УДК 681.3.07

**І. Ю. Цимбал**

# Порівняння JavaScript трансляторів у браузерах «Chrome» та «Firefox»

Вінницький національний технічний університет

***Анотація***

*Проведено порівняльний аналіз сучасних архітектурних рішень у сфері трансляції JavaScript коду на прикладі рушіїв «V8» від «Google Chrome» та «SpiderMonkey» від «Mozilla Firefox». Досліджено специфіку оптимізації з використанням «Ignition» та «TurboFan» у браузері «Chrome», а також багаторівневу систему «WarpBuilder» у «Firefox». Обґрунтовано доцільність вибору конкретного браузера залежно від типу обчислювального навантаження та доступних системних ресурсів. Детально описано принципи роботи обох рушіїв та технології, що використовуються.*

***Ключові слова:*** *алгоритм, JavaScript, веб, Chrome, Firefox.*

***Abstract***

*A comparative analysis of modern architectural solutions in the field of JavaScript code translation is carried out using the «V8» engines from «Google Chrome» and «SpiderMonkey» from «Mozilla Firefox» as an example. The specifics of optimization using «Ignition» and «TurboFan» in the «Chrome» browser, as well as the multi-level «WarpBuilder» system in «Firefox», are studied. The feasibility of choosing a specific browser depending on the type of computational load and available system resources is substantiated. The principles of operation of both engines and the technologies used are described in detail.*

***Keywords:*** *algorithm, JavaScript, web, Chrome, Firefox.*

## Вступ

Актуальність обраної теми зумовлена стрімкою еволюцією веб-технологій, де швидкість обробки даних на стороні клієнта стала вирішальним фактором конкурентоспроможності програмних продуктів. Сучасний етап розвитку глобальної мережі характеризується переходом від простих статичних сторінок до складних високонавантажених веб-застосунків, що за своєю функціональністю не поступаються десктопним аналогам. Одне з основних місць в цій екосистемі посідає мова програмування «JavaScript».

Основними гравцями на цьому ринку є компанії «Google» та «Mozilla», що є власниками та розробниками браузерів «Chrome» та «Firefox» відповідно, що використовують принципово різні архітектурні підходи до обробки «JavaScript». Рушії «V8» від «Chrome» та «SpiderMonkey» від «Firefox» є еталонами рушіїв. Вони використовують технологію JIT-компіляції (Just-In-Time), яка дозволяє перетворювати високорівневий код у машинний безпосередньо під час його виконання, адаптуючись до типів даних у реальному часі.

Порівняльний аналіз цих трансляторів дозволить не лише оцінити поточний стан продуктивності браузерів, а й зрозуміти вектор розвитку веб-технологій загалом. Розуміння механізмів оптимізації, таких як інлайнінг функцій та керування пам'яттю, є критично важливим для сучасного IT-фахівця при проектуванні ефективних та масштабованих систем.

Метою даної роботи є проведення порівняльного дослідження архітектурних особливостей та алгоритмів оптимізації JavaScript-трансляторів у браузерах «Google Chrome» та «Mozilla Firefox», а також оцінка їхнього впливу на швидкість рендерингу складного контенту та використання системних ресурсів.

Об’єктом дослідження є процеси трансляції та виконання програмного коду на мові JavaScript у сучасних браузерних середовищах.

Предметом дослідження є архітектурні особливості, алгоритми JIT-компіляції та методи оптимізації рушіїв «V8» та «SpiderMonkey».

**Механізми роботи сучасних JS-рушіїв**

Сучасна трансляція JavaScript – це не просто лінійне читання коду. Це багаторівневий процес, що включає:

1. Перетворення тексту коду в абстрактне синтаксичне дерево (AST).
2. Швидкий запуск коду за допомогою байт-код інтерпретатора.
3. Виявлення частин коду, що виконуються часто та їх перетворення в оптимізований машинний код.

**Аналіз архітектури рушія «V8» від «Google Chrome»**

Рушій «V8» відіграє ключову роль у сучасній веб-розробці, оскільки він став основою не лише для браузера «Chrome», а й для середовища виконання «Node.js». Головною особливістю «V8» є відмова від проміжного байт-коду на ранніх етапах розвитку, проте сучасна версія використовує систему «Ignition».

Ignition [1] – це швидкий низькорівневий регістровий інтерпретатор, написаний з використанням серверної частини «TurboFan».

Основна причина застосування інтерпретатора Ignition – економія пам'яті. Це пояснюється тим, що інтерпретатор компілюватиме лише необхідні рядки: на відміну від компілятора, який компілює всю програму.

Однак інтерпретатор «Ignition» відповідає лише за перший запуск вашого коду. Потім згенерований байт-код буде використовуватися компілятором під назвою «TurboFan». Він оптимізовує код на основі даних, які отримуються під час виконання коду, і перекомпілює його в оптимізовану версію.

Також «V8» використовує технологію «Inline caching», що дозволяє уникати повторних пошуків властивостей об'єктів, запам'ятовуючи результати попередніх операцій. А для того, аби пришвидшити доступ до об’єктів та їх полів, використовується підхід прихованих класів.

На рисунку 1 продемонстровано принцип роботи рушія «V8».

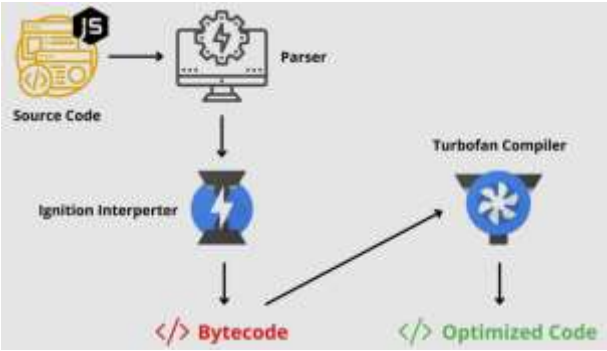


Рисунок 1. Принцип роботи «V8»

Таким чином на вхід подається код, який проходить через процес парсингу, а далі, використовуючи «Ignition» розкладається на байти, що дозволяє програмі запуститися миттєво. Під час виконання безпосередньо коду програми «TurboFan» спостерігає за виконанням функцій, і ті, що виконуються найчастіше – перетворює на машинний код, що є швидшим, ніж байтів, бо комп’ютер розуміє його одразу.

**Аналіз архітектури рушія «SpiderMonkey» від «Firefox»**

«SpiderMonkey» є першим JS-рушієм. Наразі він демонструє унікальний підхід до багатопоточності та керування пам'яттю. Його архітектура включає декілька рівнів JIT-компіляції.

Рушій «SpiderMonkey» складається із наступних рівнів [2]:

1. На першому рівні генерується фіксований байт-код, який працює швидше за звичайну інтерпретацію, але ще не повністю оптимізований.
2. Коли рушій бачить, що функція викликається кілька разів, то вона передається на другий рівень, тобто в «Baseline JIT». На цьому рівні код перетворюється у машинний, але без надлишкових оптимізацій. Головна мета – це зібрати всю інформацію про типи даних і змінні, що використовуються, аби в подальшому мати швидшкий доступ до них.

3. На цьому рівні застосовуються агресивні методи оптимізації. Тут працює «WarpBuilder», що аналізує результати виконання різних ділянок коду, і на основі зібраної аналітики прибирає зайві перевірки.

Особлива увага в «SpiderMonkey» приділяється безпеці та стабільності споживання ресурсів. На відміну від «V8», який може споживати велику кількість оперативної пам'яті для кешування оптимізованого коду, «SpiderMonkey» використовує більш консервативні стратегії, що робить його ефективним у системах з обмеженими ресурсами.

Використання "inline caching" у Firefox реалізовано таким чином, щоб забезпечити баланс між швидкістю доступу до властивостей об'єктів та обсягом метаданих у пам'яті.

**Порівняння «SpiderMonkey» та «V8»**

У таблиці 1 наведено критерії оцінювання рушіїв.

Таблиця 1 – Порівняльний аналіз «SpiderMonkey» та «V8»

| | V8 | SpiderMonkey |
|---|---|---|
| Використання JIT-компіляції | Так | Так |
| Вмкористання технології швидкого доступу (приховані класи) | Так | Ні |
| Перетворення в машинний код функцій, що часто викликаються | Так | Так |
| Оптимізоване використання оперативної пам’яті | Ні | Так |
| Аналіз коду і оптимізація на основі припущення (аналітики) | Так | Так |
| Оптимізована робота з об’єктами | Так | Ні |
| Сумарний коефіцієнт | 5 | 4 |

Проаналізувавши таблицю 1 можна зробити висновок, що обидва рушії надзвичайно схожі. Але є незначні відмінності, що можуть стати в нагоді в різних випадках та проектах.

**Висновок**

Отже, можна зазначити, що перевага «V8» у роботі з об'єктами робить його ідеальним вибором для застосунків, які оперують великими масивами даних у реальному часі. Водночас «SpiderMonkey» залишається сильнішим у питаннях стабільності виконання довготривалих скриптів, де критично важливим є постійний рівень продуктивності без раптових стрибків споживання оперативної пам’яті.

Вибір цільового браузера для розробника часто залежить від цільової аудиторії: для мобільних пристроїв та систем з обмеженою пам'яттю «Firefox» є кращим варіантом, тоді як для комп’ютерів «Chrome» забезпечує швидшу обробку даних при масштабних проектах або іграх.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ignition [Електронний ресурс] – Режим доступу до ресурсу: https://v8.dev/docs/ignition
2. SpiderMonkey [Електронний ресурс] – Режим доступу до ресурсу: https://spidermonkey.dev/assets/pdf/SpiderMonkey%20Byte-sized%20Architectures.pdf

***Цимбал Іван Юрійович*** – студент групи 1ПІ-25м, факультет інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: tsymbal.ivan2004@gmail.com

***Науковий керівник****: Васильківський Микола Володимирович* - кандидат технічних наук, доцент, доцент кафедри інфокомуні-каційних систем і технологій, Вінницький національний технічний університет, м. Вінниця, e-mail: mvasylkivskyi@gmail.com

***Ivan Tsymbal*** – student of group 1PI-25m, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Ukraine.

***Supervisor****: Vasylkivskyi Mykola V* - candidate of technical sciences, associate professor, associate professor of the Department of Information Communication Systems and Technologies, Vinnytsia National Technical University, Vinnytsia,e-mail: mvasylkivskyi@gmail.com