Кузьменко О. В., ст. викладач кафедри комп'ютерних наук,
Вознюк Ю. М., магістрант
Державний університет «Житомирська політехніка»

МЕТОДИ ТА ЗАСОБИ ОПТИМІЗАЦІЇ VUE.JS ДОДАТКІВ

В житті сучасної людини веб-браузери відіграють надзвичайно важливу роль, віддаючи перевагу швидкому та безпечному пошуку контенту, користувачі підштовхують розробників до постійної оптимізації. Найголовнішою вимогою до браузерів є зручність, тому останні роки активно розроблюються різноманітні технології, аби користувач міг швидко вирішувати задачі як зі стаціонарного комп'ютера, так і з мобільних гаджетів. Однією з таких технологій є розробка додатків за допомогою Vue.js фреймворку. Оскільки даний фреймворк застосовується для розробки клієнтських додатків, тому потрібно забезпечити щоб результат якісно відображався на усіх пристроях, незалежно від їх характеристик, та підтримувався більшістю браузерів В основному вони використали такий підхід, як віртуальна побудова DOM-дерева, що значно пришвидшило відображення контенту та виконання скриптів на стороні клієнта [1].

Зазвичай фреймворк Vue.js застосовують для побудови SPA (Single Page Application) додатків, які виконуються лише браузером тобто на стороні клієнта. Така побудова створена для покращення відображення веб-сторінок для користувачів, але якщо в додатку реалізовано забагато бізнес-логіки, яка взаємодія з користувацьким інтерфейсом, то лише одного підходу виявляється замало для комфортної роботи з додатком. Тому для оптимізації клієнтської частини веб-додатку на Vue.js існує декілька способів [2]:

1. Створення функціональних компонентів. Якщо розробляється компонент, який не змінюється сам та не змінює інші елементи сторінки, то до цього компоненту можна додати атрибут functional. Функціональний компонент компілюється в звичайну функцію, яка не залежить від локального стану додатку, тому при перебудові сторінки сам компонент залишається незмінним, що пришвидшує роботу додатку.
2. Відокремлення бізнес-логіки, яка використовує багато ресурсів браузеру в окремі компоненти. Якщо додаток має компонент, який містить в собі обчислення, залежні від дій користувача, то такі обчислення потрібно винести в окремий компонент, тобто дочірній. Дана зміна ієрархії компонентів в системі допоможе тим що, коли користувач буде взаємодіяти з інтерфейсом системи, то батьківський компонент не буде перемальовуватись браузером, а лише будуть відпрацьовувати бізнес-логіка [1, 2].
3. Використовувати директиву v-show, замість v-if. Якщо на веб-сторінці рендеринг елементів відбувається залежно від умов дій користувача, то для оптимізації додатку слід використовувати таку директиву як v-show. Різниця директив v-if і v-show в тому, що для рендерингу першої відбувається перебудова DOM-дерева, що сповільнює роботу веб-додатку, а друга – вбудовується під час ініціалізації сторінки браузером і приховується від користувача за допомогою стилів [1, 2].
4. Використання вбудованого компоненту <keep-alive>. Даний компонент вбудований в фреймворк Vue.js та працює як обгортка над компонентом-маршрутизатором <router-view/>. Завдяки даному компоненту-обгортки при переході між різними маршрутами веб-сторінок, компоненти не видаляється з пам'яті браузера, що знижує обчислювальні ресурси браузером [3, 4].
5. Динамічне завантаження компонентів на сторінку. Зазвичай додатки мають декілька веб-сторінок, які завантажуються по відповідному маршруту. Коли користувач переходить на таку сторінку по одному з маршрутів, то відбувається рендеринг не лише цієї сторінки з її компоненатами, але й інших, що сповільнює отримання користувачем відповідної інформації [3, 4].

Використовуючи лише декілька практик можна значно пришвидшити роботу додатків написаних за допомогою фреймворку Vue.js. У будь-якому випадку оптимізувати сайт потрібно максимально. Чим швидше він буде працювати, тим більше буде задоволено користувачів, тим більший буде прибуток [1, 2].

Список використаних джерел

1. Керівництво по Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/web/vuejs/.
2. Документація Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/.
3. Документація Vuex [Електронний ресурс] – Режим доступу: https://vuex.vuejs.org/ru/guide/.
4. Документація Vue router [Електронний ресурс] – Режим доступу до ресурсу: https://router.vuejs.org/ru/.