the Render Virtual DOM, and a web browser with a typical Google Chrome structure was used as the base.

According to the proposed algorithm, the plugin works as follows: A corporate group of people is registered in a certain network. Each participant visits the links. The extension acts as a "listener". It "listens" to browser events, namely the opening of a new tab or window. If such an event occurs, the plugin will make a call to the browser API in order to get the link that the user opened. Then the extension sends a request to the server and receives a response. If such a link was not registered for another user, then the add-on captures visits of the current user. And if a match is found, the server will return a response to the extension, and the application, in turn, will show a message to the user that the corresponding group member has already reviewed this link.

Бессонова Наталия Юрьевна – магистр Днепровского Национального Университета имени Олеся Гончара.

Волковский Олег Степанович - кандидат технических наук, доцент кафедры компьютерных наук и информационных технологий Национального Университета имени Олеся Гончара.

Бессонова Наталія Юріївна - магістр Дніпровського Національного Університету імені Олеся Гончара.

Волковський Олег Степанович - кандидат технічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій Дніпровського Національного Університету імені Олеся Гончара.

Bessonova Nataliia - Master of the Dnipro National University named after Oles Honchar.

Volkovskyy Oleg - Candidate of Technical Sciences, Associate Professor of the Department of Computer Science and Information Technology of the Dnipro National University named after Oles Honchar.