Web Workers vs Shared Workers:

Modern web applications must be fast, responsive, resilient, and multi-tab aware.

JavaScript normally runs on a single thread, which means heavy tasks (loops, calculations) can freeze the UI.
To solve this problem run scripts in the background:

Web Workers

Shared Workers

Dedicated Web Worker

What is a Web Worker?

A Web Worker is a background JavaScript thread that is dedicated to a single web page.

A Dedicated Web Worker:

- Runs in a separate thread
- Is tied to one page/tab
- Has no DOM or UI access
- Communicates via postMessage

When to Use Web Workers

Use a Web Worker when:

- One page needs heavy computation
- You want to keep UI smooth
- No cross-tab coordination is needed

Common Use Cases

- JSON parsing
- Image/video processing

- Cryptography
- Large data transformations
- Search indexing

Advantages

Easy to use
Improves performance
Prevents UI freezing

Limitations

Cannot be shared across tabs
Cannot access DOM
Page-specific only

Shared Worker

What is a Shared Worker?

A Shared Worker is a background worker that can be shared by multiple pages or tabs from the same origin.

A SharedWorker:

- Is shared by multiple tabs/windows
- Belongs to the same origin
- Maintains in-memory shared state
- Uses MessagePort instead of onmessage

Why Shared Workers Exist

Without Shared Workers:

- Each tab opens its own WebSocket
- Each tab maintains duplicate state
- Backend load increases

Shared Workers solve this by acting as a single coordinator.

When to Use Shared Workers

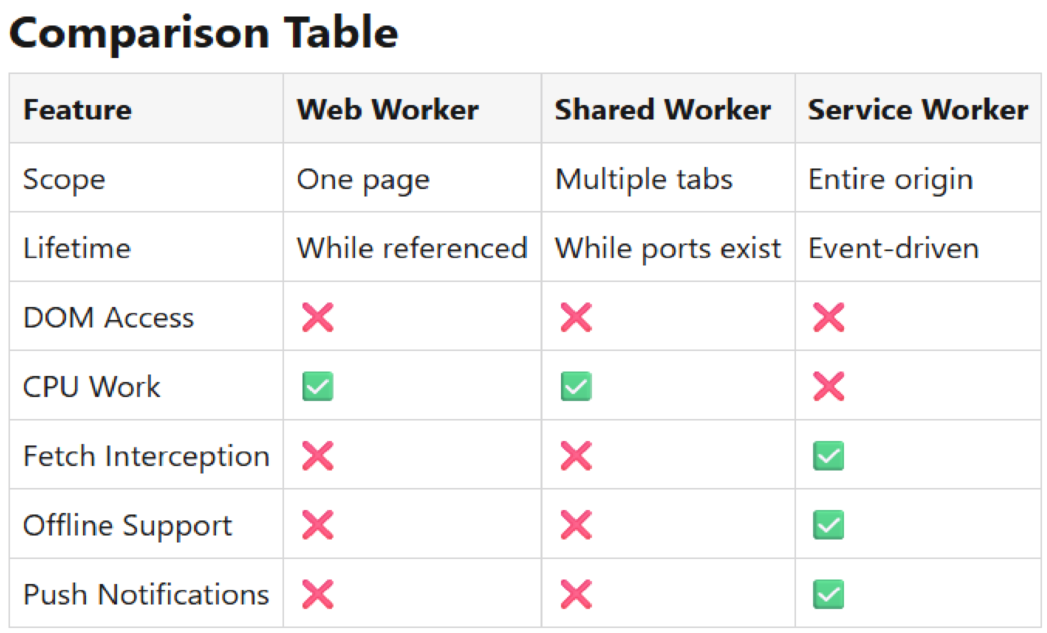
Use a SharedWorker when:

- Multiple tabs need shared state
- You want one WebSocket per user
- You need cross-tab coordination

Advantages

Shared state across tabs
Efficient resource usage
Persistent background process

Limitations
More complex to implement
Limited browser support (not supported in some mobile browsers)
Cannot access DOM



Comparison Table

Feature	Web Worker	Shared Worker	Service Worker
Scope	One page	Multiple tabs	Entire origin
Lifetime	While referenced	While ports exist	Event-driven
DOM Access	❌	❌	❌
CPU Work	✅	✅	❌
Fetch Interception	❌	❌	✅
Offline Support	❌	❌	✅
Push Notifications	❌	❌	✅