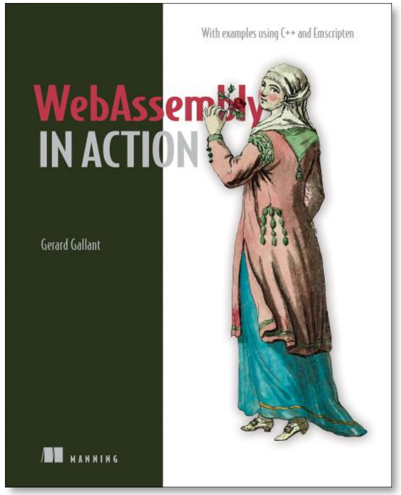
# Using WebAssembly in a Web Component as a Polyfill

Talk Feedback

# Gerard Gallant

CIO at Dovico Software



“WebAssembly in Action”
bit.ly/37zJbp5

@Gerard_Gallant

linkedin.com/in/gerard-gallant

# Demo 1

# Options if an image type is not Supported

1. Go with a format that is supported: **AVIF, JPEG, etc**

2. Use the **Picture** element and fall back to a type that is supported:

   ```
   <picture>
     <source srcset="format-we-want.jxl" type="image/jxl" />
     <img src="format-that-will-have-to-do.avif" alt= "an ok image" />
   </picture>
   ```

3. Generate the alternate images during CI/CD

4. Use **WebAssembly**

# WebAssembly (wasm)

In Modern Browsers & Node.js

**On the server, IoT, or in your code** thanks to runtimes like Wasmtime, Wasmer, WasmEdge, WAMR, and others

**Edge and Serverless Compute**
Fermyon, Cloudflare, Fastly, and others

**Containers and Micro-VMs**
Docker, Kubernetes, and Hyperlight

# SharedArrayBuffer Access

**Cross-Origin-Opener-Policy** (*COOP*) with the value **same-origin**

**Cross-Origin-Embedder-Policy** (*COEP*) with the value **require-corp**

**crossorigin** attribute to allow the load of files from things like a CDN if you trust the link

```
<script src="some link" crossorigin></script>
```

The blog post: **bit.ly/2Foeg4b**

# Demo 2

# Links

libjxl: github.com/libjxl/libjxl

Rust version of libjxl: github.com/tirr-c/jxl-oxide

Site to convert JPEG to JPEG XL: cloudinary.com/tools/jpg-to-jxl

Mutation Observer approach: github.com/niutech/jxl.js

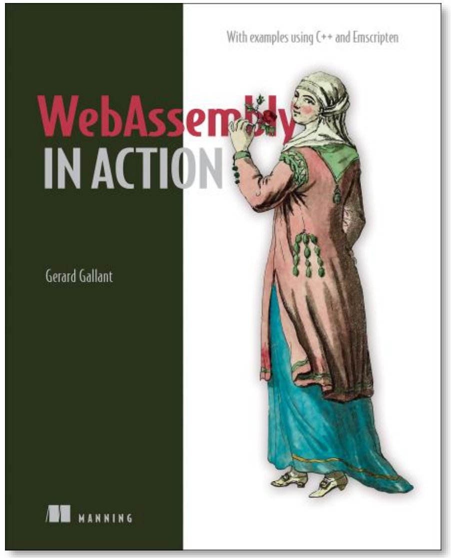
Service worker approach: github.com/libjxl/libjxl/blob/main/tools/wasm_demo/README.md

WebAssembly threading article: bit.ly/2Foeg4b

PGLite: pglite.dev

SQLite: sql.js.org

Digital version of my book where the 1st chapter is free: bit.ly/37zJbp5



Today's code and slides:
**bit.ly/4ieto0T**

@Gerard_Gallant

linkedin.com/in/gerard-gallant

# Thank You

Talk Feedback