Список використаних джерел

1. Фоменко І. Сучасні інформаційні технології в навчанні бібліотечних та інформаційних фахівців // Вісник Книжкової палати № 7, 1997.

2. Розенфельд Л., Морвіль П. Бібліотечна справа: Термінологічний словник [Текст] / Розенфельд Л. - 3 видання. - М. : Символ-Плюс, 1997.

3. Людина та інформаційне середовище [Електронний ресурс]. - Електронні дані. - Режим доступу: World Wide Web. (http://www.congress2008.dialog21.ru/Doklady/16410.htm).

4. Марчук Г. І., Агошков В. І. Введення в проекційно-сіткові методи. - М.: Наука, 1981.

УДК 004.06

Гончар В.М асистент
кафедри інформаційних технологій

ПОРІВНЯННЯ РІЗНИХ ВИДІВ РЕНДЕРУ ІНФОРМАЦІЇ НА ВЕБ-СТОРІНКУ

Донецький національний університет імені Василя Стуса, м. Вінниця

Зараз, через зростання кількості користувачів на веб-сторінках, а відповідно і до навантаження на сервер, на якому цей сайт і знаходиться, дуже важливим стає питання, яким чином виконувати процес відображення веб-сторінки користувачам, або як його називають програмісти, процесс рендеру, ви як користувач могли і не задумуватися над тим, яким чином вся інформація що є на сайті, потрапляє в ваш браузер, чиї ресурси для цього задіяні, адже коли ви вводите назву сайту в адресний рядок браузера, натискаєте enter, і за звичкою бачите сторінку, що запитується. Все просто: ввів назву сайту – сайт відобразився. Однак для більш допитливих хочу розповісти, що відбувається тим часом, як браузер починає отримувати шматки сайту (так, сайт приходить шматками, по-іншому — чанками) [1] і відображає повністю намальовану сторінку. Це і називається рендером, і він може бути різних видів, про що і буде йти мова нижче. Для початку давайте перерахуємо яких видів вони існують, а вже тоді детальніше розглянемо кожен з них. Отже види рендеру веб-сторінок бувають такі:

- SSR (Server-Side Rendering) - рендеринг на сервері клієнтської частини або універсальної програми в HTML
- CSR (Client-Side Rendering) - рендеринг програми на стороні клієнта (в браузері), зазвичай за допомогою DOM
- Регідратація — «завантаження» JavaScript компонентів на клієнт таким чином, щоб вони повторно використовували DOM-дерево та HTML дані, сформовані на стороні сервера
- Пререндеринг — запуск клієнтської програми під час складання для збереження початкового стану у вигляді статичного HTML.

Серверний рендеринг - це вид рендеру при якому у відповідь на запит на сервері генерується весь HTML-код сторінки. Це виключає необхідність додаткових запитів даних із боку клієнта, оскільки сервер бере всю роботу він, як відправити відповідь. Такий підхід дозволяє досягти швидкого першого відмальовування і першого змістового відмальовування [2]. Виконання логіки сторінки та рендеринг на сервері дозволяють уникнути надсилання клієнту великої кількості JavaScript, що призводить до меншого часу до інтерактивності. І це логічно, адже при серверному рендерингу користувачу надсилаються лише текст та посилання. Цей підхід добре спрацює на широкому діапазоні пристроїв та мережевих умов, а також відкриває можливості для цікавих браузерних оптимізації на кшталт потокового парсингу документа. При використанні серверного рендерингу користувачам не потрібно чекати на завершення роботи JavaScript, що забирає ресурси процесора, перш ніж вони зможуть почати працювати з сайтом [3]. Навіть якщо не можна уникнути використання стороннього JavaScript, серверний рендеринг дозволяє зменшити кількість вашого власного JavaScript і дає більше «бюджету» для решти. Однак цей підхід має один істотний недолік: формування сторінки на сервері займає певний час, що може призвести до більшого часу до першого байта.

Статичний рендеринг відбувається на етапі складання та надає швидке перше відмальовування, перше змістовне відтворення та час до інтерактивності — за умови, що кількість клієнтського JavaScript обмежена. На відміну від серверного рендерингу, тут вдається домогтися стабільно швидкого часу до першого байта, оскільки HTML-код сторінки не повинен генеруватися на льоту. Як правило, статичний рендеринг передбачає попереднє створення окремого HTML-файлу для кожного URL-адреси. Оскільки HTML-відповіді створені заздалегідь, статичний рендеринг можна розгорнути на кількох CDN, щоб скористатися перевагою кешування. Але такий спосіб рендерингу має один недолік — необхідно заздалегідь створити HTML-файли для всіх можливих URL. Це може бути дуже складно або навіть неможливо, якщо ви не можете заздалегідь сказати, які URL можливі, або якщо у вас сайт з великою кількістю унікальних сторінок.

Клієнтський рендеринг має на увазі рендеринг сторінок прямо у браузері за допомогою JavaScript. Вся логіка, отримання даних, шаблонизація та маршрутизація обробляються на клієнта, а не на сервері. За такого рендерингу складно підтримувати високу швидкість на мобільних пристроях. Можна наблизитись до продуктивності чистого серверного рендерингу, якщо виконувати мінімум роботи, мати вузький бюджет для JavaScript [4] та доставляти дані з мінімальною круговою затримкою. Критичні скрипти та дані можна надіслати пізніше за допомогою HTTP/2 Server Push або <link rel=preload>, що говорить парсерові розпочати роботу заздалегідь. Шаблони на кшталт PRPL варті уваги, тому що можуть забезпечити відчуття миттєвого першого та наступних переходів між сторінками. Основний недолік клієнтського рендерингу полягає в тому, що кількість необхідного JavaScript зазвичай

збільшується разом із зростанням програми. Ситуація погіршується з підключенням нових JavaScript-бібліотек, поліфілів та іншого стороннього коду.

Універсальний рендеринг (або просто «SSR») намагається усунути недоліки серверного та клієнтського рендерингу, використовуючи обидва підходи. Навігаційні запити на кшталт повного завантаження чи перезавантаження сторінки обробляються сервером, який рендерит додаток у HTML, потім JavaScript та дані, що використовуються для рендерингу, вбудовуються у підсумковий документ [5]. При правильній реалізації час першого змістового відмальовування буде як при серверному рендерингу, а повторний рендеринг буде проводитися на клієнті за допомогою техніки, яка називається регідратацією. Це нове рішення, проте не позбавлене певних проблем із продуктивністю. Основний недолік універсального рендерингу з регідратацією полягає в тому, що такий підхід може дуже негативно вплинути на якийсь час до інтерактивності навіть при поліпшенні першого відмальовування. Сторінки часто виглядають оманливо готовими та інтерактивними, але за фактом не можуть ніяк реагувати на дії користувача до виконання JS на стороні клієнта та приєднання обробників подій. Це може тривати кілька секунд або навіть хвилин на мобільних пристроях.

З всього матеріалу який був поданий вище, можна зробити висновок, що кожна технологія, має на меті виконання завдань, тобто є лише інструментом, а вже програміст може обирати серед інструментів той, який найкраще підходить під виконання його конкретного завдання. Як приклад для дуже великих сайтів в вас не вийде використати статистичний рендер, адже вам буде необхідно створити безліч HTML сторінок, і це просто фізично дуже важко зробити, не кажучи про те, що їх треба підтримувати й надалі.

Список використаних джерел

1. *Rendering on the web. URL: https://web.dev/rendering-on-the-web/*
2. *Populating the page:how browser works . URL: https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work*
3. *Rendering(Web development) Definition . URL: https://www.seobility.net/en/wiki/Rendering*
4. *Server-Side Rendering vs. Static Site Generation .URL: https://medium.com/better-programming/server-side-rendering-vs-static-site-generation*
5. *Webpage Rendering: How It Works + Tips on Optimization. URL: https://qarea.com/blog/webpage-rendering-how-it-works-tips-on-optimization*