**УДК 004.43**

*Лобода Р.М. студент*

*кафедри комп’ютерна інженерія*

*Полтавський національний технічний університет*

*імені Юрія Кондратюка*

**ОПТИМІЗАЦІЯ КОДУ МОВИ JAVASCRIPT: ПРОБЛЕМИ І ПЕРСПЕКТИВИ**

*В статті було розглянуто та виділено проблеми неоптимізованого коду, написаного мовою JavaScript. Проаналізовано способи і методи їх ефективного вирішення. Виявлено переваги і недоліки для оптимізації коду від використання framework’ів і додаткових бібліотек. Проаналізовані переваги використання збірників front-end’у, їх ефективність і продуктивність.*

***Ключові слова:*** *Front-end, Інтернет, web-розробка, оптимізація, JavaScript, JS, AngularJS, Gulp.*

**Вступ**

Розповсюдження мережі Інтерет в усіх сферах соціальної, наукової, економічної, виробничої та інших діяльностей призвело до збільшення об’єму розробки web-спрямованих інформаційних ресурсів. Хоча вона являється стандартизованою для більшості браузерів, оптимізація коду, або її відсутність, спричиняє посилене навантаження на середовище інтерпретації коду, негативно пливає на швидкість виконання та на поведінку одного варіанту коду в інших аналогах браузерів. Не можна оминати факт нанесення шкоди інтернет-ресурсам через малозахищеність та нелокалізованість написаного коду. З технологічної точки зору, область розробки функціональної частини web-інтерфейсу полягає в написанні сценарію виконання дій скриптовою мовою JavaScript. Найпростіші ресурси даного типу, зазвичай, мають в собі дві частини: веб-інтерфейс і сервер. Найпоширенішою мовою програмування у браузерному середовищі є JavaScript, яка є розширенням ECMAScript – вбудованої, стандартизованої мови, затвердженої міжнародною організацією [ECMA](https://uk.wikipedia.org/wiki/ECMA) згідно зі [специфікацією](https://uk.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F) ECMA-262. Використовується мова як у чистому вигляді, так і з використанням допоміжних бібліотек або framework’ів - програмного забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту. Framework надає початковий необхідний функціонал і позбавляє основних проблем неоптимізованого коду. На даний час вже закінчений і актуальний стандарт мови ECMAScript 6. Враховуючи недостатню підтримку останнього стандарту мови сучасними браузерами, критичної необхідності у розгляданні питання оптимізації не виникає. На відміну від стандарту ECMAScript 5. Особливості оптимізації коду мови JavaScript були розглянуті у таких науковій і інтернет-статтях як Медведєв Ю.С. [1], Джо Леннон [2], Антон Супрун [3], Ясюкевич Павло [5], Краковецький Олександр [6]. Основним колом людей, що займаються дослідженням оптимізації є front-end розробники, тому проводячи дослідницьку роботу я зіткнувся з проблемою недостатньої кількості наукового матеріалу по даному питанню, яку поверхнево компенсує наявність інтернет-статей і навчальних посібників. Звідси постає проблема асистематизованості і неструктуризованості наявних на даний час проблем оптимізації коду мови JavaScript. Розглянемо це питання детальніше.

В більшості опублікованих інтернет-статтях систематично повторюються наступні проблеми релізного коду:

1. Глобалізація змінних і функцій.
2. Утворення витоків пам’яті.
3. Велика кількість звертань до структури DOM і локального (сесійного) сховища.

До загальних проблем додаються переваги і недоліки використання framework’ів і сторонніх бібліотек, а також особливості використання складальників front-end.

Ці проблеми зустрічаються в кожному web-додатку і функціональній частині інформаційних web-ресурсів.

**Основна частина**

Згідно інтернет-статті автора Антона Супруна [3], першим правилом є використання замикання – внутрішньої функції зі своєю областю видимості, яка містить в собі всі необхідні змінні і функції. Перевагою даного методу є:

1. Захист елементу коду від зовнішнього доступу, при цьому всі зовнішні змінні і функції будуть доступні.
2. Розбиття коду на легко підтримувані модулі.
3. Захист від перезатирання вже існуючої функціональної частини.
4. Критичних недоліків у цього методу немає.

Цей метод відмінно підходить для вирішення проблеми глобалізації змінних і функцій. Перейдемо до наступної проблеми утворення витоків пам’яті.

Витоки пам'яті відбуваються , коли браузер по невідомим причинам не може звільнити пам'ять від недосяжних об'єктів. Зазвичай це відбувається автоматично і в більшості випадків причиною є браузер, але це може спричинити і архітектура JavaScript-коду. Для запобігання небезпеки створення витоку пам’яті потрібно завжди позбавлятись невикористовуваних змінних і очищати непотрібні ділянки пам’яті. Дотримуючись цього простого методу можливо зменшити ймовірність утворення витоку пам’яті на 40% і тим самим зробити додаток менш вимогливим до ресурсів оперативної пам’яті.

Щодо проблеми звертань. Звертання до DOM-дерева елементів – повільніший процес у порівнянні зі зчитуванням змінної, при одноразовому звертанні різниця не помітна, але зробивши двадцять запитів до DOM-архітектури браузер помітно отримає навантаження до моменту виконання останього запиту. Професійні JavaScript-розробники рекомендують у таких випадках робити один запит і результат поміщати у змінну. В подальшому звертання до елемента роботи саме через змінну, тим самим розвантажуючи браузер від повільних задач і прискорюючи роботу в цілому.

Ці прості і ефективні методи часто кладуться в основу функціоналу framework’ів, тому новачки, які починають вивчення мови JavaScript із framework’ів, мають проблеми з написанням коду на чистій мові. Саме тому автор Антон Супрун [3] в кінці статті рекомендує починати вивчення мови JavaScript саме з елементарних частин, без використання допоміжних додатків.

Великою перевагою більшості framework’ів є полегшення роботи в написанні проектів будь-якого маштабу і представлення мови у більш зручному синтаксисі досвідченому web-розробнику. Але для новачка framework може стати відправною точкою в написанні неякісного і неоптимізованого коду.

Згідно з GetInstance [7] Gulp - це інструмент збірки веб-додатків, що дозволяє автоматизувати повторювані завдання, такі як складання і мініфікація CSS- і JS-файлів, запуск тестів, перезавантаження браузера і т.д. Тим самим Gulp прискорює і оптимізує процес веб-розробки. Код web-інтерфейсу, написаний мовою JavaScript перед інтрпритацією мінімізується. І якщо після написання код мінімізувати, то це значно розвантажує інтерпритатор і робить додаток швидшим в роботі. Тому досвідчені front-end розробники завжди використовують інтсрументи збірки в роботі, на які прилягає робота мінімізації скриптового коду, конкатенація і мінімізація коду каскадних таблиць стилю і інші операції, які збірник може робити за долі секунди.

**Висновки**

Отже, результатом дослідження є систематизована і структуризована інформація про оптимізацію коду мови JavaScript. Було проаналізовано проблеми глобалізації змінних і функцій, утворення витоків пам’яті, звертань до структури DOM і локального (сесійного) сховища. Виділено ефективні методи вирішення проблем оптимізації. Розглянуто питання доцільності використання framework’ів, допоміжних бібліотек і збірників front-end-частини інтернет інтерфейсів.

*Література:*

1. *Медведєв Ю.С., Терехов В.В. К опросу о достижении максимальной производительности Ajax-приложений / Рецензируемый, реферируемый научный журнал «Вестник АГУ». Выпуск 4(125), 2013. Режим доступу:*

*http://cyberleninka.ru/article/n/k-voprosu-o-dostizhenii-maksimalnoy-proizvoditelnosti-ajax-prilozheniy*

1. *Джо Леннон. Оптимизация производительности JavaScript и Ajax [електронний ресурс] Режим доступу:* [*https://www.ibm.com/developerworks/ru/library/wa-aj-jsajaxperf/*](https://www.ibm.com/developerworks/ru/library/wa-aj-jsajaxperf/)
2. *Антон Супрун. 8 правил хорошего JavaScript-кода [електронний ресурс]Режим доступу:* [*http://popel-studio.com/blog/article/8-pravil-horoshego-javascript-koda.html*](http://popel-studio.com/blog/article/8-pravil-horoshego-javascript-koda.html)
3. *Оптимизация выполнения JavaScript [електронний ресурс] Режим доступу:*

[*https://developers.google.com/web/fundamentals/performance/rendering/optimize-javascript-execution?hl=ru*](https://developers.google.com/web/fundamentals/performance/rendering/optimize-javascript-execution?hl=ru)

1. *Ясюкевич Павло. Оптимизация: опыт, проверенный временем [електронний ресурс] Режим доступу: <https://habrahabr.ru/post/137318/>*
2. *Краковецький Олександр. Об идеальном коде и суровой реальности [електронний ресурс] Режим доступу:* [*https://habrahabr.ru/post/167877/*](https://habrahabr.ru/post/167877/)
3. *Gulp - інструмент збірки веб-додатків [електронний ресурс]. Режим доступу: http://getinstance.info/articles/tools/introduction-to-gulp/*