УДК 004.42
DOI https://doi.org/10.32782/2663-5941/2025.6.2/11

Войтюк О.В.
Державний університет «Житомирська політехніка».

АНАЛІЗ МЕТОДІВ ТА ТЕХНОЛОГІЙ РЕНДЕРИНГУ ВЕБ-ЗАСТОСУНКІВ

Найбільш значущим та досліджуваним інноваційним питанням у сфері веб-розробки є вдосконалення методів рендерингу веб-застосунків. Вони є одним із ключових завдань сучасних інформаційних технологій та потребують системного вирішення. У статті запропоновано розглядати рендеринг як комплексну модель трансформації архітектурних та алгоритмічних рішень у високопродуктивний користувацький досвід. У роботі здійснено порівняльний аналіз основних підходів — клієнтського рендерингу (CSR), серверного рендерингу (SSR), статичної генерації (SSG) та інкрементальної регенерації (ISR), які формують основу сучасних веб-фреймворків. Показано, що клієнтський рендеринг забезпечує інтерактивність та модульність, але супроводжується проблемами продуктивності та SEO. Висвітлено, що серверний рендеринг дозволяє підвищити швидкість завантаження та якість індексації, однак вимагає значних серверних ресурсів і ретельної синхронізації стану. Показано, що SSG та ISR поєднують переваги статичних і динамічних стратегій, водночас створюють умови для масштабованості та ефективного оновлення контенту. Особливу увагу приділено роботі з деревами компонентів і механізмами управління станом (Redux, MobX, Vuex, Zustand), які визначають узгодженість між серверним та клієнтським середовищами. Висвітлено інноваційні напрями дослідження, зокрема прогресивну та селективну гідратацію, islands-архітектуру, стрімінговий серверний рендеринг. Доведено, що подальший розвиток веб-рендерингу пов'язано із гібридними моделями, які здатні забезпечити баланс між швидкодією, актуальністю даних і масштабованістю. Запропонований підхід може бути використаний під час проєктування корпоративних систем і високонавантажених платформ, де ефективність рендерингу безпосередньо впливає на якість бізнес-процесів, а також користувацький досвід.

Ключові слова: *веб-застосунки, рендеринг, CSR, SSR, SSG, ISR, гібридні підходи, управління станом, гідратація.*

Постановка проблеми. Стрімкий розвиток цифрових технологій і глобальна діджиталізація суспільства визначають нові вимоги до якості, швидкодії та масштабованості веб-застосунків. З огляду на значне зростання кількості користувачів та ускладнення інформаційних систем методи рендерингу вважаються ключовим інструментом забезпечення ефективної взаємодії людини з цифровим середовищем. Сучасні економічні та соціальні процеси (від електронної комерції до державних сервісів) поступово масштабуються та базуються на стабільній роботі веб-платформ. Це обумовлює потребу у виборі оптимальних стратегій відображення контенту та управління даними. Водночас різні підходи до рендерингу (клієнтський, серверний та гібридний) мають як переваги, так і суттєві обмеження, які вимагають їхнього ґрунтовного аналізу та удосконалення. Актуальність теми зумовлена необхідністю комплексного дослідження інноваційних технологій рендерингу, які здатні забезпечити баланс між продуктивністю, інтерактивністю та доступністю веб-застосунків, що безпосередньо впливатиме на якість бізнес-процесів, конкурентоспроможність компаній і комфорт користувачів у цифровому просторі.

Аналіз останніх досліджень і публікації. Питанням порівняльного аналізу стратегій рендерингу присвячено роботи закордонних науковців з цієї тематики – В. Чалла [8], Т. Іскандар, М. Лубіс, Т. Кусумасарі та А. Лубіс [15], а також Ф. Павіч і Л. Бркіч [25]. Вони систематизують переваги й обмеження клієнтського та серверного підходів. Архітектурні аспекти серверного рендерингу та його вплив на продуктивність корпоративних систем досліджують С. Женне [13], К. Валламсетла [33] та Х. Екпобімі [10]. Автори підкреслюють значущість SSR для SEO-оптимізації та масштабованості. Проблеми клієнтського рендерингу у сучасних мережевих застосунках висвітлюють К. Насенок і М. Войцеховська [22]. Гібридним підходам та інноваційним технологіям ренде-

рингу приділяють увагу Ф. Карвалью та П. Фіалю [6, 7], які досліджують прогресивний серверний рендеринг з асинхронними API, а також К. Чен [9], який аналізує модульний рендеринг та адаптивну гідратацію у React-застосунках. Порівняльні дослідження продуктивності сучасних веб-фреймворків представлено у працях Л. Курапаті [18], Р. Олліла, Н. Мякітало та Т. Мікконен [23]. Однак досліджувана проблематика недостатньо висвітлено в сучасних українських наукових дослідженнях. Потребують подальшого системного дослідження питання комплексної оптимізації гібридних моделей рендерингу, які поєднують переваги різних стратегій для забезпечення балансу між продуктивністю, масштабованістю та якістю користувацького досвіду у високонавантажених веб-застосунках.

Постановка завдання. Метою статті є висвітлення результатів дослідження сучасних методів рендерингу веб-застосунків, порівняння їхніх можливостей та обмежень.

Виклад основного матеріалу. Упродовж останніх років спостерігається стрімкий розвиток методів рендерингу веб-застосунків, які спрямовані на підвищення продуктивності, масштабованості та якості користувацького досвіду. Сучасні веб-застосунки використовують різноманітні стратегії рендерингу: клієнтський рендеринг (CSR - Client-Side Rendering), серверний рендеринг (SSR - Server-Side Rendering), статичну генерацію сайтів (SSG - Static Site Generation), інкрементальну статичну регенерацію (ISR - Incremental Static Regeneration), прогресивну гідратацію, стрімінговий SSR та edge-side рендеринг (ESR) [8; 13; 15; 22]. Кожен підхід має свої переваги та недоліки стосовно продуктивності, оптимізації для пошукових систем (SEO - Search Engine Optimization), масштабованості та складності впровадження.

Клієнтський рендеринг (CSR) – це підхід, який передбачає, що основна логіка рендерингу веб-застосунку виконується у браузері користувача, а не на сервері. CSR забезпечує модульність, гнучкість і швидку розробку, використовуючи компонентну архітектуру та сучасні фреймворки (React, Vue, Angular). Він сприяє створенню застосунків на основі компонентів [22]. Важливою перевагою є висока інтерактивність. Динамічне оновлення інтерфейсу без повного перезавантаження сторінки забезпечує плавний користувацький досвід. До того ж, ефективне управління станом досягається за допомогою сучасних підходів, таких як: Flux, Redux та реактивне програмування, що дозволяють централізовано керувати станом застосунку та підвищує його надійність і масштабованість [8].

Обмеження CSR полягають у складнощах з SEO та індексацією, тому що контент генерується в браузері користувача (на клієнті). Це може призводити до некоректного відображення сторінок у пошукових системах. Додатковою проблемою є великі розміри файлів. Для повноцінної роботи застосунку потрібно завантажувати значний обсяг JavaScript, що негативно впливає на продуктивність у повільних мережах або слабких пристроях. Також виникають ризики безпеки через експонування логіки застосунку та API в браузері, що підвищує вразливість до атак.

Серверний рендеринг (SSR) як ключова технологія для підвищення продуктивності, SEO та масштабованості сучасних веб-застосунків, насамперед для великих платформ і корпоративних рішень – це підхід, за якого HTML-код сторінки генерується на сервері, а не в браузері користувача [10; 18]. Його ключові переваги полягають у значному покращенні продуктивності, внаслідок того, що цей підхід зменшує час до першого відображення контенту (Time-to-First-Byte, First Contentful Paint). Також перевагою є SEO та індексація. Сервер одразу повертає повноцінний HTML, завдяки чому пошукові системи коректно індексують сторінки, водночас підвищують видимість сайту. До того ж, SSR забезпечує масштабованість, у такий спосіб дозволяючи обслуговувати тисячі спільних користувачів, насамперед за умови використання сучасних фреймворків, таких як Next.js, Nuxt.js чи Angular. Сучасні технології SSR включають гібридні моделі, які поєднують цей підхід з інкрементальною статичною генерацією (ISR), статичною генерацією (SSG) та клієнтською інтерактивністю. Це дозволяє збалансувати швидкість, динамічність і SEO [10; 13; 33]. Також важливим напрямом досліджуваного питання є прогресивний SSR, який, завдяки асинхронним шаблонам і поступовому рендерингу, дозволяє відображати контент відповідно до надходження даних. У такий спосіб це покращує досвід користувачів, навіть у випадку складних запитів [6; 7; 9]. Додатково використовується модульна гідратація, що забезпечує гнучке керування відновленням роботи компонентів залежно від пристрою, мережі та їхньої важливості, а також зменшуючи навантаження на клієнтів.

Водночас SSR має свої обмеження. Однією з основних проблем є складність синхронізації стану між сервером і клієнтом, яка потребує ретельної реалізації, щоб уникнути розбіжнос-

тей під час гідратації [33; 9]. Крім того, спостерігаються високі витрати на серверні ресурси за великої кількості синхронних запитів, що потребує застосування кешування та CDN [10; 13; 21]. Також існують потенційні ризики безпеки, якщо не дотримуватися сучасних практик захисту [26]. Впровадження SSR як ефективного підходу для створення швидких SEO-оптимізованих і масштабованих веб-застосунків, вимагає врахування складності синхронізації стану та оптимізації серверних ресурсів і безпеки, але сучасні фреймворки та архітектурні патерни значно спрощують ці завдання [4; 9; 10; 13; 26; 33].

Гібридні стратегії (ISR, SSG) поєднують переваги SSR і CSR – швидке завантаження статичних сторінок, можливість динамічного оновлення контенту та масштабованість. ISR дозволяє оновлювати сторінки на сервері без повної перегенерації, а SSG – попередньо генерувати сторінки під час збірки. Це знижує навантаження на сервер і забезпечує високу продуктивність для великої кількості користувачів [8; 13], водночас поєднує швидкість статичної генерації (сторінки доставляються з CDN) із динамічністю серверного рендерингу (оновлення контенту на сервері), що гарантує високу продуктивність навіть для великих застосунків. Важливою перевагою також є оптимізація ресурсів, тому, що оновлюються лише змінені сторінки, що зменшує навантаження на сервер і дозволяє ефективно обробляти високий трафік [38; 20].

Однак впровадження ISR супроводжується низкою певних проблем. Зокрема, виникає складність налаштування механізмів оновлення, що необхідно для уникнення розбіжностей у даних. Крім того, існує потреба у додаткових заходах безпеки через динамічне оновлення контенту. Також слід враховувати витрати на інфраструктуру, які для великих проектів вимагають оптимізації кешування та балансування. Таким чином, ISR у веб-розробці постає як гнучкий та високопродуктивний метод рендерингу, який балансує між статичністю та динамічністю, а аналогічні гібридні підходи у суміжних сферах, зокрема в Image Super-Resolution, підтверджують ефективність комбінування стратегій для досягнення оптимального співвідношення якості, швидкості й використання ресурсів [3; 20; 38]. Гібридний метод рендерингу ISR – є сучасною стратегією, що забезпечує високу продуктивність, масштабованість і динамічність контенту.

Гібридний підхід до рендерингу SSG поєднує переваги попередньої генерації сторінок із динамічними або інтерактивними можливостями, що дозволяє створювати швидкі, масштабовані та сучасні веб-застосунки. Основні його принципи полягають у попередній генерації контенту. HTML-сторінки створюються під час збірки, що забезпечує миттєве завантаження та високу продуктивність. За такої умови гібридний підхід дозволяє додавати динамічні елементи за допомогою клієнтського JavaScript, API-запитів або інкрементального оновлення сторінок, зокрема через ISR [8]. Унаслідок використання компонентної архітектури та можливості поєднання SSG із SSR чи CSR досягається модульність і масштабованість, що створює гнучкі сценарії використання. Практична реалізація цього підходу забезпечує покращений користувацький досвід (основний контент відображається швидко, а динамічні частини підвантажуються асинхронно), завдяки цьому підвищується інтерактивність. Водночас попередньо згенерований HTML оптимізує SEO, а гібридні механізми дозволяють оновлювати контент без потреби у повній перегенерації сайту. Це створює гнучкість для складних застосунків, де можна комбінувати SSG із SSR, ISR чи клієнтським рендерингом, адаптуючи методи до статичних, динамічних або персоналізованих сторінок [8]. Подібні гібридні підходи застосовуються і в суміжних галузях, зокрема у 3D-рендерингу [35].

Усі розглянуті підходи базуються на компонентній моделі, де дерево компонентів визначає структуру UI. У CSR дерево формується на стороні клієнта, в SSR – на сервері, в гібридних підходах – комбіновано. Управління станом (state management) критично важливе для синхронізації між сервером і клієнтом, насамперед під час гідратації. З цією метою використовуються Redux, Context API, MobX, а також оптимізації для мінімізації повторних рендерів і ефективної роботи з великими деревами компонентів [30; 31]. Відповідно, сучасні веб-фреймворки базуються на компонентній моделі, де дерево компонентів визначає структуру користувацького інтерфейсу незалежно від конкретної технології реалізації. Загальною концепцією є проміжний шар абстракції між компонентами застосунку та браузерним DOM, що дозволяє оптимізувати оновлення інтерфейсу.

Алгоритм узгодження є критичним компонентом для фреймворків з Virtual DOM. Класичний алгоритм порівняння дерев має складність $O(n^3)$, що є неприйнятним для реальних застосунків. Сучасні фреймворки реалізують евристичні O(n) алгоритми. Вони базуються на двох припущеннях – елементи різних типів генерують різні

дерева, а розробники можуть надавати підказки через ключі (key prop) для ефективного відстеження елементів. Процес узгодження включає порівняння попереднього та нового дерева, ідентифікацію мінімального набору змін та застосування лише необхідних оновлень до реального DOM. React Fiber представляє еволюцію цього підходу. Він впроваджує інкрементальний рендеринг через розбиття роботи на малі одиниці, що дозволяє призупиняти, відновлювати та пріоритизувати оновлення, водночас підтримуючи дві версії дерева – current tree та work-in-progress tree [1]. Vue використовує подібну стратегію з реактивною системою на основі Proxy, що забезпечує автоматичне відстеження залежностей.

У клієнтському рендерингу (CSR) дерево компонентів формується повністю в браузері після завантаження JavaScript. Сервер надсилає мінімальний HTML-документ, браузер завантажує ресурси, фреймворк ініціалізується та будує дерево компонентів, динамічно рендерячи інтерфейс. Користувацькі взаємодії викликають зміни стану компонентів, що призводить до часткового повторного рендерингу без перезавантаження сторінки. Управління станом у CSR досягається через спеціалізовані бібліотеки. Серверний рендеринг (SSR) вимагає складнішої архітектури через необхідність синхронізації між сервером та клієнтом. Код фреймворку виконується на сервері, дерево компонентів рендериться у HTML з початковим станом, після чого сервер надсилає повністю сформований HTML браузеру. Коли клієнт отримує HTML разом з JavaScript, відбувається процес гідратації (hydration) – універсальна концепція для SSR-фреймворків (Next.js для React, Nuxt для Vue, SvelteKit для Svelte, Angular Universal). В межах процесу фреймворк реконструює своє внутрішнє представлення компонентів на стороні клієнта, прикріплює обробники подій та робить сторінку інтерактивною [12]. Гідратація вимагає узгодження між серверно-рендереним DOM та клієнтським представленням для виявлення розбіжностей. Критичною проблемою є те, що вся сторінка залишається невідгукуючою до завершення гідратації, внаслідок того, що інтерактивні елементи не можуть реагувати на взаємодії до прикріплення обробників подій. Vue автоматично намагається відновити та скоригувати pre-rendered DOM відповідно до клієнтського стану під час виявлення невідповідностей, хоча це призводить до втрат продуктивності [29].

Синхронізація стану між сервером та клієнтом є фундаментальною проблемою SSR незалежно від фреймворку. Дані, які отримано на сервері для початкового рендерингу, мають бути доступні на стороні клієнта для гідратації без повторного завантаження. Загальноприйнятий підхід передбачає серіалізацію стану у глобальну змінну (window.INITIAL_STATE, window.NUXT, тощо) в HTML-рядку, яку клієнт використовує для ініціалізації сховища управління станом. Сучасні фреймворки впроваджують революційні можливості для SSR. Зокрема, Streaming HTML дозволяє починати відправку HTML до завершення завантаження всіх даних, розбиваючи HTML на фрагменти для прогресивного рендерингу браузером, що забезпечує швидший First Contentful Paint [32]. Selective Hydration надає пріоритет гідратації частинам, з якими взаємодіє користувач, за такої умови створюючи ілюзію миттєвої гідратації. React 18 реалізує це через Suspense, Next.js 15 використовує Server Components з автоматичним streaming, а Vue 3 впроваджує async components з подібною функціональністю [27].

Гібридні підходи (ISR, SSG) поєднують переваги різних стратегій рендерингу та підтримуються усіма основними meta-фреймворками. Прогресивна та селективна гідратація представляють новий етап еволюції веб-рендерингу. Так, Islands Architecture є framework-agnostic підходом, який розбиває сторінку на незалежні «острівці» інтерактивності, що оточені статичним HTML. Astro підтримує React, Vue, Svelte та Solid синхронно з директивами client:visible, client:idle, client:load для контролю гідратації. Fresh використовує Preact з islands-first підходом, Marko впроваджує власну реалізацію, а Eleventy інтегрує islands через web components [16; 17]. На відміну від монолітних JavaScript-бандлів, «острівці» завантажуються паралельно та гідруються ізольовано, при цьому компоненти з низьким пріоритетом не блокують критичні. Прогресивна гідратація фокусується на критичних компонентах, рендерує статичний контент на сервері та гідрує лише інтерактивні елементи post-load [11].

React Server Components (RSC) представляють гібридний підхід, який поєднує серверо-центричну модель SSR з клієнто-центричною моделлю CSR. Він дозволяє створювати змішане дерево компонентів з серверними та клієнтськими компонентами. Цей підхід забезпечує виконання частини логіки та рендерингу безпосередньо на сервері, що суттєво зменшує навантаження на клієнтське середовище та прискорює завантаження застосунку. Практичне застосування RSC демонструє його ефективність у системах із висо-

кими вимогами до динамічності та персоналізації інтерфейсу. Водночас RSC вимагає ретельного розподілу логіки між серверними та клієнтськими компонентами, який вимагає від розробників глибокого розуміння архітектурних особливостей гібридного рендерингу та механізмів синхронізації стану [24]. Узагальнену схему (розроблено автором) процесу гідратації веб-застосунків (Hydration) зображено на рисунку 1.

SSR, насамперед з використанням сучасних фреймворків (наприклад, Next.js), демонструє значні переваги для високонавантажених застосунків. Спостерігається покращення Time-to-First-Byte, зниження bounce rate та підвищення SEO [13; 15]. CSR забезпечує гнучкість, модульність і швидкість розробки, але може призводити до збільшення розміру файлів, проблем із SEO та продуктивністю на слабких пристроях [15; 22; 25]. Гібридні та інкрементальні підходи дозволяють поєднувати переваги SSR і CSR.

У питанні, яке пов'язано із конкретним вибором фреймворків та архітектур, слід зазначити, що популярні фреймворки (React, Angular, Vue, Svelte, Blazor) мають різні стратегії рендерингу, що впливають на масштабованість і продуктивність. Найкращі результати досягаються за рахунок компіляційних оптимізацій та реактивних моделей [23; 25]. Для великих корпоративних систем актуальні архітектури з динамічною інклюзією віддалених компонентів (composable renderer services) [28].

Висновки. Проведене дослідження дозволяє зробити висновки, що ефективність веб-платформ безпосередньо залежить від обраної стратегії відображення контенту та управління даними. Жоден із відомих підходів не може бути визнаний універсальним рішенням для всіх типів застосунків. Найбільший потенціал для створення високопродуктивних веб-застосунків демонструють гібридні стратегії, зокрема статична генерація сайтів та інкрементальна статична регенерація. Особливу увагу приділено ролі компонентної архітектури та механізмів управління станом у забезпеченні узгодженості між різними рівнями рендерингу. Правильна організація дерева компонентів і вибір відповідних інструментів управління станом визначають загальну ефективність застосунку та якість користувацького досвіду. Подальший розвиток досліджень пов'язаний з поглибленим вивченням гібридних моделей рен-

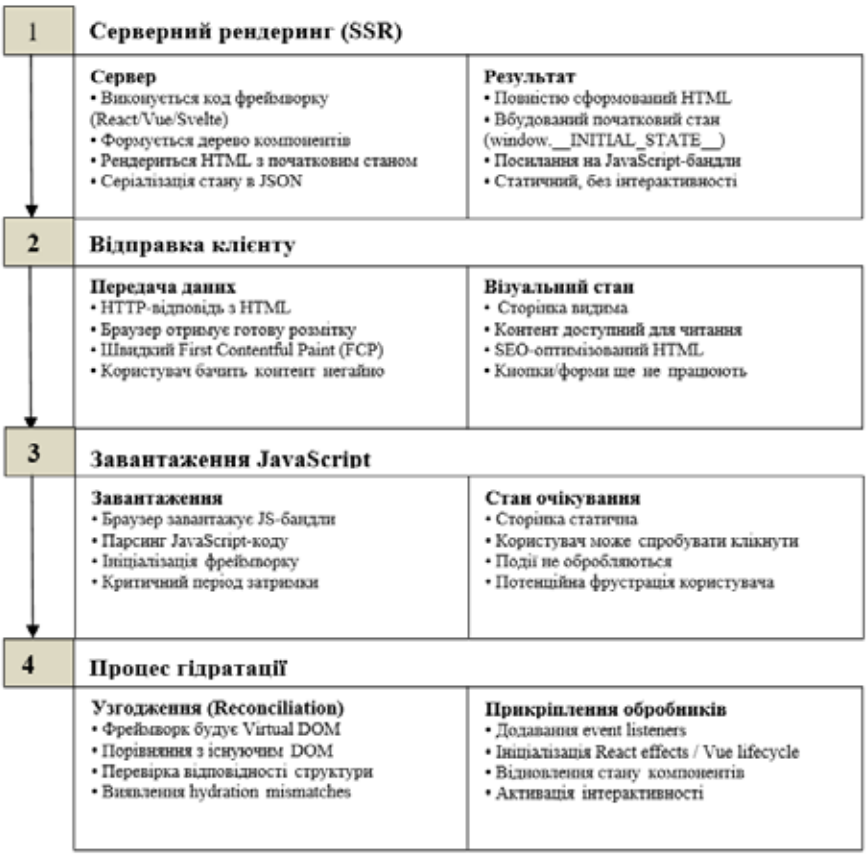

Рис. 1. Узагальнена схема процесу гідратації (Hydration) веб-застосунків

дерингу та розробкою методик автоматизованого вибору оптимальної стратегії залежно від специфіки конкретного застосунку. Важливим завданням залишається розробка комплексних методик оцінювання та прогнозування продуктивності різних підходів рендерингу на етапі проєктування інформаційних систем, що дозволить обґрунтовано приймати архітектурні рішення для корпоративних платформ і високонавантажених вебсервісів.

Список літератури:

1. Kalyanaraman K. A deep dive into React Fiber. *LogRocket.* URL: https://blog.logrocket.com/deep-dive-react-fiber/.
2. Abdallah A., Anadani S., Ismail H. Internet-Based 3D Volumes with Signed Distance Fields: Establishing a WebGL Rendering Infrastructure. *Proceedings of the International Conference on Computer and Applications (ICCA)*. 2024. Pp. 1–6. DOI: https://doi.org/10.1109/ICCA62237.2024.10927852.
3. Badiy M., Amounas F., Azrour M., Hammoudeh M. Advanced image super-resolution using deep learning approaches. *Radioelectronic and Computer Systems.* 2025. Is. 1. Pp. 187–198. DOI: https://doi.org/10.32620/reks.2025.1.13.
4. Bekmanova G., Yergesh B., Omarbekova A. et. al. Requirements for the Development of a Website Builder with Adaptive Design. *Proceedings of the 9th International Conference on Computer Science and Engineering (UBMK)*. 2024. Pp. 265–270. DOI: https://doi.org/10.1109/UBMK63289.2024.10773412.
5. Boutsi A., Ioannidis C., Verykokou S. Multi-Resolution 3D Rendering for High-Performance Web AR. *Sensors*. Basel, Switzerland. 2023. №23(15). DOI: https://doi.org/10.3390/s23156885.
6. Carvalho F.M. Progressive Server-Side Rendering with Suspendable Web Templates. *Web Information Systems Engineering – WISE 2024. 2024.* Vol. 15440. DOI: https://doi.org/10.1007/978-981-96-0576-7_33.
7. Carvalho F., Fialho P. Enhancing SSR in Low-Thread Web Servers: A Comprehensive Approach for Progressive Server-Side Rendering with any Asynchronous API and Multiple Data Models: *Proceedings of the 19th International Conference on Web Information Systems and Technologies*. 2023. Pp. 37–48. DOI: https://doi.org/10.5220/0012165300003584.
8. Challa V. Comprehensive Analysis of Modern Application Rendering Strategies: Enhancing Web and Mobile User Experiences. *Journal of Engineering and Applied Sciences Technology*. 2022. Vol. 4. Is. 4. Pp. 1–6. DOI: https://doi.org/10.47363/JEAST/2022(4)248.
9. Chen K. Improving Front-end Performance through Modular Rendering and Adaptive Hydration (MRAH) in React Applications. 2025. DOI: https://doi.org/10.48550/ARXIV.2504.03884.
10. Harrison Oke Ekpobimi. Building high-performance web applications with NextJS. *Computer Science & IT Research Journal*. 2024. Is. 5(8). Pp. 1963–1977. DOI: https://doi.org/10.51594/csitrj.v5i8.1459.
11. Exploring Web Rendering: Progressive Hydration. Babbel Magazine. URL: https://surl.li/xdxlso.
12. Frontend SSR frameworks benchmarked: Angular, Nuxt, NextJs and SvelteKit. URL: https://surl.li/egijxr.
13. Genne S. Optimizing Enterprise Web Performance Through Server-Side Rendering: A Next.Js Framework Implementation. *European Modern Studies Journal*. 2025. Vol. 9. Is. 3. Pp. 255–264. DOI: https://doi.org/10.59573/emsj.9(3).2025.21.
14. Hamzaturrazak M., Jonemaro E., Pinandito A. Performance Analysis of 3D Rendering Method on Web-Based Augmented Reality Application Using WebGL and OpenGL Shading Language. *Proceedings of the 8th International Conference on Sustainable Information Engineering and Technology*. 2023. Pp. 637–643. DOI: https://doi.org/10.1145/3626641.3626949.
15. Iskandar T., Lubis M., Fabrianti K. T., Ridho Lubis A. Comparison between client-side and server-side rendering in the web development. *IOP Conference Series: Materials Science and Engineering*. 2020. Vol. 801. DOI: https://doi.org/10.1088/1757-899X/801/1/012136.
16. Islands Architecture Definition. Sanity. DOI: https://surl.li/dkbgjr.
17. Islands Architecture. JuniorDev4Life. DOI: https://surl.li/ventzt.
18. Kurapati L. Different Types of Architectures in Frontend Design and Development. *International Journal For Multidisciplinary Research*. Is. 6(5). DOI: https://doi.org/10.36948/ijfmr.2024.v06i05.28707.
19. Li L., Huang, Y., Qiao X. et. al. Toward Distributed Collaborative Rendering Service for Immersive Mobile Web. *IEEE Network*. 2024. Vol. 38. Is. 3. Pp. 137–145. DOI: https://doi.org/10.1109/MNET.133.2200524.
20. Li S. The Application of Deep Learning in Image Super-resolution Reconstruction and Enhancement Technology. *Proceedings of the 2024 9th International Conference on Cyber Security and Information Engineering*. 2024. Pp. 350–355. DOI: https://doi.org/10.1145/3689236.3689246.
21. Lytvynov O., Hruzin D. Methods for optimizing the loading and updating of web pages using cloud technologies. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2023. Is. 4, Pp. 40–50. DOI: https://doi.org/10.32782/IT/2023-4-5.
22. Nasenok K., Voitsekhovska M. Client-side rendering issues in the modern worldwide networ. *Technical sciences and technologies*. 2024. Is. 4(38), Pp. 197–207. DOI: https://doi.org/10.25140/2411-5363-2024-4(38)-197-207.
23. Ollila R., Mäkitalo N., Mikkonen T. Modern Web Frameworks: A Comparison of Rendering Performance. *Journal of Web Engineering*. 2022. №21(03), Pp. 789–814. DOI: https://doi.org/10.13052/jwe1540-9589.21311.

24. Patil O. Real Time Inventory Management System powered by Generative User Interface. *Interantional journal of scientific research in engineering and management*. 2024. Is. 8(4). Pp. 1–5. DOI: https://doi.org/10.55041/IJSREM32623.
25. Pavić F., Brkić L. Methods of Improving and Optimizing React Web-applications. *Proceedings of the 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*. 2021. Pp. 1753–1758. DOI: https://doi.org/10.23919/MIPRO52101.2021.9596762.
26. Pellegrino G., Catakoglu O., Balzarotti D., Rossow C. Uses and Abuses of Server-Side Requests. *Research in Attacks, Intrusions, and Defenses.* 2016. Vol. 9854, Pp. 393–414. DOI: https://doi.org/10.1007/978-3-319-45719-2_18.
27. React 18: Partial Hydration. *Medium*. URL: https://surl.li/ivdlzl.
28. Sathyakumar D. Composable Renderer Services: A Study of Architectures for Dynamic View Transclusion in Distributed Web Applications. *2024 IEEE Cloud Summit*. 2024. Pp. 178–187. DOI: https://doi.org/10.1109/Cloud-Summit61220.2024.00036.
29. Server-Side Rendering (SSR). Vue.org. DOI: https://surl.lt/vuyrkw.
30. Sun Y. Server-Side Rendering. *Practical Application Development with AppRun.* 2019. Pp. 191–217. DOI: https://doi.org/10.1007/978-1-4842-4069-4_9.
31. Thakkar M. Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications. Apress. 2020. DOI: https://doi.org/10.1007/978-1-4842-5869-9.
32. Sumit S. The Next.js 15 Streaming Handbook — SSR, React Suspense, and Loading Skeleton. URL: https://surl.li/ozqdcc.
33. Vallamsetla K. The Impact of Server-Side Rendering on UI Performance and SEO. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. 2024. Is. 10(5). Pp. 795–804. DOI: https://doi.org/10.32628/CSEIT241051067.
34. Wang Z., Yan B., Xing S. et. al. Research on large-scale 3D scene rendering method for web application. *Eighth Symposium on Novel Photoelectronic Detection Technology and Applications*. SPIE. 2022. P. 731. DOI: https://doi.org/10.1117/12.2627369.
35. Xuan J., Huang Y., Guo Y., Jia B. Image-Based 3D Gaussian Rendering: A Hybrid Approach for Scene Visualization. *2024 12th International Conference on Information Systems and Computing Technology (ISCTech)*. 2024. Pp. 1–5. DOI: https://doi.org/10.1109/ISCTech63666.2024.10845536.
36. Ye L., Zhou C., Peng H. et. al. Multi-level feature interaction image super-resolution network based on convolutional nonlinear spiking neural model. *Neural Networks.* 2024. DOI: https://doi.org/10.1016/j.neunet.2024.106366.
37. Yu G., Liu C., Fang T. et. al. A survey of real-time rendering on Web3D application. *Virtual Reality & Intelligent Hardware*, 2023. Vol. 5. Is. 5, Pp. 379–394. DOI: https://doi.org/10.1016/j.vrih.2022.04.002.
38. Yu M., Shi J., Xue C. et. al. A review of single image super-resolution reconstruction based on deep learning. *Multimedia Tools and Applications*. 2023. Vol. 83. Is. 18. Pp. 55921–55962. DOI: https://doi.org/10.1007/s11042-023-17660-4.

Voitiuk O.V. ANALYSIS OF METHODS AND TECHNOLOGIES FOR WEB APPLICATION RENDERING

The most significant and actively studied innovative issue in the field of web development is the improvement of web application rendering methods. These methods represent one of the key challenges in modern information technologies and require a systematic approach to their optimization. This paper proposes to consider rendering as a complex model of transforming architectural and algorithmic solutions into a high-performance user experience. The study provides a comparative analysis of the main rendering approaches — Client-Side Rendering (CSR), Server-Side Rendering (SSR), Static Site Generation (SSG), and Incremental Static Regeneration (ISR) — which form the foundation of contemporary web frameworks. It is shown that CSR ensures interactivity and modularity but is often associated with performance and SEO challenges; SSR improves loading speed and indexing quality, although it requires considerable server resources and careful state synchronization; SSG and ISR combine the benefits of static and dynamic strategies, creating favorable conditions for scalability and efficient content updates. Special attention is given to the operation of component trees and state management mechanisms (Redux, MobX, Vuex, Zustand), which determine the consistency between server and client environments. The paper also highlights innovative research directions such as progressive and selective hydration, islands architecture, and streaming SSR. It is concluded that the future of web rendering lies in hybrid models capable of achieving a balance between performance, data freshness, and scalability. The proposed approach can be applied in the design of corporate systems and high-load platforms, where rendering efficiency directly affects business process quality and user experience.

Key words: *web applications, rendering, CSR, SSR, SSG, ISR, hybrid approaches, state management, hydration.*

Дата надходження статті: 06.11.2025
Дата прийняття статті: 25.11.2025
Опубліковано: 30.12.2025