Web Workers

Muhammad Umar
2021-CS-24

1. Brief Description of the Project

This project is a React application that fetches population data from the Data USA API and provides two sorting mechanisms for the data: one utilizing Web Workers for parallel processing and another without Web Workers. The goal is to compare the performance of sorting large datasets with and without Web Workers in a React application.

2. Instructions to Run the Project Locally

1. Clone the repository:
   `git clone https://github.com/MuhammadUmarAleem/Web-Worker.git`
2. Navigate to the project directory:
   `cd web-worker`
3. Open Project in a web browser:
   Run `npm start` in the project directory in the terminal.
4. Open the browser's developer tools and check the console for logs. Check logs for calculations with web workers and not web workers. The application will be accessible at `http://localhost:3000` in your web browser.

3. Summary of Findings on Performance Improvements with Web Workers

The project demonstrates the use of Web Workers to perform sorting operations in parallel, which can potentially enhance performance when dealing with large datasets. By offloading the sorting task to a separate thread, Web Workers allow the main thread to remain responsive, providing a smoother user experience.

4. Challenges Faced and Overcoming Them

- Web Worker Communication: Communicating between the main thread and the Web Worker required careful handling. The `postMessage` and `onmessage` events were used to exchange data efficiently.

- Data Fetching: Fetching data from the Data USA API introduced asynchronous operations. The `async/await` pattern was employed to handle these operations and ensure data availability before sorting.

5. References/Resources Used

- Data USA API: `https://tasty-windy-eyebrow.glitch.me/GetPlans`
- MDN Web Docs - Web Workers: `https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API`
- React Documentation: `https://reactjs.org/docs/getting-started.html`

6. Conclusion

Utilizing Web Workers for computationally intensive tasks, such as sorting large datasets, can lead to improved performance in web applications. The parallel processing capabilities of Web Workers allow for efficient multitasking without affecting the responsiveness of the main thread. However, it's essential to consider the specific use case and complexity of the task to determine the most suitable approach. By running the project locally and comparing the sorting times with and without Web Workers, users can observe firsthand the potential performance benefits and make informed decisions based on their application requirements.