УДК 004.63

Д. Рекечинський, А. Волокита

СПОСІБ ВІДСТЕЖЕННЯ ЗМІН У ІМПОРТОВАНИХ ФАЙЛАХ НА ПРИКЛАДІ МОВИ NFML

Анотація. Сучасний підхід до програмування заохочує розбивати сирцевий код проекту на складні ієрархії з файлів, які відповідають їхньому призначенню. З одного боку, це полегшує подальший процес розробки, оскільки розширення функцій програми або її видозміна стає в рази простішою. З іншого боку, якщо необхідно розробити алгоритм, який би відстежував усі файли системи для того, щоб реагувати на зміни (наприклад, для гарячої компіляції), це стає доволі нетривіальною задачею. Ця робота несе мету вирішити проблему відстеження файлів у проекті, розглянувши на прикладі мови розмітки NFML. Розроблений алгоритм було реалізовано на мові програмування JavaScript для середовища Node.js із застосуванням бібліотеки chokidar.

Ключові слова: алгоритм, ієрархія, файлова система, відстеження змін, Node.js.

Вступ

Програмування сьогодення має чимало інструментів, які значно спрощують розробку: система контролю версій (наприклад, Git), інтегровані середовища розробки, налагоджувач програми тощо. Одним із таких інструментів є гаряча компіляція, яка дозволяє «на льоту» компілювати всю програму, якщо було відстежено зміни бодай у одному файлі,

Втім, це буває складно реалізувати для проектів із довільною структурою папок. Крім цього, можуть виникати проблеми, якщо один із файлів був щойно імпортований під час активації гарячої компіляції. Для вирішення цієї задачі потрібні нові підходи, що дозволяють швидко та надійно реагувати на зміни. Метою дослідження є розробка алгоритму, який забезпечить ефективне відстеження змін у файлах за будь-якої ієрархії файлів.

Матеріали та підходи

Припустимо, що необхідно працювати з проектом, який застосовує мову NFML, компільовану за допомогою програми, написаній на JavaScript. Для компіляції вказується головний файл, який може імпортувати інші за допомогою директиви import 'шлях-до-файлу'. Необхідно знайти такий метод гарячої компіляції, який дасть змогу працювати з цим проектом незалежно від розміщення усіх файлів цього проекту,

 ISSN 1560-8956

а також матиме коректну поведінку при додаванні або видаленні імпортованих файлів:

- Якщо було додано новий файл в якості імпортованого у файл, який зараз відстежується, новий файл має відстежуватись також.
- Якщо імпорт для існуючого файлу було видалено у файлі, який його використовував і більше такий файл ніде не вказується, зміни існуючого файлу більше не відстежуються.

Для середовища Node.js існує декілька способів відстеження стану файлів, і серед них є як і стандартне рішення, так і зовнішні:

- Запуск node з параметром --watch або --watch-path, стандартне рішення
- nodemon, зовнішня утиліта
- chokidar, зовнішня бібліотека

Починаючи з версії 18.11.0, Node.js почав підтримувати режим відстеження змін файлів [1]. Це робиться за допомогою запуску програми node із параметром --watch. Втім, цей параметр призначено безпосередньо для скрипту, який виконується, що несе за собою обмеження: це не можна використати для файлів, які не написані мовою JavaScript.

Крім параметру --watch, в Node.js було також додано параметр --watch-path, який дозволяє додавати відстежування за всією директорією, і зміни у ній відстежуються незалежно від вмісту. Також, можна додавати декілька директорій для відстежування за допомогою додавання декількох пар --watch-path та шляху до директорії. Однак, є ще одне обмеження, яке стосується що --watch, що --watch-path: при виявленні змін у файлах процес Node.js просто перезапускається, що обмежує можливості програми щодо реагування на ці зміни.

Утиліта nodemon має функціональність, подібну до тієї, що надають стандартні параметри Node.js, однак ця утиліта надає більшу гнучкість для конфігурації необхідної поведінки. Наприклад, можна задавати розширення файлів, які будуть відстежуватись у певній директорії, або шляхи, які будуть ігноруватись [2]. Крім цього, можна використовувати nodemon в якості функції у коді. Втім, як і node --watch, nodemon перезапускає програму повністю в якості реакції на зміни.

Бібліотека chokidar надає для взаємодії об’єкт chokidar, який може бути використаний всередині одного зі скриптів [3]. Його суттєвою перевагою є те, що він дозволяє встановити стеження за окремими файлами незалежно від їхнього типу. Методи цього об’єкту та їх детальний опис надані всередині таблиці 1.

Бібліотека chokidar надає всі функції, які необхідні для побудови ефективного алгоритму гарячої компіляції в середовищі Node.js. Наступними кроками є розробка алгоритму та перевірка алгоритму на відповідність вимогам.

Таблиця 1.

Методи об’єкту chokidar

Назва	Аргументи	Призначення
watch	paths: String або String[]. Шлях до файлу, або масив шляхів до файлу. options: Object. Об’єкт з конфігураціями.	Цей метод відповідає за початкову ініціалізацію chokidar та безпосередній запуск процесу відстеження за змінами файлів.
on	event: String. Назва події, на яку треба реагувати. callback: Function. Функція, яка викликається в якості реакції на зміну.	Цей метод дозволяє додавати поведінку залежно від того, що відбувається з файлами. Наприклад, для події change можна вказати, що виконувати при зміні змісту файлів.
add	paths: String або String[]. Шлях до файлу, або масив шляхів до файлу.	Додає файли для відстеження до уже ініціалізованого об’єкту chokidar.
unwatch	paths: String або String[]. Шлях до файлу, або масив шляхів до файлу.	Видаляє файли з відстеження.
getWatched	Немає	Метод повертає об’єкт ключ-значення. В якості ключа виступає повний шлях до папки, в якості значення – масив назв файлів, які відстежуються.
close	Немає	Припинити діяльність chokidar.

Розробка алгоритму

В першу чергу необхідно зрозуміти порядок дій гарячої компіляції. Спочатку слід прочитати головний файл компільованого проекту, щоб зрозуміти, які модулі він імпортує. Відповідно, слід також прочитати ті модулі, щоб побачити, що вони імпортують і так далі. Для цього треба виконати парсинг всієї системи, що також може охоплювати першу ітерацію компіляції.

Після цього список всіх файлів можна зібрати, щоб запустити фазу відстежування. Як тільки користувач виконає зміну одного із файлів, функція повинна зреагувати на це. Реакцією повинна виступати логіка, яка запустить наступну ітерацію компіляції та складе новий масив усіх файлів проекту.

Як видно з таблиці 1, chokidar не має логіки для заміни шляхів. Натомість, можна отримати список файлів, які відстежуються станом на зараз за допомогою методу getWatched та відфільтрувати ті файли, які більше не застосовуються в проекті.

 ISSN 1560-8956

Їх можна вилучити з відстеження за допомогою методу unwatch. Крім цього, слід додати нові файли до відстежуваних методом add. Рисунок 1 відображає схему розробленого алгоритму.

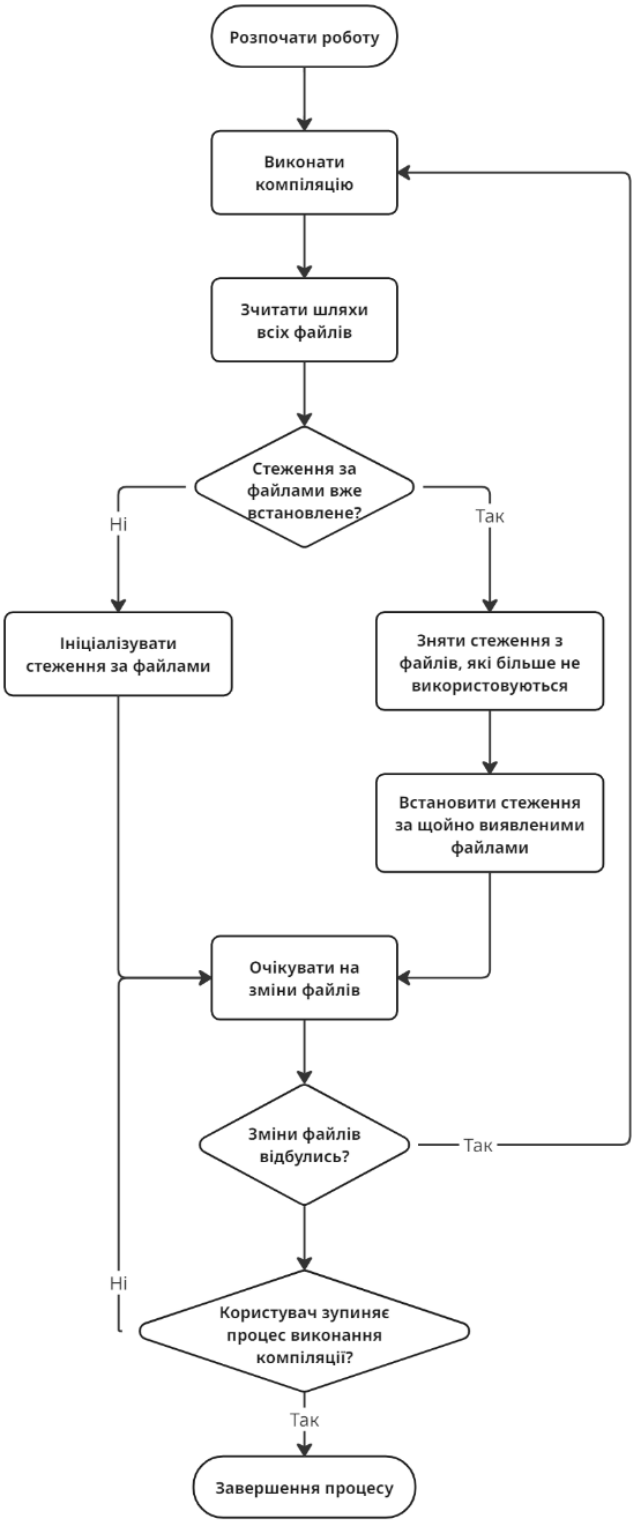

Рисунок 1. Схема розробленого алгоритму

Результати та обговорення

Для перевірки алгоритму було створено тестову структуру файлів:

upperscope.nfml
project/
    main.nfml
    submodules/
        a.nfml
        b.nfml

Файл main.nfml імпортує файли upperscope.nfml та a.nfml. Файл a.nfml імпортує b.nfml. Це дає змогу тестувати 2 випадки: реагування на файл поза межею головного файлу (upperscope.nfml відносно main.nfml) та багато вкладень (b.nfml відносно main.nfml).

Перший тест був проведений, щоб перевірити реакцію алгоритму гарячої компіляції на збереження будь-якого із зареєстрованих файлів проекту. Консоль виводу результатів програми показала, що алгоритм успішно спрацьовує для усіх файлів. Приклад виводу зображено на рисунку 2.

На консоль виводиться файл, який було змінено (та, відповідно, що спричинило реагування) та список абсолютних шляхів до наразі відстежуваних файлів.

```
File changed: C:\Users\Dmytro\Documents\master_stuff\nfml\examples\submodule\b.nf
ml
[
  'C:\\Users\\Dmytro\\Documents\\master_stuff\\nfml\\examples\\submodule\\a.nfml'
,
  'C:\\Users\\Dmytro\\Documents\\master_stuff\\nfml\\upperscope.nfml',
  'C:\\Users\\Dmytro\\Documents\\master_stuff\\nfml\\examples\\submodule\\b.nfml'
,
  'C:\\Users\\Dmytro\\Documents\\master_stuff\\nfml\\examples\\main.nfml'
]
```

Рисунок 2. Вивід результатів першого тесту

В якості другого тесту виконана перевірка реакції алгоритму на те, що один із файлів вилучений із імпортованих, а потім його видозміна щоб зрозуміти, чи реагує система на файли, які більше не є частиною проекту. На рисунку 3 зображено результат вилучення файлу a.nfml із імпортованих файлів.

Як результат, алгоритм більше не реагував на зміни файлів a.nfml та b.nfml, оскільки вони більше не є частиною проекту. Якщо вилучити використання b.nfml, а натомість імпортувати тільки a.nfml, то алгоритм реагує на зміни в a,nfml, але не в b.nfml, як і очікувалось.

Таким чином, всі тести були пройдені успішно, а вимоги до розробленого алгоритму були виконані.

 ISSN 1560-8956

```
File changed: C:\Users\Dmytro\Documents\master_stuff\nfml\examples\main.nfml
[
  'C:\\Users\\Dmytro\\Documents\\master_stuff\\nfml\\upperscope.nfml',
  'C:\\Users\\Dmytro\\Documents\\master_stuff\\nfml\\examples\\main.nfml'
]
```

Рисунок 3. Вивід результат другого тесту

Висновки

В ході досліджень було виявлено декілька способів для відстеження змін файлів в середовищі Node.js: стандартне рішення node --watch, nodemon та chokidar. Що node --watch, що nodemon мали суттєвий недолік: при виявленні змін вони повністю перезапускають програму. Крім цього, вони надають змогу відстежувати або поточний скрипт запуску (та його залежні модулі), або окремі папки. Натомість, chokidar дає змогу встановити стеження за кожним файлом окремо, а також в якості реакції га зміни дає змогу викликати функцію, не перезапускаючи процес повністю.

Розроблено алгоритм, який дає змогу відстежувати поточні файли проекту, які є його безпосередньою частиною, а також враховує, коли файли додаються чи видаляються з імпортованих у проекті.

Отримані результати можуть бути використані для розробки гарячої компіляції чи іншої системи, для якої важливо реагувати на зміни файлів незалежно від мови програмування та ієрархії файлів.

Список використаних джерел

1. Command-line API | Node.js v23.10.0 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/api/cli.html (дата звернення: 26.03.2025).

2. nodemon | GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/remy/nodemon (дата звернення: 26.03.2025).

3. chokidar | GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/paulmillr/chokidar (дата звернення: 27.03.2025).