УДК 004.9

Вознюк Ю. М., магістрант, гр. ІПЗм-21-1
Науковий керівник: Кузьменко О. В., старший викладач кафедри КН
Державний університет «Житомирська політехніка»

МЕТОДИ ОПТИМІЗАЦІЇ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-ДОДАТКІВ

У наш час люди не готові витрачати багато часу на очікування, поки завантажиться сторінка веб-додатку. Ідеальною тривалістю завантаження сайту вважається 1 секунда, проте такого результату досить важко досягнути за допомогою стандартних методів розробки. Тому для розробників це являється певним ідеалом, якого потрібно прагнути під час створення продукту.

Під час того, як користувач тільки переходить по посиланню на сайт, браузер за лічені секунди повинен відмалювати всі елементи сторінки, за це відповідає клієнтська частина додатку, яка відповідає за рендеринг візуальної частини і скриптів. Стали популярні такі технології як serverless архітектура, яка не передбачає розподіл на серверну та клієнтську частини, а працює лише на стороні клієнта, SPA (SinglePageApplication) – додатки, які теж виконуються лише на стороні браузера. Але буває і цього замало для досягнення цілі – в швидкості подачі інформації користувачу [1].

Для оптимізації клієнтської частини веб-додатку існує декілька методів:

1. Побудова DOM-структури за допомогою спеціальних методів браузера. Така операція як побудова DOM-дерева чи зміна його збільшує навантаження на браузер, навіть не дивлячись на те, що з кожним роком покращується їх продуктивність. Тому потрібно до мінімуму звести створення складної структури та вносити зміни однією або двома ітераціями. Для цього потрібно використовувати DocumentFragement, який являється контейнером для дочірніх елементів. У контейнері створюються та додаються HTML-елементи і потім його потрібно вже завантажити до DOM-дерева. Дана операція покращує роботу сайту, завдяки внесення фрагментів сторінки блоками.

2. Зменшити інтервал виконання подій та методів, які часто вик-ликаються. У більшій частині роботи необхідно додавати обробку подій, які відбуватимуться досить часто під час взаємодії з користувачем. Наприклад події window’sresize або onmouseover. Якщо обробка даних подій надто ресурсна, можна створити велике навантаження на браузер, а це, в свою чергу, призведе до поганих наслідків на стороні користувача. Debouncing обмежуватиме кількість разів виконання функції в межах часового інтервалу.

3. Використовувати кешування інформації. API-інтерфейси типу Web-сховищ покращили і спростили роботу з API Cookie, яку розробники використовували протягом багатьох років. Одна зі стратегій, яку можна використовувати під час роботи з пам’яттю для зберігання несуттєвих даних, був статичний контент. Тут маються на увазі фрагменти HTML, які були завантажені за допомогою AJAX та інших різноманітних методів, які потрібно запрошувати не більше одного разу. Таким чином, позбавляємося частих повторюваних запитів до сервера [2].

4. Використання CSS-анімацій. Можна стверджувати, що зростання бібліотек JavaScript, таких як JQuery та MooTools, призвели до складних анімацій. Сьогодні багато розробників все ще використо-вують JavaScript, щоб оживити елементи, незважаючи на те, що відповідні браузери підтримують CSS анімації через transform і keyframe. CSS-анімації ефективніші, ніж анімації JavaScript. CSS анімації також мають додаткову перевагу, набагато менше коду. Багато анімацій CSS обробляються GPU, і таким чином більш згладжені. Подібні властивості CSS-анімації підвищують продуктивність та якість анімації, використовуючи субпіксельну інтерполяцію. Таку анімацію процесору простіше виконувати, оскільки задіюється графічний процесор, що дуже важливо для мобільних пристроїв. Однак найбільш прийнятним буде використання CSS-анімації разом із підключенням керування через js. Наприклад, коли правила анімації прописані в CSS, а за допомогою JavaScript змінюються вхідні параметри (колір, тло, розмір тощо)[3].

За допомогою таких методів розробки можна значно пришвидшити обробку інформації та рендеринг на стороні клієнта.

Список використаних джерел

1. Еспозіто, Д. Розробка сучасних веб-додатків: аналіз предметних областей і технологій / Д. Еспозіто. - М.: Вільямс І. Д., 2017. - 464 с.

2. Заєць, А. М. Проектування і розробка WEB-додатків. Введення в frontend і backend розробку на JavaScript і node.js: Навчальний посібник / А. М. Заєць, Н.П. Васильєв. - СПб.: Лань, 2019. - 120 с.

3. Оптимізація графіки для веб [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/post/422531/#04.