Information Technology Institute

HTML5 & CSS3

Lab-2 Question-4

Ahmed Ebrahim Morsy Ebrahim

compare between web worker and shared worker.

1.1. Web Worker

- **Definition**
    - This worker is linked to one specific script file
    - It is dedicated to that single page.
    - If you open the same website in three different tabs, you get three separate workers running three separate tasks.
    - They don't know about each other.
- **Lifecycle**
    - terminate immediately when we close website.
- **When to choose**
    - Isolation
        - Each Tab has its own worker thread (Don’t mess with another).
    - Background Tasks
        - Fetching data from API (only for this specific page).
    - Heavy Calculations
        - like in lecture summation example, big math (Fibonacci).

How to use

Script file

```
let count = document.getElementById("count");
let add = document.getElementById("add");
let sub = document.getElementById("sub");

let woker = new Worker("worker.js");



woker.onmessage = (message) => {
    count.innerText = "";
    count.innerText = message.data;

}


add.onclick = () => {

    woker.postMessage("add");

}

sub.onclick = () => {
    woker.postMessage("sub");
}
```

Worker file

```
let count = 0;


this.onmessage = (e) => {


    if (e.data === "add") {
        count++;
    }
    else if (e.data === "sub") {
        count--;
    }

    self.postMessage(count);


}
```

1.2. Shared Worker

- **Definition**
    - This worker links to multiple scripts and could be accessible by multiple tabs, iframes, or windows if they come from the same origin.
    - If you open your website in 5 tabs, they all connect to one single Shared Worker.
- **Lifecycle**
    - Start when at least one page connected to it is open
    - Terminate when the last page closes.
- **When to choose**
    - Synchronization Tabs
        - when we have multiple tabs for same page and we want to update all tabs without reloading each tab to be updated to any change in data (Data Sharing between tabs).
    - Resource Optimization
        - The Shared Worker opens one connection and post data to all tabs instead of opening new port for each tab.

How to use

Script file

```
let count = document.getElementById("count");
let add = document.getElementById("add");
let sub = document.getElementById("sub");

let sharedWorker = new SharedWorker("sharedWorker.js");

sharedWorker.port.start();

sharedWorker.port.onmessage = (message) => {
    count.innerText = "";
    count.innerText = message.data;

}


add.onclick = () => {

    sharedWorker.port.postMessage("add");

}

sub.onclick = () => {
    sharedWorker.port.postMessage("sub");
}


sharedWorker.port.postMessage("");
```

Shared Worker file

```
let count = 0;
let ports = [];
let port;
this.onconnect = (e) => {

    port = e.ports[0];


    ports.push(port);
    port.start();

    port.onmessage = (e) => {
        if (e.data === "add") {
            count++;
        }
        else if (e.data === "sub") {
            count--;
        }


        for (let i = 0; i < ports.length; i++) {
            ports[i].postMessage(count);
        }


    };


}
```

Summary

	Web Worker	Shared Worker
Definition	A background task linked to a single page.	A background task linked to multiple tabs from the same origin.
Scope	Only one tab that created it,	Shared. Any tab from the same origin,
Lifecycle	terminate immediately when we close website.	Start when at least one tab connected to it is open and terminate when the last tab closes.
How it works	only uses postMessage() to start worker.	We need to use port object to manage connections port.postMessage().
Resource Usage	Each tab opens we need to create worker threads which consume resource too much.	All tab opens only need one worker threads which optimize resource usage.
Data Sharing	Isolated. Each Tab has its own worker thread.	Synchronized. Each tab could be sharing data between other tabs.
When we use	Heavy calculations that only affect one tab.	syncing data across tabs (like counter example).