УДК 004.4
В. Сербін
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РЕАЛІЗАЦІЯ ПАРАЛЕЛЬНОЇ ОБРОБКИ ДАНИХ В МОВІ ПРОГРАМУВАННЯ JAVASCRIPT

UDC 004.4
V. Serbin

IMPLEMENTATION OF PARALLEL DATA PROCESSING IN JAVASCRIPT PROGRAMMING LANGUAGE

Станом на сьогодні, мова програмування JavaScript більше не обмежується браузером, а використовується для створення програмного забезпечення у будь-якому пристрої, від серверів до пристроїв Інтернету речей. Багато з написаних програм є важкими і можуть отримати значне підвищення продуктивності від використання паралельних обчислень. В загальному, причина розпаралелювати JS така ж, як і причина розпаралелювати будь-яку іншу мову: закон Мура вмирає, а багатоядерна архітектура захоплює світ [1].

Використання API Web Workers, ймовірно, є єдиним способом досягти справжньої мультипроцесорності в Javascript [2]. Web Worker дозволяють користувачеві паралельно запускати JavaScript без втручання в інтерфейс користувача. Робочий сценарій буде завантажено та запущено у фоновому режимі, повністю незалежно від сценаріїв інтерфейсу користувача. Це означає, що Workers не мають доступу до елементів інтерфейсу користувача, таких як DOM і поширених функцій JS, як-от getElementById (але можуть здійснювати виклики AJAX). Основний варіант використання API Web Worker полягає у виконанні обчислювально дорогих завдань у фоновому режимі, без процесу переривання або переривання взаємодії користувача [1].

Щоб реалізувати Worker, необхідно створити його екземпляр за допомогою коду, який він запускатиме [2]:

```
const worker = new Worker(«worker.js»);
```

Web Workers спілкуються з основним документом/основним потоком через техніку передачі повідомлень. Передача повідомлень здійснюється за допомогою API postMessage [1]:

```
worker.postMessage(num);
```

Згідно зі специфікацією існує два типи Web Worker: спільні та виділені.

Web Worker за замовчуванням називається виділеним. Цей тип робочого файлу доступний лише зі сценарію, який його створив, тоді як спільний робочий файл можна отримати з кількох сценаріїв.

Принцип взаємодії Web Workers та основного потоку наведено на рисунку 1.

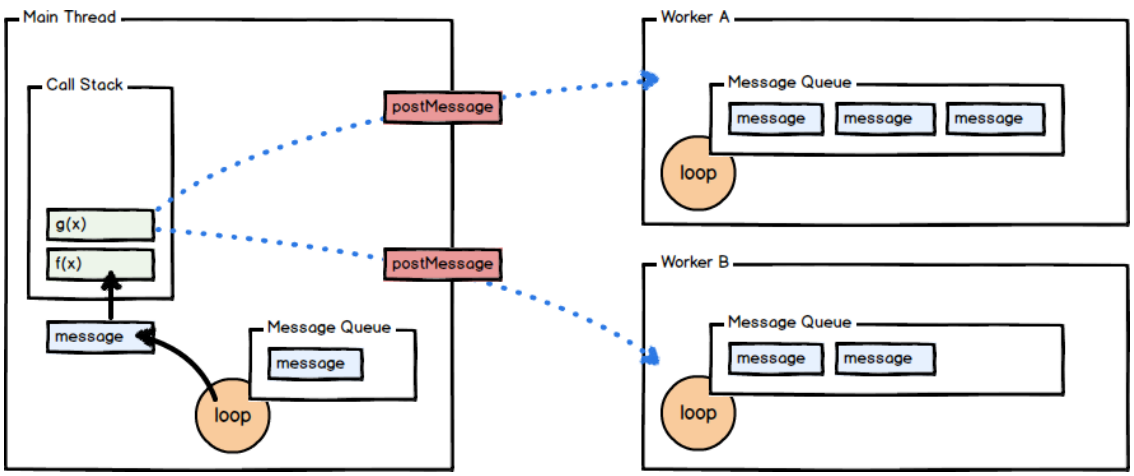

Рисунок 1. Схема взаємодії Web Workers та основного потоку

Для спільного Web Worker потрібен інший конструктор: SharedWorker. Спільний робочий файл доступний кількома сценаріями, навіть якщо до них звертаються різні вікна, iframe або навіть робочі.

Прослуховування події відбувається через обробник «onmessage()», в якому e.data міститиме передане значення

Цей процес працює в обох напрямках, тож можна повернути деякі дані назад із робочого коду у основний потік [2]. Для цього використаємо у коді worker-у «postMessage()» з результатом, а у основному потоці використовуємо «worker.onmessage» для прийому даних.

Отже, потоки Web Worker допомагають нам розвантажити інтенсивні завдання ЦП з циклу подій, щоб виконувати їх паралельно без блокування. Робочий потік виконує частину коду відповідно до вказівок батьківського потоку окремо від батьківського та інших робочих потоків. Кожен робочий потік має власне ізольоване середовище, цикл подій, чергу подій тощо. Робочий і батьківський потік можуть спілкуватися один з одним через канал обміну повідомленнями, а також робочі потоки дають нам можливість запускати кілька потоків в одному процесі [3]. Крім утримання циклу подій від виконання трудомістких операцій ЦП можемо використовувати пул робочих потоків для розділення та паралельного виконання важких операцій ЦП з ціллю підвищення продуктивності нашої програми [3].

Література

1. Розпаралелювання JavaScript для розваги та прибутку. URL: https://www.codementor.io/@madhugnadig/parallelizing-javascript-for-fun-and-profit-naxmo4lam.
2. Паралельна обробка в JS. URL: https://advancedweb.hu/parallel-processing-in-js/.
3. Паралельна обробка в Node.js з використанням робочих потоків. URL: https://deepsource.io/blog/nodejs-worker-threads/.