Comparing WebAssembly and JavaScript for a web based photo editor

Sam Robbins

Durham University

What is the project about?

For a long time, JavaScript was the only method for doing computation in web browsers, however in 2017, Web Assembly was introduced as an alternative [1].

Definition (Web Assembly)

Web Assembly is a language that allows for the execution of languages that compile to assembly on the web

Research Question

The Research question is to find in which cases Web Assembly offers a benefit over JavaScript, finding this out by implementing image processing algorithms using both mechanisms.

Deliverables

The basic aim is compare the performance of one complex algorithm between Web Assembly and JPEG, for a complex algorithm, it should not simply loop over the pixels of the image, and instead be performing more sophisticated techniques. The intermediate aim is to extend this to multiple complex algorithms. The advanced aim is to look into tweaks that can be used to improve performance, such as compile time optimizations for Web Assembly.

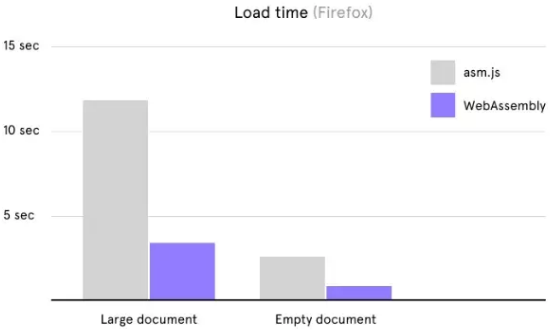
What has been done by others?

Using Web Assembly for image processing — Next.js

.JS

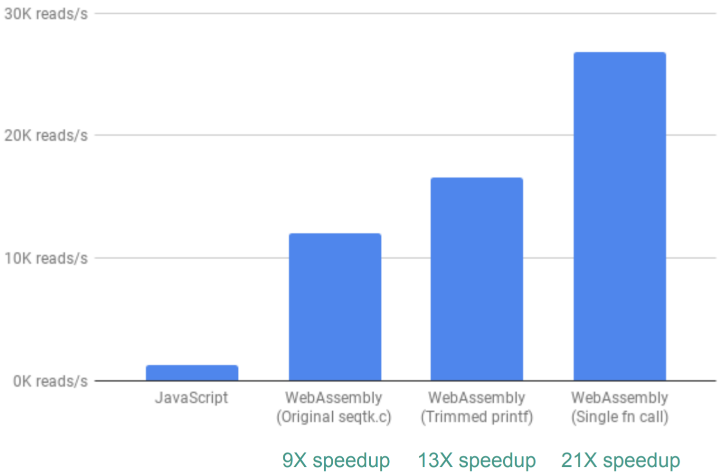
96.5 MB $\Rightarrow$ 69.2 MB

Performance of Web Assembly — Figma



$3\times$ faster load times

Performance of Web Assembly — Fastq.bio



Performance of Web Assembly — Compared to native

Benchmark	Field	Native time(s)	Google Chrome time(s)
bzip2	Compression	370	864
mcf	Combinatorial	221	180
milc	Chromodynamics	375	369
namd	Molecular Dynamics	271	369
gobmk	Artificial Intelligence	352	537

How does it compare?

My Solution

Brightness

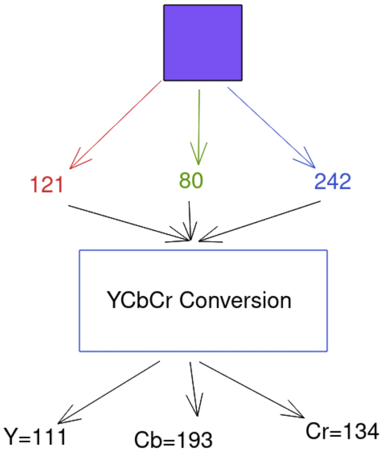
 

Contrast

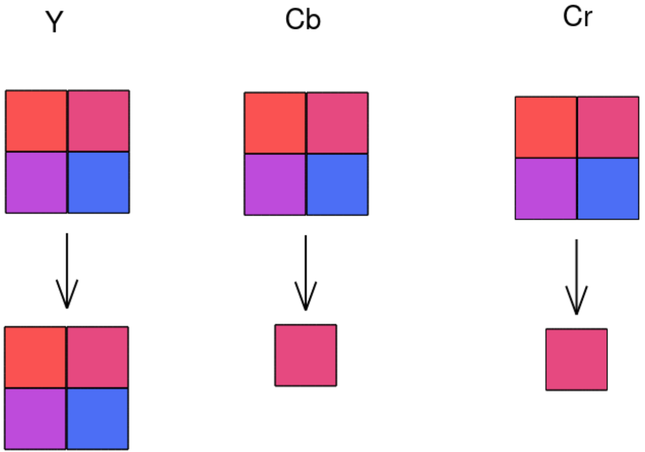
 

Gaussian Blur

JPEG — YCbCr Conversion



JPEG — Downsampling



JPEG — DCT

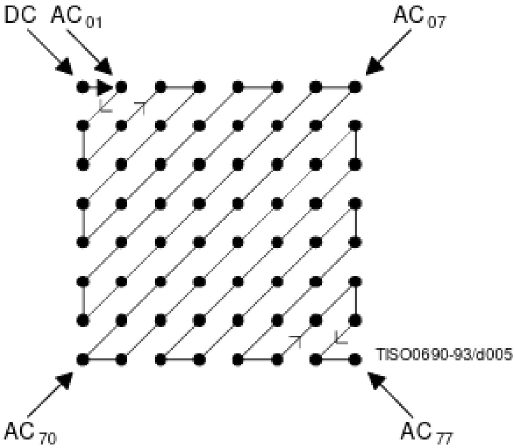
$$T_{i,j} = \begin{cases} 1/\sqrt{N} & \text{if } i = 0 \\ \sqrt{\frac{2}{N}}\cos(\frac{(2j+1)i\pi}{2N}) & \text{if } i > 0 \end{cases}$$

N is the dimension of the matrix (here 8)

JPEG — Quantization

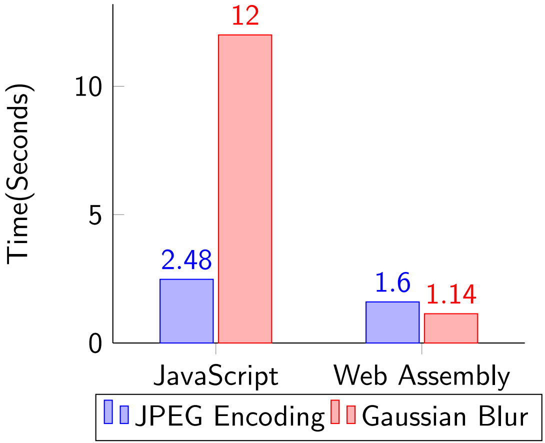
$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

JPEG — Zigzag

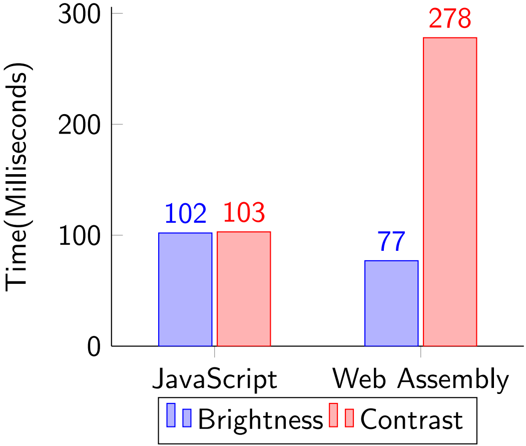


Results

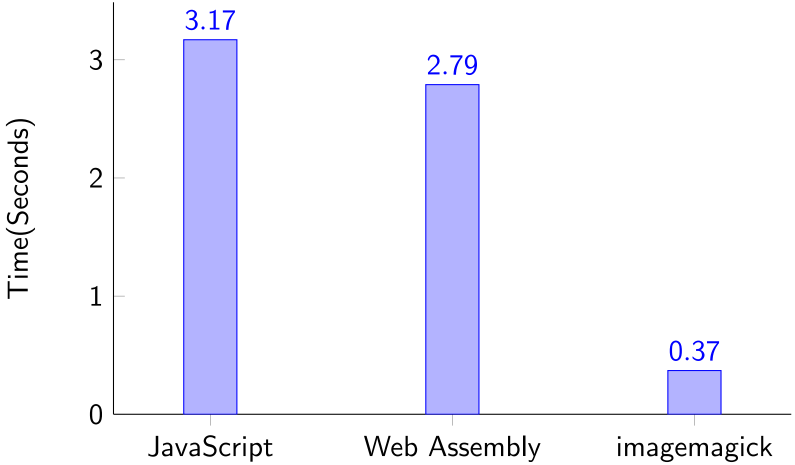
Complex Algorithms



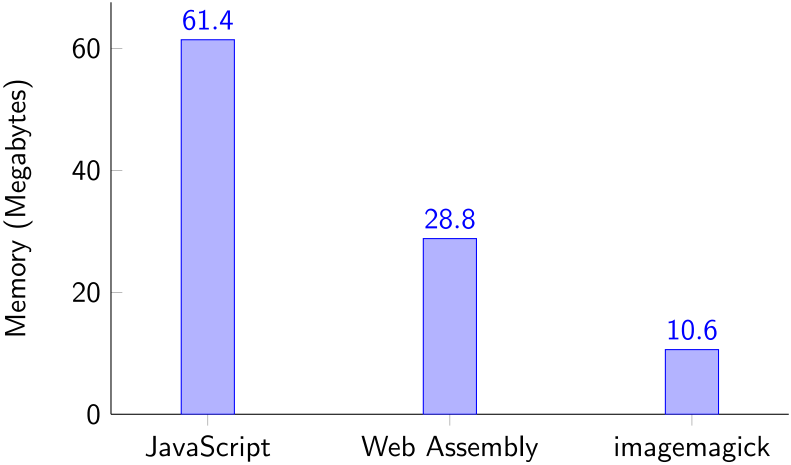
Simple Algorithms



Time for the full process



Memory consumption of the full process



Evaluation

- ✓ Faster
- ✓ Lower memory consumption
- ✓ Static types
- ✗ Static types
- ✗ Slower than native
- ✗ Need to use multiple languages
- ✗ Libraries have had less development

Further Work

- WebGPU
- Different methods
- Different languages

References I

[1] Andreas Haas et al. “Bringing the web up to speed with WebAssembly”. In: *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2017, pp. 185–200.