WebAssembly - a new technology and its potential for geospatial application

Pirmin Kalberer
@implgeo
Sourcepole, Switzerland
www.sourcepole.com
What is WebAssembly?

- WebAssembly (Wasm) is a simple machine model and binary executable format.
- Designed to be portable, compact, and execute at or near native speed.
- Memory-safe, sandboxed execution environment.
- A Wasm module has access to a single "linear memory", which is essentially a flat array of bytes.
(module
  (func $fac (param f64) (result f64)
    get_local 0
    f64.const 1
    f64.lt
    if (result f64)
      f64.const 1
    else
      get_local 0
      get_local 0
      f64.const 1
      f64.sub
      call $fac
      f64.mul
    end)
  (export "fac" (func $fac)))

WebAssembly Text (wat)
asm.js

- 2013: Firefox 22
- Subset of JavaScript
- Can be generated with Emscripten
- Fallback for Browsers without Wasm support
Generating WebAssembly

- C / C++

  ![Diagram](image)

- Rust

- With GC: C++, Go, ...

- Assembly Script

- Writing WebAssembly directly (wat)
What can be done in Wasm

- **Supported:**
  - Calculations
  - Calling Javascript (and vice versa)
  - Futures
  - Transferring data via linear memory (e.g. Canvas data)

- **Preview:**
  - Multithreading

- **Can’t be done (yet):**
  - Direct DOM access
  - Direct Web API access (via Javascript calls only)
Browser support

- Firefox 52 (March 2017)
- Chrome 57 (March 2017)
- Safari 11 (Sept. 2017)
- Edge 16 (Oct. 2017)
- Android, iOS
- No support: IE

https://caniuse.com/#feat=wasm
Specifications

- [https://webassembly.org/](https://webassembly.org/)
- W3C WebAssembly Community Group
- W3C WebAssembly Working Group
  - WebAssembly Specification Release 1.0 (Draft, May 30, 2019)
Applications: Autocad

- https://web.autocad.com/
- Google I/O Keynote
Applications: Google Earth

- https://g.co/earth/beta
- Chromium Blog
Libraries

- Client-side projection engine (ESRI Blog)
- Qt Demos & Examples
- Skia (OSS C++ graphics library)
- PSPDFKit
Demo applications

- Liquid simulation (Demo)
- VR: Julia set on a planar object (Demo)
- Doom 3
- WasmBoy / VaporBoy (GameBoy emulators)
- Vim
- Nginx web server
- Python interpreter, Go Compiler, ...
Chapter 2: WASM Runtimes

- **Browser**
- **Standalone**
  - Wasmer
  - Wasmtime
  - Lucet
  - Intel WebAssembly Micro Runtime (WAMR)
- **Standard: WASI** ([https://wasi.dev/](https://wasi.dev/))
- **Embedding Wasm in other languages**
  - Wasmer: Rust, C/C++, Python, Ruby, Go, C++, PHP
- **Runtime as a service**
  - Cloudflare workers
What does that mean?

- Portable binary code between different platforms
- Alternative to NodeJS, Electron, ...?
- Common runtime format for multilingual Edge/Serverless applications (sandboxed!)
- Common runtime format for application plugins/extensions (sandboxed!)
- Portable runtime format for embedded/IoT devices?
Geospatial applications

▷ Top candidates
  ◦ Proj
  ◦ GEOS

▷ Ideas
  ◦ Client-side map printing (PDF)
  ◦ Rendering vector tiles (MVT V3)
  ◦ Importing Shapefiles with GDAL
  ◦ Routing
  ◦ Desktop GIS in the browser?
    ◦ Complex GUI → possible, but a lot of work
    ◦ Web optimized data formats: COG, FlatGeobuf
Wasm Speed

- Expectation:
  - 20-30% faster than Javascript

- PSPDFKit benchmark (early 2018)
  - Slower than Javascript on most browsers, 70% faster on Firefox

- WasmBoy benchmark
  - About 30% faster than JS on Chrome. Can be faster at around 60% on mobile and on Firefox it can be faster around 90%. Slower than 30% on Safari.
Comparison WASM / Native

World Map Generator 1000x1000 cells

- Native: 2.9s
- Wasm (2D) Firefox: 6.4s
  - 45% of native speed
Conclusion / Outlook

- Wasm is here
- FOSS4G will catch up
- Funding needed for core libraries?
- Wasm will get even better
Thank you!

Pirmin Kalberer
@implgeo