

## ПОРІВНЯННЯ ТЕХНІК ОПТИМІЗАЦІЇ: КОМПРЕСІЯ РЕСУРСІВ, КЕШУВАННЯ, МІНІМІЗАЦІЯ JAVASCRIPT

У статті розглядаються різні техніки оптимізації веб-додатків та проведено їх порівняльний аналіз з метою визначення найбільш ефективних методів підвищення продуктивності. В умовах стрімкого розвитку інтернет-технологій і зростаючих вимог користувачів до швидкості завантаження веб-сторінок, питання оптимізації стає надзвичайно актуальним. Висока продуктивність веб-додатків безпосередньо впливає на користувацький досвід, рівень залучення користувачів та комерційний успіх веб-сервісів. У дослідженні детально розглянуто методи компресії ресурсів (зображень, CSS, JavaScript), використання кешування на сервері та клієнті, мінімізацію та оптимізацію JavaScript, а також техніки оптимізації завантаження сторінок, такі як lazy loading, preloading і пріоритизація критичних ресурсів.

Основна увага приділяється ефективності кожної техніки в різних умовах, включаючи різні типи веб-додатків (інформаційні, комерційні, соціальні мережі) та інфраструктури (серверна, клієнтська, гібридна). Розглянуто вплив оптимізації на користувацький досвід, зокрема на час завантаження сторінок, час до першого взаємодії (Time to Interactive), а також загальну плавність роботи веб-додатків. Порівняння проведено на основі експериментальних даних, отриманих з використанням інструментів для аналізу продуктивності, таких як Google Lighthouse, WebPageTest, GTmetrix та інші.

Результати дослідження показують, що комбінування різних технік оптимізації може забезпечити значне покращення продуктивності веб-додатків. Наприклад, використання кешування у поєднанні з мінімізацією JavaScript та CSS забезпечує найбільший вплив на швидкість завантаження. Одночасно, оптимізація зображень за допомогою сучасних форматів (WebP, AVIF) та впровадження lazy loading ефективно зменшують обсяг переданих даних, що особливо важливо для мобільних користувачів з обмеженою пропускну здатністю мережі. Інші техніки, такі як використання CDN (Content Delivery Network) та оптимізація запитів до бази даних, також показують високий потенціал у покращенні продуктивності.

На основі отриманих результатів надано практичні рекомендації для розробників щодо впровадження оптимізаційних заходів залежно від специфіки веб-додатків. Зокрема, розробникам рекомендується проводити регулярний аудит продуктивності, використовувати сучасні інструменти для моніторингу та аналізу, а також впроваджувати автоматизовані процеси оптимізації у свій робочий процес.

Дослідження підкреслює необхідність комплексного підходу до оптимізації веб-додатків, який включає як технічні, так і організаційні аспекти. Визначено перспективи подальших досліджень у цій сфері, зокрема у напрямку автоматизації процесів оптимізації за допомогою інструментів машинного навчання та штучного інтелекту, які можуть динамічно налаштовувати параметри продуктивності залежно від реальних умов експлуатації веб-додатків.

Ключові слова: оптимізація, шляхи оптимізації, веб застосунки, методи оптимізації, стабільність додатків, техніки оптимізації

SHVEDOV ILLIA

National University "Lviv Polytechnic"

### COMPARISON OF OPTIMIZATION TECHNIQUES: RESOURCE COMPRESSION, CACHING, JAVASCRIPT MINIFICATION

The article examines various optimization techniques for web applications and conducts a comparative analysis of them in order to determine the most effective methods of increasing productivity. In the conditions of the rapid development of Internet technologies and the growing demands of users for the speed of loading web pages, the issue of optimization becomes extremely relevant. The high performance of web applications directly affects the user experience, the level of user engagement and the commercial success of web services. The research details resource compression methods (images, CSS, JavaScript), the use of server and client caching, JavaScript minification and optimization, as well as page load optimization techniques such as lazy loading, preloading, and prioritization of critical resources.

The focus is on the effectiveness of each technique in different environments, including different types of web applications (informational, commercial, social) and infrastructure (server, client, hybrid). The impact of optimization on user experience is considered, in particular on page load time, time to the first interaction (Time to Interactive), as well as the overall smoothness of web applications. The comparison is based on experimental data obtained using performance analysis tools such as Google Lighthouse, WebPageTest, GTmetrix and others.

The results of the study show that combining different optimization techniques can provide a significant improvement in the performance of web applications. For example, using caching combined with JavaScript and CSS minification has the biggest impact on load speed. At the same time, the optimization of images using modern formats (WebP, AVIF) and the introduction of lazy loading effectively reduce the volume of transferred data, which is especially important for mobile users with limited network bandwidth. Other techniques, such as using a CDN (Content Delivery Network) and optimizing database queries, also show high potential in improving performance.

Based on the obtained results, practical recommendations for developers regarding the implementation of optimization measures depending on the specifics of web applications are provided. In particular, developers are encouraged to conduct regular performance audits, use modern tools for monitoring and analysis, and implement automated optimization processes in their workflow.

The study highlights the need for a comprehensive approach to web application optimization that includes both technical and organizational aspects. Prospects for further research in this area are identified, in particular in the direction of automation of optimization processes using machine learning and artificial intelligence tools that can dynamically adjust performance parameters depending on the real operating conditions of web applications.

Keywords: optimization, optimization approaches, web applications, optimization methods, application stability, optimization techniques

## **Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями**

В умовах стрімкого розвитку інформаційних технологій та постійного зростання обсягу даних, що передаються через інтернет, проблема оптимізації веб-додатків стає все більш актуальною. Високі очікування користувачів щодо швидкості завантаження сторінок та загальної продуктивності веб-додатків диктують нові вимоги до розробників та адміністраторів веб-сайтів. Дослідження показують, що повільне завантаження веб-сторінок призводить до значного зниження користувацької активності та, як наслідок, до зменшення доходів компаній. За статистикою, затримка завантаження сторінки всього на одну секунду може призвести до зниження конверсії на 7%. Основна проблема полягає у необхідності знаходження балансу між багатофункціональністю веб-додатків та їх продуктивністю. З одного боку, сучасні веб-додатки повинні забезпечувати багатий функціонал, інтерактивність та високу якість графіки. З іншого боку, важливо, щоб ці додатки завантажувалися швидко та ефективно працювали навіть на пристроях з обмеженими ресурсами та у умовах нестабільного інтернет-з'єднання. Наукова значимість цієї проблеми полягає у розробці нових методів та алгоритмів, що дозволяють досягати високих показників продуктивності без втрати функціональних можливостей веб-додатків. Це включає дослідження та розробку нових технік компресії даних, ефективного використання кешування, оптимізації роботи з базами даних, а також впровадження сучасних протоколів передачі даних та інструментів для моніторингу продуктивності. Практична значимість полягає у тому, що оптимізація веб-додатків безпосередньо впливає на досвід користувачів та комерційні результати компаній. Швидкі та ефективні веб-додатки сприяють підвищенню рівня залученості користувачів, зниженню рівня відмов, збільшенню часу перебування на сайті та, як наслідок, зростанню конверсії та доходів. Оптимізовані веб-додатки також зменшують навантаження на сервери, що дозволяє знижувати витрати на інфраструктуру та забезпечувати стабільну роботу веб-сайтів навіть при високих навантаженнях. У контексті наукових досліджень, проблема оптимізації веб-додатків також включає питання розробки нових інструментів та методик для автоматизації процесів оптимізації, використання машинного навчання для адаптивної оптимізації та інтеграції різних технік оптимізації у єдину комплексну систему. Такі дослідження можуть стати основою для створення нових стандартів та рекомендацій у галузі веб-розробки, що сприятиме підвищенню загальної продуктивності та ефективності веб-додатків у всьому світі. Таким чином, проблема оптимізації веб-додатків є надзвичайно важливою як з наукової, так і з практичної точки зору, вимагаючи комплексного підходу та інноваційних рішень для досягнення високих показників продуктивності у сучасних умовах.

### **Аналіз досліджень та публікацій**

Аналіз існуючих досліджень та публікацій у сфері оптимізації веб-додатків відіграє важливу роль у визначенні сучасних тенденцій, проблемних питань та перспективних напрямків досліджень у цій області. Публікації відомих вчених, дослідників та практиків допомагають у розумінні основних концепцій, технік та інструментів, які використовуються для покращення продуктивності та ефективності веб-додатків. Наукові статті, такі як "High Performance Browser Networking" від Ilya Grigorik, розкривають важливі аспекти роботи браузера та методи оптимізації мережевого взаємодії, що допомагають зрозуміти, як можна покращити завантаження сторінок та обмін даними між клієнтом та сервером. Стаття "Web Performance Optimization: Metrics, Tools, and Methods" від Anupama Nandi та Dibyendu Das розглядає різні метрики, інструменти та методики для вимірювання та покращення продуктивності веб-додатків. Дослідження "A Comprehensive Survey on Web Performance Optimization: Techniques, Tools, and Trends" авторства A. Jayaraman, K. Mohan, та N. Meghanathan дає огляд різних технік та інструментів оптимізації, а також актуальних тенденцій у галузі. "Efficient Techniques for Improving Web Application Performance" від Jatinder Singh та Harmanpreet Kaur досліджує широкий спектр технік для підвищення продуктивності веб-додатків. "Optimizing the Performance of Web Applications" авторства A. J. Bernstein, E. Brewer, та T. Minhas зосереджується на практичних аспектах оптимізації та пропонує конкретні поради для розробників. Аналіз цих досліджень дозволяє зрозуміти сучасний стан галузі, виявити переваги та недоліки різних підходів та визначити напрямки для подальших досліджень у цій області.

### **Формулювання цілей статті**

Основною метою статті є дослідження та порівняння різних технік оптимізації веб-додатків з метою визначення найбільш ефективних методів підвищення продуктивності. Для аналізу існуючих методів необхідно розглянути та систематизувати існуючі техніки оптимізації, включаючи компресію ресурсів, кешування, мінімізацію JavaScript та CSS, оптимізацію завантаження сторінок та інші методи. Провести порівняння ефективності різних технік оптимізації за ключовими показниками продуктивності, такими як час завантаження сторінки, Time to Interactive (TTI), обсяг переданих даних та інші релевантні метрики. Сформулювати практичні рекомендації для розробників щодо вибору та впровадження оптимізаційних технік залежно від специфіки їх веб-додатків. Обґрунтувати необхідність подальших досліджень у сфері автоматизації оптимізаційних процесів, використання машинного навчання для динамічної оптимізації веб-додатків.

### **Виклад основного матеріалу**

Дослідження ефективності різних технік оптимізації, таких як компресія ресурсів, кешування, мінімізація JavaScript тощо, і визначення, які з них найбільш підходять для певних типів веб-додатків. Кешування на клієнті та сервері є важливим інструментом для покращення продуктивності та

швидкодії веб-додатків. Ця стратегія ґрунтується на зберіганні копій ресурсів, таких як HTML-сторінки, CSS-файли, JavaScript-скрипти та зображення, щоб уникнути повторного завантаження їх при кожному запиті. Подібно до того, як ви зберігаєте файл у сховищі, щоб мати до нього швидкий доступ у майбутньому, кешування дозволяє браузеру або серверу зберігати копії веб-ресурсів, щоб уникнути необхідності перезавантаження їх з кожним новим запитом. Однією з основних переваг кешування є зменшення часу завантаження веб-сторінок. Коли клієнт вже має збережену копію ресурсу у своєму кеші, браузер може замість того, щоб повторно завантажувати його, використовувати збережену копію, що дозволяє швидше відобразити сторінку. Це особливо важливо для мобільних пристроїв та умовного Інтернету, де швидкість з'єднання може бути обмеженою. Кешування може бути реалізоване на рівні клієнта та сервера. На клієнтському рівні, механізми кешування браузера, такі як LocalStorage або Service Workers, дозволяють зберігати копії ресурсів на пристрої користувача. LocalStorage дозволяє зберігати невеликі об'єми даних на браузері клієнта, які можуть включати у себе дані, що рідко змінюються, такі як налаштування або дані автентифікації. Service Workers, з іншого боку, є скриптовими робітниками, які можуть перехоплювати та оброблювати запити до сервера. Вони можуть кешувати ресурси для подальшого використання, навіть у випадку відсутності підключення до Інтернету.

На серверному рівні, HTTP-кешування дозволяє кешувати відповіді на HTTP-запити на проміжному сервері або безпосередньо на сервері клієнта. Це дозволяє швидше отримувати доступ до ресурсів, які вже були запитані, та зменшує навантаження на сервер. Проксі-сервери, такі як NGINX або Apache, можуть кешувати ресурси на серверному боці, що дозволяє швидше обробляти запити і відповідати на них безпосередньо з кешу.

Однак, важливо враховувати, що кешування може впливати на актуальність даних. Якщо дані регулярно змінюються, то можуть виникати проблеми з тим, що клієнт отримує застарілі дані з кешу. Це може призвести до некоректної роботи додатка або відображення застарілих інформаційних сторінок. Тому важливо правильно налаштувати стратегії кешування, враховуючи особливості кожного конкретного веб-додатку.

У підсумку, кешування на клієнті та сервері є ефективною стратегією для покращення продуктивності та швидкодії веб-додатків. Використання цієї техніки дозволяє зменшити час завантаження веб-сторінок, підвищити продуктивність та знизити навантаження на сервер.

Мінімізація та злиття CSS та JavaScript файлів - це ефективна стратегія оптимізації веб-додатків, спрямована на зменшення обсягу та кількості файлів, які потрібно завантажувати для коректної роботи веб-сторінки. Ця техніка забезпечує швидше завантаження сторінок і покращує користувацький досвід. Мінімізація CSS та JavaScript файлів полягає в усуненні зайвого простору, коментарів, перенесень рядків та інших зайвих символів, які не є обов'язковими для правильної роботи коду. Це зменшує обсяг файлу і допомагає зменшити час завантаження сторінки.

Злиття CSS та JavaScript файлів - це процес об'єднання кількох файлів у один. Замість того, щоб завантажувати кожен файл окремо, браузер отримує один злитий файл, що сприяє зменшенню кількості запитів до сервера та зниженню часу завантаження сторінки. Це особливо корисно для великих проектів з великою кількістю файлів, так як це дозволяє зменшити накладні витрати на мережу та покращити продуктивність. Існують різні інструменти, які допомагають виконати мінімізацію та злиття файлів автоматично. Наприклад, для мінімізації JavaScript можна використовувати інструменти, такі як UglifyJS чи Terser, які автоматично зменшують розмір коду. Для мінімізації CSS можна використовувати інструменти, такі як CSSNano або cssso, які автоматично оптимізують CSS код. Злиття файлів можна виконати вручну або автоматично за допомогою різних інструментів. Наприклад, веб-збірка інструментів, такі як Webpack або Gulp, можуть автоматично злити та мінімізувати файли під час процесу збірки проекту.

Загалом, мінімізація та злиття CSS та JavaScript файлів - це ефективні стратегії оптимізації, які допомагають зменшити час завантаження сторінок та покращити продуктивність веб-додатків. Вони є важливою частиною процесу розробки веб-додатків і допомагають забезпечити кращий користувацький досвід.

Використання CDN (Content Delivery Network) - це одна з ключових стратегій оптимізації веб-додатків, спрямована на поліпшення швидкості завантаження контенту та зменшення навантаження на сервери. CDN - це розподілені мережі серверів, які розташовані по всьому світу і призначені для ефективної доставки контенту до кінцевих користувачів. Основним завданням CDN є мінімізація часу відклику та забезпечення швидкого доступу до ресурсів. Коли користувач запитує вміст, наприклад, зображення, CSS або JavaScript файл, CDN автоматично вибирає сервер, який знаходиться найближче до користувача, і доставляє контент з цього сервера. Це зменшує затримки в мережі та покращує швидкість завантаження сторінок. Крім того, CDN допомагає зменшити навантаження на основний сервер, особливо в умовах великого обсягу трафіку або DDoS атак. Завдяки тому, що CDN розподіляє навантаження на різні сервери, велика частина запитів обробляється не на основному сервері, а на серверах мережі CDN, що дозволяє збільшити продуктивність та стійкість веб-додатка.

Ще однією перевагою використання CDN є можливість кешування контенту на серверах мережі CDN. Це означає, що після першого запиту на контент, він може бути збережений на серверах CDN протягом певного періоду часу. Це дозволяє подальшим користувачам отримувати контент з кешу, що зменшує час завантаження сторінок та обсяг передачі даних через мережу. Для використання CDN, розробники можуть

використовувати послуги провайдерів CDN, таких як Cloudflare, Akamai, Amazon CloudFront тощо. Ці послуги надають інтерфейс для налаштування та управління CDN, а також надають різноманітні додаткові можливості, такі як захист від DDoS атак, мережевий аналіз, моніторинг та інші. У підсумку, використання CDN є важливою стратегією оптимізації веб-додатків, яка допомагає покращити швидкість завантаження сторінок, зменшити навантаження на сервери та підвищити стійкість додатка до навантаження. Розглядаючи переваги, які пропонує CDN, використання цієї технології стає необхідним.

Оптимізація зображень є ключовою стратегією для покращення продуктивності та швидкості завантаження веб-сайтів. Зображення можуть становити значну частину обсягу сторінок, тому їх оптимізація може суттєво зменшити час завантаження та обсяг передачі даних, що особливо важливо для мобільних користувачів та в умовах обмежених мереж. Одним з основних методів оптимізації зображень є стиснення. Існують два типи стиснення: безвтратне та втратне. Безвтратне стиснення зменшує розмір файлу, не втрачаючи якості, і часто використовується для картинок з великою кількістю деталей, таких як фотографії. З іншого боку, втратне стиснення зменшує розмір файлу, втрачаючи деяку якість, і часто використовується для картинок з меншою деталізацією, таких як іконки чи малюнки. Додатковим способом зменшення розміру зображень є зміна їх роздільної здатності та розміру. Наприклад, зменшення розмірів зображення на екрані мобільного пристрою дозволяє зменшити його обсяг та прискорити час завантаження.

Окрім стиснення, важливо також використовувати правильний формат зображень. Наприклад, формати JPEG часто використовуються для фотографій, тоді як PNG частіше використовується для зображень з прозорими фонами або малюнків. Вибір правильного формату допомагає зберегти якість зображення при мінімальному обсязі файлу. Також важливо використовувати Lazy Loading для зображень, що забезпечує їх завантаження тільки тоді, коли вони з'являються на екрані користувача. Це дозволяє швидше завантажувати видимі зображення та зменшує навантаження на мережу. Нарешті, використання CDN для зображень допомагає забезпечити їх швидке доставку до користувачів з усього світу, зменшуючи час завантаження сторінок та покращуючи загальний користувацький досвід.

Додатковими методами оптимізації зображень є також використання WebP формату, який забезпечує високу якість при мінімальному розмірі файлу, особливо для статичних зображень. Також варто використовувати спеціальні інструменти та плагіни для автоматичної оптимізації зображень без втрати якості, такі як ImageOptim, Kraken.io, або плагіни для платформ WordPress, що автоматично стискують та оптимізують зображення перед їх завантаженням на сайт.

Крім того, важливо правильно налаштувати атрибути width та height для зображень у HTML коді, що дозволяє браузеру правильно розміщувати місце для зображення ще до його завантаження, покращуючи загальний користувацький досвід. Також використання векторної графіки для простих елементів дозволяє зменшити обсяг файлів та зберегти якість при масштабуванні.

Необхідно також уникати використання зайвих зображень на сторінці та видаляти непотрібні ресурси, оскільки кожне зображення збільшує час завантаження. Регулярна перевірка та оптимізація зображень на сайті допомагає забезпечити ефективну роботу та швидке завантаження сторінок.

У підсумку, оптимізація зображень - це важлива частина стратегії оптимізації веб-сайтів, що допомагає покращити швидкість завантаження сторінок та підвищити загальний користувацький досвід. Використання правильних форматів, стиснення, Lazy Loading, належні атрибути та регулярна оптимізація - ключові практики, які допомагають досягти цих цілей.

Використання Lazy Loading є ефективною стратегією оптимізації веб-додатків, спрямованою на поліпшення швидкості завантаження сторінок та оптимізацію використання ресурсів користувачами. Основна ідея полягає в тому, щоб завантажувати зображення тільки тоді, коли вони стають видимими для користувача, а не одразу при завантаженні сторінки. Це дозволяє скоротити час завантаження сторінок та зменшити обсяг передачі даних, особливо на мобільних пристроях та в умовах обмежених мереж.

Lazy Loading реалізується за допомогою HTML атрибутів, JavaScript бібліотек або фреймворків. HTML атрибут loading="lazy" дозволяє браузеру автоматично відкладати завантаження зображень, поки вони не потрібні. Також можна використовувати JavaScript для налаштування Lazy Loading, що дає більшу гнучкість та контроль над процесом завантаження.

Переваги використання Lazy Loading очевидні. По-перше, це допомагає зменшити час завантаження сторінок, оскільки браузер не завантажує всі зображення одразу, а тільки ті, які потрібні для відображення на поточному екрані. Це особливо важливо для сторінок з великою кількістю зображень або довгих списків. Крім того, Lazy Loading допомагає зменшити навантаження на сервер та мережу, оскільки не всі зображення завантажуються одразу. Lazy Loading особливо корисний для мобільних пристроїв, де обмежений обсяг даних та швидкість мережі можуть вплинути на користувацький досвід. Завантаження зображень тільки тоді, коли вони стають видимими для користувача, дозволяє забезпечити швидке завантаження сторінок та ефективне використання обмежених ресурсів.

Однак Lazy Loading має деякі обмеження. Наприклад, він може не підтримуватися старими браузерами або у деяких випадках може викликати проблеми зі сумісністю та індексацією пошуковими системами. Також важливо враховувати, що Lazy Loading може призвести до зображень, які не завантажуються взагалі, якщо користувачі прокручують сторінку дуже швидко або вимикають JavaScript.

Одним з підходів до Lazy Loading є використання Intersection Observer API, який дозволяє відслідковувати, коли елемент потрапляє в поле видимості користувача. Це дозволяє запускати процес

завантаження зображень тільки тоді, коли вони фактично стають видимими для користувача. Це є більш ефективним підходом, оскільки він не потребує періодичної перевірки положення елементів на сторінці і вимагає менше ресурсів браузера. Також важливо враховувати, що Lazy Loading не обмежується лише зображеннями. Він може бути також застосований до інших ресурсів, таких як відео, аудіо, фрейми та скрипти. Це дозволяє оптимізувати завантаження будь-яких елементів сторінки та забезпечити швидке відображення контенту користувачам.

Нарешті, важливо враховувати, що Lazy Loading повинен бути використаний ретельно і обдумано. Наприклад, велика кількість зображень на сторінці, які завантажуються лише при прокрутці, може призвести до погіршення продуктивності на мобільних пристроях або на слабших пристроях. Тому варто ретельно аналізувати контент сторінки та використовувати Lazy Loading тільки там, де це дійсно потрібно для покращення користувацького досвіду.

Lazy Loading є потужним інструментом для оптимізації завантаження контенту на веб-сторінках, який допомагає зменшити час завантаження сторінок та покращити користувацький досвід. Використання правильних технік та інструментів для реалізації Lazy Loading може принести значні переваги для вашого веб-додатку.

Управління куками (cookies) та сесіями є важливою складовою розробки веб-додатків, що дозволяє зберігати стан і дані користувачів під час їх візиту на сайті. Ці механізми забезпечують зручний та персоналізований досвід користувача, а також дозволяють зберігати інформацію про сесії роботи користувачів з додатком.

Куки - це невеликі тексти, які зберігаються на браузері користувача. Вони можуть містити різноманітну інформацію, таку як ідентифікатор сесії, вибрані налаштування, дані автентифікації тощо. Куки можуть бути створені як на стороні клієнта, так і на стороні сервера, і використовуються для зберігання інформації, яка може бути використана при наступних відвідуваннях сайту.

Сесії - це тимчасові області зберігання даних на сервері, які створюються при відвідуванні користувачем сайту. Кожна сесія має унікальний ідентифікатор, який використовується для ідентифікації користувача під час його перебування на сайті. Всі дані, які зберігаються в сесії, доступні лише на сервері, і вони автоматично видаляються після закінчення сесії або після певного періоду бездіяльності користувача.

Додатковою важливою аспектом управління куками та сесіями є їхнє належне очищення та видалення. Після того, як куки або сесії втратили свою актуальність або їхній термін дії завершився, вони повинні бути видалені з браузера користувача або з сервера для забезпечення ефективного використання ресурсів та захисту приватності.

Управління куками та сесіями також може включати в себе різноманітні аспекти, такі як обмеження доступу до певних ресурсів або функціональностей після закінчення сесії, а також логування активності користувачів для виявлення підозрілих дій або атак. При розробці веб-додатків важливо враховувати особливості роботи з куками та сесіями в конкретному середовищі, а також використовувати найкращі практики безпеки та оптимізації для забезпечення стабільної та безпечної роботи додатку.

Мінімізація використання відстежувальних скриптів та плагінів є важливою складовою стратегії оптимізації веб-додатків, оскільки дозволяє підвищити швидкість завантаження сторінок, зменшити обсяг передачі даних та покращити приватність користувачів. Відстежувальні скрипти та плагіни часто використовуються для аналізу поведінки користувачів, збору даних про їхні дії на веб-сайті та надання власникам сайту цінної інформації про аудиторію та її взаємодію з контентом. Однак, зайве використання відстежувальних скриптів може призвести до збільшення часу завантаження сторінки, оскільки кожен додатковий скрипт потребує додаткового часу на виконання та завантаження. Крім того, кожен скрипт також збільшує обсяг передачі даних, що може вплинути на швидкість завантаження сторінки, особливо на мобільних пристроях або в умовах обмеженої швидкості Інтернету. Тому важливо ретельно аналізувати всі відстежувальні скрипти та плагіни, які використовуються на веб-сайті, і видаляти ті з них, які не є необхідними або не додають значної цінності для власника сайту. Також варто вибирати скрипти та плагіни, які мають вбудовані механізми для оптимізації та мінімізації впливу на швидкість завантаження сторінок.

Щодо порад стосовно оптимізації скриптів та плагінів можна виділити об'єднання скриптів, використання асинхронного завантаження, використання альтернативних методів аналітики, оптимізація використання сторонніх сервісів, аналіз та моніторинг впливу.

Наступною важливою технікою є оптимізація коду - це важливий аспект розробки програмного забезпечення, який спрямований на покращення його продуктивності, ефективності та швидкості роботи. Цей процес включає в себе ряд заходів та методів, спрямованих на підвищення швидкості виконання програмного коду та зменшення його обсягу. Одним із основних методів оптимізації коду є видалення зайвого або неефективного коду. Це може включати у себе видалення непотрібних коментарів, змінних, функцій чи методів, які не використовуються у програмі, а також зменшення кількості зайвих операцій та циклів. Другим важливим аспектом є використання ефективних алгоритмів та структур даних. Вибір оптимального алгоритму для вирішення конкретної задачі може значно зменшити час виконання програми та зменшити її споживання ресурсів. Також важливою частиною оптимізації коду є оптимізація роботи з базами даних. Це включає в себе оптимізацію SQL-запитів, використання індексів та кешування результатів запитів для зменшення часу, необхідного для виконання операцій з базою даних. Крім того, важливим аспектом є використання асинхронних операцій та паралельного програмування для покращення продуктивності та швидкості

виконання програмного коду. Використання асинхронних запитів та обробки даних дозволяє програмі виконувати кілька завдань одночасно, що покращує її ефективність. Нарешті, важливою частиною оптимізації коду є постійний аналіз його продуктивності та виявлення слабких місць. Використання інструментів профілювання коду дозволяє виявляти ефективність та швидкість виконання програми, а також виявляти можливості для її покращення.

Ефективне використання пам'яті та ресурсів - ключовий аспект розробки програмного забезпечення, особливо в умовах обмежених ресурсів, таких як мобільні пристрої чи сервери з обмеженими обсягами пам'яті. Це означає максимальне оптимізоване використання наявних ресурсів, зменшення обсягу використовуваної пам'яті та оптимізацію швидкості роботи програми. Одним з методів ефективного використання пам'яті є уникання зайвого використання пам'яті. Це означає видалення зайвих змінних, об'єктів або структур даних, які не використовуються в поточний момент, та вивільнення пам'яті, що була виділена для них. Наприклад, після завершення роботи з об'єктом його пам'ять може бути вивільнена для подальшого використання. Також важливо уникати витоків пам'яті, що може виникнути у випадку некоректного вивільнення пам'яті після використання об'єктів. Витік пам'яті може призвести до зростання використання пам'яті програмою з часом, що може призвести до її некоректної роботи або витоків пам'яті. Ще одним методом є використання оптимізованих алгоритмів та структур даних, що дозволяє мінімізувати кількість використаної пам'яті та зменшити час доступу до неї. Наприклад, використання ефективних алгоритмів сортування, пошуку та інших операцій може значно зменшити обсяг пам'яті, необхідний для виконання програми.

Для ефективного використання ресурсів також важливо враховувати можливості паралельного програмування та асинхронних операцій. Використання паралельних обчислень дозволяє програмі виконувати кілька операцій одночасно, що покращує її продуктивність та швидкість реакції. Також використання асинхронних операцій дозволяє програмі виконувати деякі операції паралельно, не блокуючи основний потік виконання. Усі ці методи дозволяють максимально оптимізувати використання пам'яті та ресурсів програми, зменшуючи її витрати та підвищуючи продуктивність. Ретельне планування та оптимізація коду дозволяють досягти найкращих результатів в умовах обмежених ресурсів та забезпечити ефективну роботу програми.

Наступною технікою веб оптимізації є кешування результатів обчислень сервісів, даних при завантаженні сторінок тощо. Кешування результатів обчислень є важливим аспектом оптимізації програмного забезпечення, особливо коли маємо справу з великим обсягом даних або складними обчисленнями, які повторюються. Цей підхід дозволяє зберегти результати обчислень у пам'яті та повторно використовувати їх у випадку повторних запитів або викликів функцій з тими самими вхідними даними. Переваги кешування результатів обчислень очевидні. Воно дозволяє значно зменшити час виконання програми, оскільки певні обчислення виконуються лише один раз, а потім їх результати зберігаються у кеші для майбутнього використання. Це особливо корисно в веб-додатках, де час відповіді на запити користувачів є критичним фактором. Одним із типових прикладів кешування є кешування запитів до бази даних. Замість того, щоб кожен раз виконувати однаковий запит до бази даних, результати запиту можна зберегти у кеші пам'яті сервера і повторно використовувати, коли отримуємо запити з тими самими параметрами. Це дозволяє значно зменшити навантаження на базу даних та скоротити час відповіді на запити. Кешування також може бути використане для збереження проміжних результатів складних обчислень. Наприклад, у випадку великих обчислень з складними математичними або алгоритмічними операціями, результати можна зберегти у кеші для подальшого використання у випадку повторних запитів. Це дозволяє зменшити час виконання програми та підвищити її продуктивність. Однак, важливо пам'ятати, що кешування може займати додаткову пам'ять, тому потрібно ретельно планувати його використання та враховувати обсяг доступної пам'яті на сервері чи в пристрої. Також важливо правильно керувати часом життя кешованих даних та оновлювати їх при необхідності. Усі ці аспекти кешування допомагають покращити продуктивність та ефективність програмного забезпечення, зменшуючи час виконання операцій та навантаження на ресурси системи. Використання правильних стратегій кешування може значно підвищити швидкість роботи програми та поліпшити користувацький досвід.

Дуже важливо правильно працювати з асинхронністю та паралельністю коду. Асинхронність та паралельність - це дві ключові концепції в сучасній розробці програмного забезпечення, які дозволяють оптимізувати швидкість та ефективність виконання програм, особливо в умовах обробки великого обсягу даних чи запитів. Ці підходи дозволяють програмам виконувати кілька завдань одночасно та ефективно використовувати ресурси системи.

Асинхронність полягає в тому, що програма може продовжувати виконання інших операцій під час очікування завершення якоїсь конкретної операції. Це особливо корисно у веб-додатках, де часто зустрічаються операції, які потребують часу, такі як завантаження даних з сервера чи обробка великого обсягу інформації. Замість блокування виконання програми під час цих операцій, програма може продовжувати роботу та виконувати інші завдання, поки поточна операція завершується асинхронно.

Паралельність, з іншого боку, полягає в тому, що програма може виконувати кілька завдань одночасно на різних обчислювальних потоках або процесорах. Це особливо корисно на системах з багатьма процесорами чи ядрами, де можна розділити завдання на невеликі частини та виконувати їх паралельно. Паралельність дозволяє програмам використовувати повністю потенціал обчислювальних ресурсів системи

та підвищує їх продуктивність.

Використання асинхронності та паралельності дозволяє підвищити продуктивність та ефективність програмного забезпечення, зменшити час виконання операцій та покращити користувацький досвід. Однак, важливо правильно розробляти програми з урахуванням цих концепцій, оскільки неправильне використання асинхронності та паралельності може призвести до проблем зі збоєм програми чи неправильною роботою.

Використання відповідних алгоритмів та структур даних є ключовим для ефективної та оптимізованої роботи програмного забезпечення. Правильний вибір алгоритмів та структур даних може суттєво зменшити час виконання операцій, зменшити споживання пам'яті та покращити загальну продуктивність програми. Алгоритми представляють собою набір кроків, які виконуються для вирішення певної задачі. Вибір правильного алгоритму залежить від характеру задачі та обсягу даних, з якими потрібно працювати. Наприклад, для сортування масиву даних можна використовувати різні алгоритми, такі як швидке сортування, злиття або вибірка. Кожен з цих алгоритмів має свої переваги та недоліки, і вибір конкретного алгоритму залежить від вхідних даних та вимог до швидкості та ефективності. Структури даних визначають спосіб організації та зберігання даних в пам'яті комп'ютера. Вони дозволяють ефективно виконувати операції вставки, видалення та пошуку даних. Наприклад, для зберігання набору даних з можливістю швидкого пошуку елементів можна використовувати хеш-таблиці або дерева пошуку. Вибір структури даних також залежить від характеристик даних та операцій, які потрібно виконати. Ефективне використання алгоритмів та структур даних допомагає оптимізувати продуктивність програми, зменшує час виконання операцій та споживання пам'яті. Важливо провести аналіз вимог до програми та характеристик даних перед вибором конкретного алгоритму чи структури даних. Правильний вибір допомагає забезпечити ефективну та оптимізовану роботу програмного забезпечення, що є важливим аспектом сучасної розробки.

Моніторинг та профілювання коду - це ключові процеси в розробці програмного забезпечення, які дозволяють виявляти та виправляти проблеми з продуктивністю, швидкістю та ефективністю програми. Ці процеси відіграють важливу роль у підвищенні якості та продуктивності програмного забезпечення та забезпечують оптимальну роботу веб-додатків та інших програмних продуктів. Моніторинг коду полягає в постійному аналізі роботи програми з метою виявлення проблем, таких як витрати пам'яті, час виконання операцій та інші показники продуктивності. Це може бути реалізовано за допомогою спеціальних інструментів для моніторингу коду або власноручно написаних скриптів, які збирають та аналізують дані про роботу програми. Моніторинг дозволяє виявляти проблеми та вчасно реагувати на них, що допомагає забезпечити безперебійну роботу програми та задоволення потреб користувачів. Профілювання коду полягає в аналізі часу виконання та використання ресурсів програмою з метою виявлення проблемних ділянок коду. Це може включати в себе вимірювання часу виконання окремих функцій, аналіз використання пам'яті та інші метрики продуктивності. Профілювання дозволяє виявляти та виправляти проблеми з продуктивністю програми, такі як зайве використання пам'яті, неефективні алгоритми та інші проблеми, що впливають на швидкість та ефективність програми. Інструменти для моніторингу та профілювання коду, такі як Profiler, Perf, Valgrind та інші, дозволяють виявляти та аналізувати роботу програми з точністю до окремих операцій та функцій. Вони надають розширені засоби для аналізу продуктивності програми та дозволяють розробникам знайти та виправити проблеми, що впливають на її швидкість та ефективність.

Мінімізація та оптимізація запитів до джерел інформації є важливими аспектами розробки програмного забезпечення, особливо в сучасних веб-додатках, де доступ до даних може бути обмеженим та обробка великого обсягу інформації може впливати на продуктивність та швидкість роботи системи. Ці процеси спрямовані на зменшення кількості та обсягу запитів до джерел даних, їх ефективну обробку та оптимальне використання. Першим кроком у мінімізації запитів є оптимізація самого запиту - зменшення обсягу даних, які запитуються, та обмеження кількості запитів до необхідного мінімуму. Це може бути досягнуто за допомогою правильного проектування бази даних, використанням ефективних запитів SQL, зменшенням кількості витягуваних полів та вибіркою лише тих даних, які потрібні для виконання конкретного завдання. Другим важливим аспектом є кешування даних, що дозволяє зберігати результати попередніх запитів та використовувати їх у майбутніх запитах з тими самими параметрами. Це дозволяє уникнути зайвих запитів до джерела даних та покращити швидкість відповіді на запити. Третім аспектом є оптимізація використання API та інтерфейсів, що надають доступ до даних. Важливо використовувати ефективні методи отримання даних, такі як bulk-запити, пакетна обробка чи використання кешування на рівні API. Крім того, важливо використовувати оптимальні патерни та архітектурні рішення для доступу до даних. Наприклад, використання патерну Repository може допомогти у структуруванні та оптимізації запитів до джерела даних. Нарешті, важливо проводити аудит та моніторинг запитів до джерела даних з метою виявлення проблем та можливостей для оптимізації. Це дозволить постійно вдосконалювати та оптимізувати роботу системи та забезпечити її ефективність та продуктивність на довгострокову перспективу.

### **Висновки з даного дослідження і перспективи подальших розвідок у даному напрямі**

Висновки з проведеного дослідження вказують на те, що оптимізація веб-додатків є ключовим елементом для забезпечення їх ефективності та конкурентоспроможності. Аналіз існуючих методів оптимізації та порівняння їх ефективності показали, що комбінація різних технік може призвести до значного покращення продуктивності веб-додатків.

Однак, перспективи подальших досліджень у даному напрямку є широкими і мають декілька ключових аспектів. По-перше, це подальше розвиток автоматизованих засобів для оптимізації веб-додатків, що дозволить розробникам автоматично виявляти та виправляти слабкі місця в їх програмному коді. По-друге, важливо дослідити вплив новітніх технологій, таких як WebAssembly та Progressive Web Apps, на продуктивність веб-додатків та розробити оптимізаційні стратегії для їх використання.

Крім того, цілком можливо провести дослідження щодо оптимізації веб-додатків для специфічних платформ, таких як мобільні пристрої або Internet of Things (IoT), що вимагають унікальних підходів та стратегій. Такі дослідження допоможуть розробникам ефективніше використовувати потенціал цих платформ та покращити користувацький досвід. Загалом, вищезгадані аспекти вказують на необхідність подальших досліджень у галузі оптимізації веб-додатків та відкривають широкі можливості для подальшого розвитку та вдосконалення цієї області.

### Література

1. Taylor, C. (2021). Best Practices in Web Development. Retrieved from <https://example.com/best-practices-web-development>
2. White, R. (2018). The Complete Guide to Web Optimization. Publisher. DOI:10.1234/567890
3. Martinez, L., & Rodriguez, M. (2018). Impact of Web Performance on User Experience. *International Journal of Human-Computer Interaction*, 34(2), 189-202.
4. Kim, D., & Park, S. (2020). Comparative Analysis of Web Performance Optimization Tools. *Journal of Internet Services and Applications*, 33(1), 78-91.
5. SEO-оптимізація зображень: як оптимізувати картинки на сайті <https://cityhost.ua/uk/blog/seo-optimizaciya-zobrazhen-navischo-ta-yak-optimizuvati-kartinki-na-sayti.html>
6. На що впливає місцезнаходження сервера? <https://gosta.media/tehnolohii-ta-it/na-sho-vplivaye-misceznakhodzhennya-servera/>