Polyglot programming in the context of advanced web engineering

Or why Web platform browsers are awesome ?

Why Web platform browsers are awesome ?

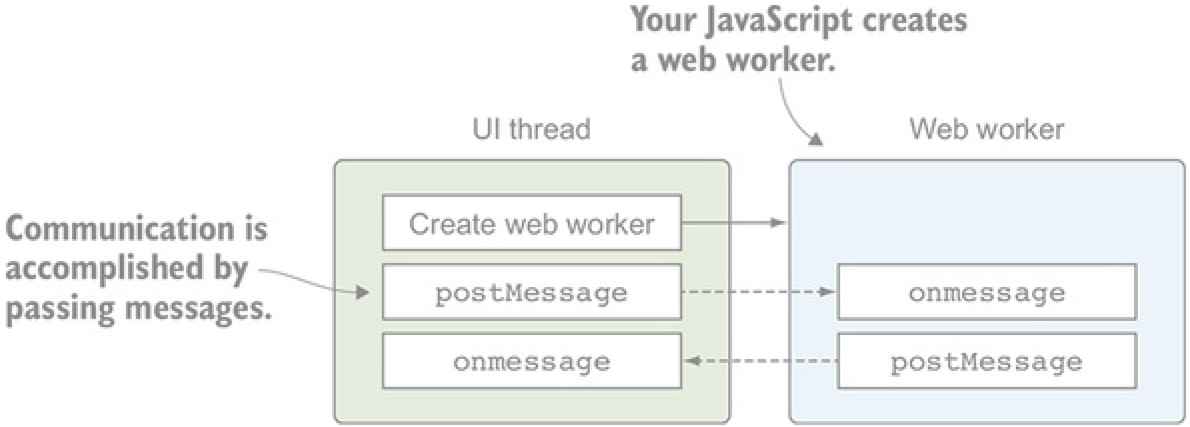
- Web Worker
- Service Worker
- WebAssembly
- OPFS
- …

Web Worker

Could you explain me what is a **Web worker** in the context of web technologies?

A **web worker** is a JavaScript script that runs in the background, independently of other scripts, without affecting the performance of the page. This allows for long-running scripts that do not interfere with the user interface, and also allows for the **concurrent execution** of multiple scripts, leading to a more responsive and faster web page. Web workers are supported by most modern web browsers, and they can be used for tasks such as processing large amounts of data, running complex calculations, or listening for server-sent events.

Web worker


Your JavaScript creates a web worker.
UI thread
Web worker
Communication is accomplished by passing messages.
Create web worker
postMessage
onmessage
onmessage
postMessage

Web Worker

- from UI thread

```
if (typeof Worker !== 'undefined') {
  // Create a new
  const worker = new Worker(new URL('./app.worker', import.meta.url));
  worker.onmessage = ({ data }) => {
    console.log(`page got message: ${data}`);
  };
  worker.postMessage('hello');
} else {
  // Web workers are not supported in this environment.
  // You should add a fallback so that your program still executes correctly.
}
```

- Web worker

```
addEventListener('message', ({ data }) => {
  const response = `worker response to ${data}`;
  postMessage(response);
});
```

Service Worker

Could you explain me what is a **service worker** in the context of web technologies?

A **service worker** is a script that runs in the background, separate from a web page, and **acts as a proxy** between a web page and the network. This allows service workers to **intercept and handle network requests**, including programmatically managing a cache of responses.

Service workers enable features such as offline support, background sync, push notifications and improve the performance of web apps by intercepting network requests and serving responses from a local cache, instead of going to the network. Service workers are supported by most modern web browsers and are a key technology for building **Progressive Web Applications** (PWAs). PWAs are web apps that can be installed on a user's device and run offline, similar to native mobile apps.

WebAssembly

> WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.

- **Efficient and fast** The Wasm stack machine is designed to be encoded in a size- and load-time-efficient binary format. WebAssembly aims to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms.
- **Safe** WebAssembly describes a memory-safe, sandboxed execution environment that may even be implemented inside existing JavaScript virtual machines. When embedded in the web, WebAssembly will enforce the same-origin and permissions security policies of the browser.
- **Open and debuggable** WebAssembly is designed to be pretty-printed in a textual format for debugging, testing, experimenting, optimizing, learning, teaching, and writing programs by hand.
- **Part of the open web platform** WebAssembly is designed to maintain the versionless, feature-tested, and backwards-compatible nature of the web. WebAssembly modules will be able to call into and out of the JavaScript context and access browser functionality through the same Web APIs accessible from JavaScript.

OPFS

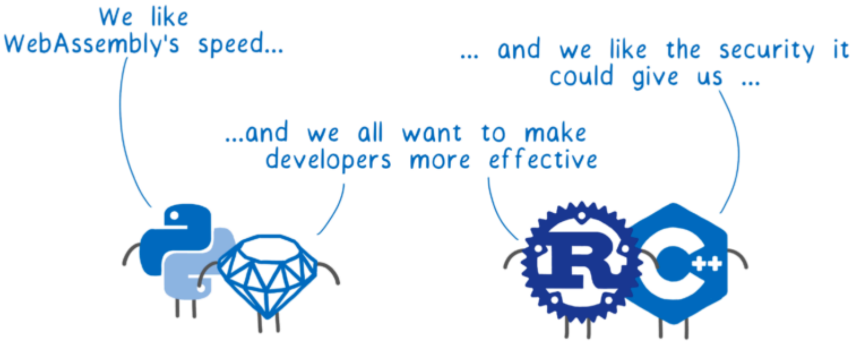
- The Origin Private File System (OPFS, part of the File System Access API) is augmented with a new surface that brings very performant access to data. This new surface differs from existing ones by offering in-place and exclusive write access to a file’s content. This change, along with the ability to consistently read unflushed modifications and the availability of a synchronous variant on dedicated workers, significantly improves performance and unblocks new use cases.

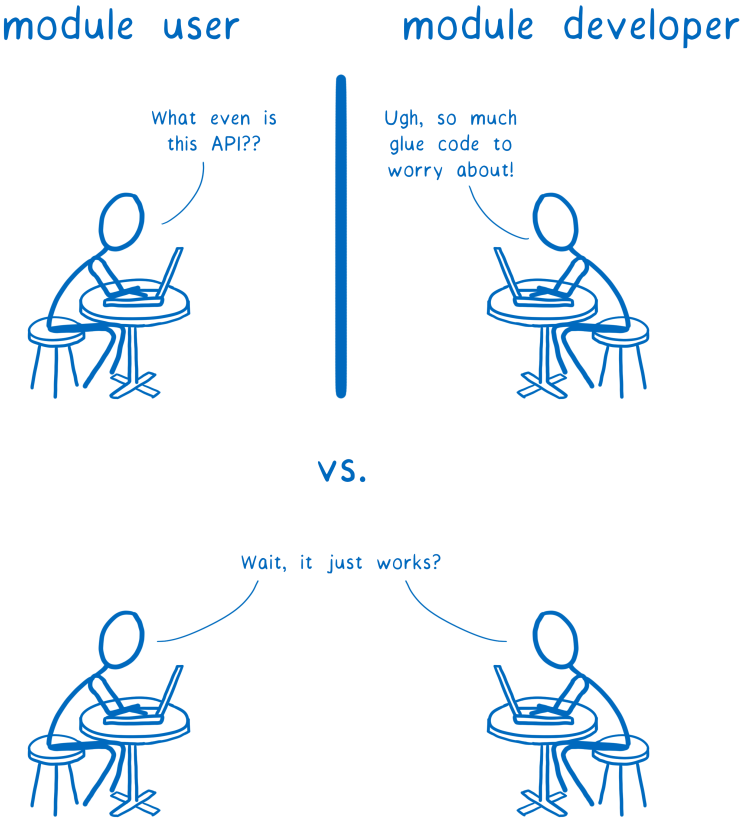
- Release with Chrome 109 one week ago.

Web worker, web assembly, OPFS, WEBGL

- You can do thread,
- You can compile C code to WASM
- You have access to a real efficient FS.
- Ready to run advanced application within the browser 🚀
- A short demo with correctExam

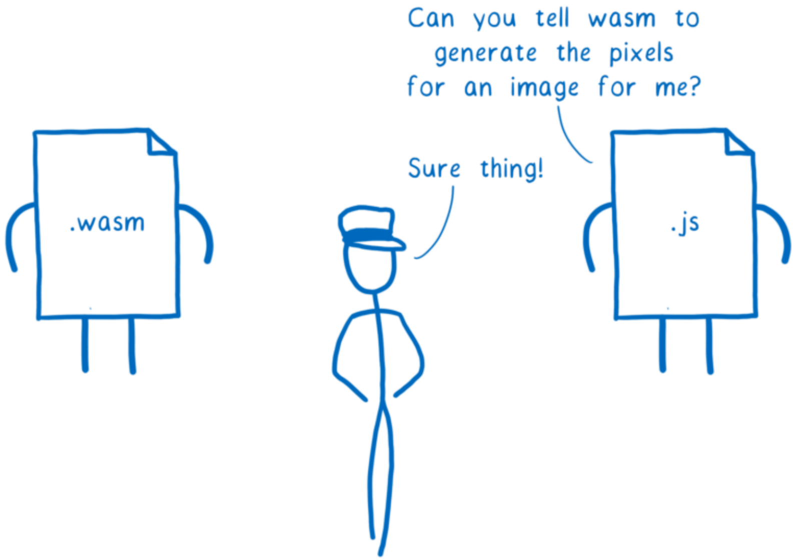
Web worker and web assembly



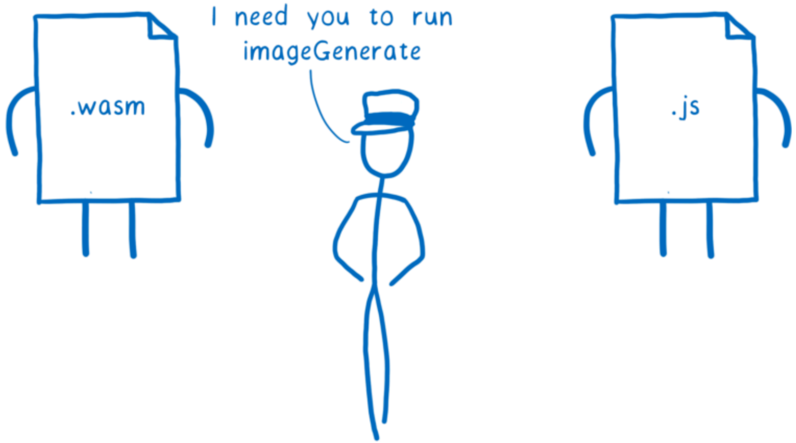


Web worker and web assembly

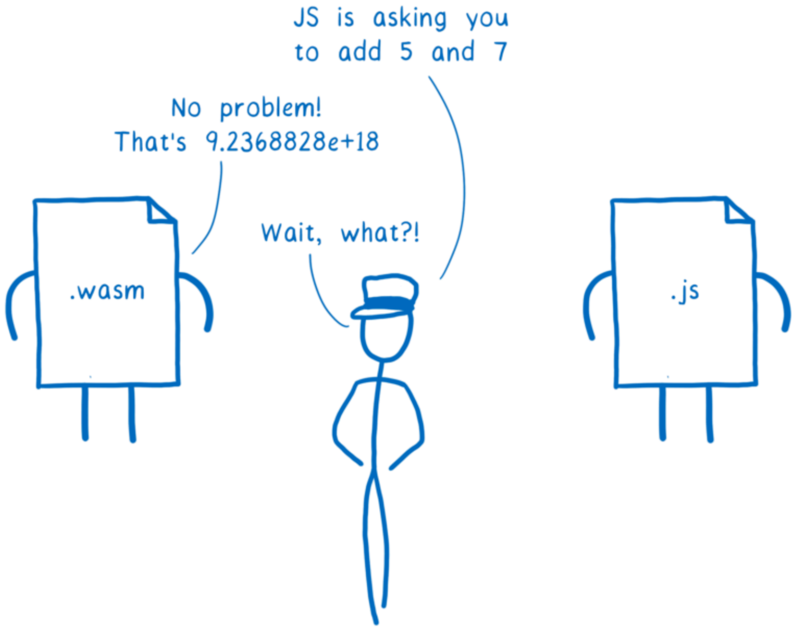
- **Step 1**

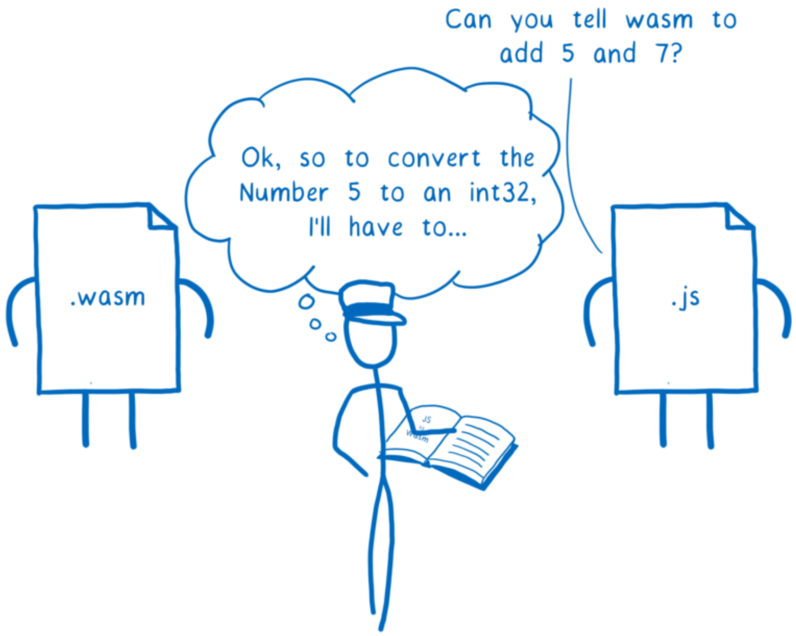


- **Step 2**



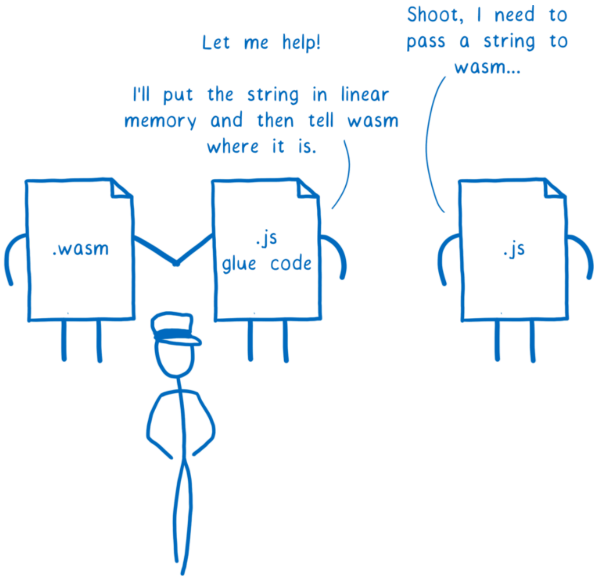
What's happen in the real life

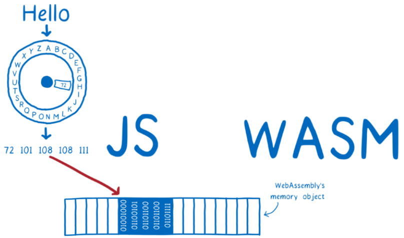


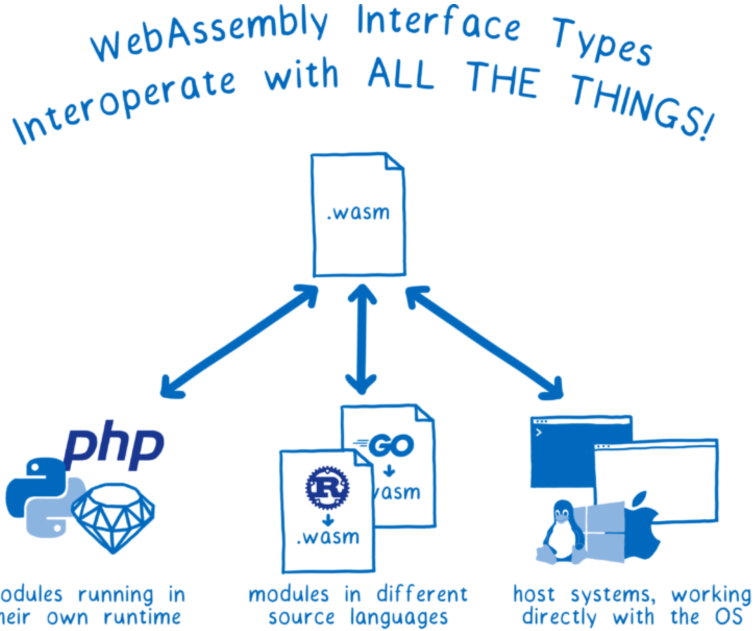


Classical FFI problems

- need a stub/skeleton







Rust to the rescue

- https://rustwasm.github.io/book/why-rust-and-webassembly.html

Time For Demo

Learn More

Documentations · GitHub · Showcases