Some notes on SharedWorkers

Chris How · Follow
3 min read · Nov 3, 2024

2

I recently needed to implement a shared worker in a project. Though they are super useful, there wasn't much info to be found in the usual places, so here are some pointers that might help searchers from the mysterious future.

Background

SharedWorkers are a special class of WebWorker that can be shared across multiple tabs, windows or other (regular) web workers.

In my application, I needed a process which would poll for new application events (for example, a customer completing a purchase), and show a notification (using the Notifications API) to logged in administrators (or more specifically, those logged in administrators who had chosen to receive notifications).

An administrator could have the application open in several tabs or windows, so it would be wasteful to have each tab polling for new events. I only wanted one notification per event, regardless of the number of open tabs or windows.

SharedWorker to the rescue! Each of the open tabs or windows shares a single worker, which polls in the background, and shows just one notification per new event.

Creating a shared worker with Vite

The first challenge was loading the shared worker in my Vite-based setup.

If you're running Vite in dev mode, Vite serves the script from a different domain and port (eg `http://[::1]:5173/`), which won't work, because shared workers must obey the same-origin policy.

I tried various Vite workarounds for web workers:

- The official Vite web worker method doesn't work for *shared* workers due to the same-origin policy requirement.
- Blob URLs aren't supported for shared workers.
- Inlining the worker as a base64 string doesn't work because the browser treats them as different workers: fine for web workers, but not for shared workers.

In the end, I created a new route to serve the script either from the resources directory in dev, or the build directory in staging and live environments.

```
Route::addRoute('GET', '/notifications-shared-worker.js', function () {
 // If in dev environment, send the file from the resources folder
 if (app()->environment('local')) {
 return response()->file(resource_path('js/notificationWatcherWorker.js'), ['Con
 } else {
 // Otherwise, send the file from the public folder
 return response()->file(public_path('build/assets/notificationWatcherWorker.js'
 }
});
```

I then create the shared worker with that route as the URL:

```
const worker = new SharedWorker('/notifications-shared-worker.js');
```

Debugging the Shared Worker

You'll quickly find that any syntax or runtime errors in your shared worker don't appear in your devtools. Nor do any console log/warn/info calls.

This one is easy, paste 'chrome://inspect/#workers' into your URL bar, find the shared worker and click on 'inspect'. Now you have a devtools window just for the shared worker.

Communicating back to the main tab or window

To communicate back to the 'parent' tab, use the `port.postMessage` method, as described in the MDN SharedWorker documentation.

However, the example code only allows communication with the most recent 'parent' tab/window because it overwrites the communication port reference each time a parent connects.

Instead, store an array of ports, and add each new port to the array when a new 'parent' connects.

```
const ports = [];
onconnect = (e) => {
 // Get the port for this new connection
 const port = e.ports[0];
 // Add it to our array of ports
 ports.push(port);
…
}
```

Then, send a message to all the parent pages like this:

```
ports.forEach(port => port.postMessage({
    message: "Hello, world!",
    flavor: "vanilla"
}));
```

Written by Chris How

2 Followers · 2 Following

Follow

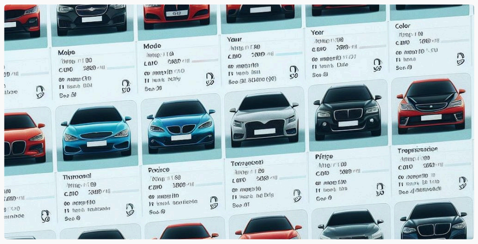
Full stack developer, Laravel, WordPress, PHP, Javascript, CSS. Based in Canary Islands, Spain.

No responses yet

More from Chris How

Chris How

“Scroll to scrub” videos



Chris How

SQLite, case sensitivity, and Laravel testing

A cool way to show off video content that puts the user in control 👇

Nov 16, 2023 · 2

Imagine you have a bunch of records like this:

Dec 21, 2023 · 2

See all from Chris How

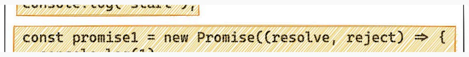
Recommended from Medium

Harendra

How I Am Using a Lifetime 100% Free Server

Get a server with 24 GB RAM + 4 CPU + 200 GB Storage + Always Free

Oct 26 · 8.1K · 125



```
const promise1 = new Promise((resolve, reject) => {
```

In Programming Domain by Shuai Li

Can You Answer This Senior Level JavaScript Promise Interview...

Most interviewees failed on it.

Aug 12 · 3.7K · 48

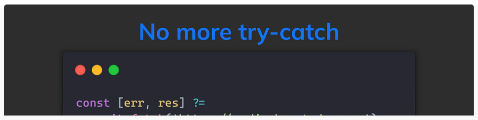
Lists

Stories to Help You Grow as a Software Developer
19 stories · 1531 saves

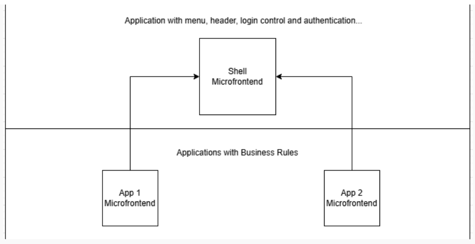
General Coding Knowledge
20 stories · 1832 saves

Generative AI Recommended Reading
52 stories · 1569 saves

Visual Storytellers Playlist
61 stories · 567 saves



```
const [err, res] ?=
```


Application with menu, header, login control and authentication...
Shell Microfrontend
Applications with Business Rules
App 1 Microfrontend
App 2 Microfrontend

In Coding Beauty by Tari Ibaba

This new JavaScript operator is an absolute game changer

Say goodbye to try-catch

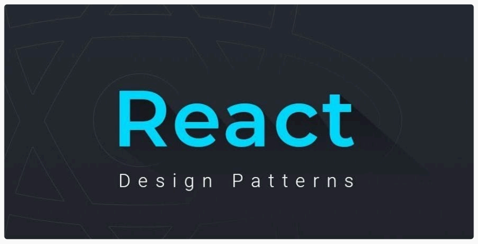
Sep 17 6.1K 95

Pedro Felipe Elias De Souza

Microsoft MSAL Authentication with Angular and MicroFrontend...

A practical guide to integrating MSAL with Angular 15 and Module Federation for...

Dec 18 2



In JavaScript in Plain English by Ferid Brković

Web Components Will Kill JavaScript Frameworks...

The year 2025 just started, and you're building a web app.

Dec 14 602 23

Amber Fung

React Pattern—Container/Presentational Pattern

Container/Presentational pattern is a React design pattern that separates the...

Jul 30 52

See more recommendations