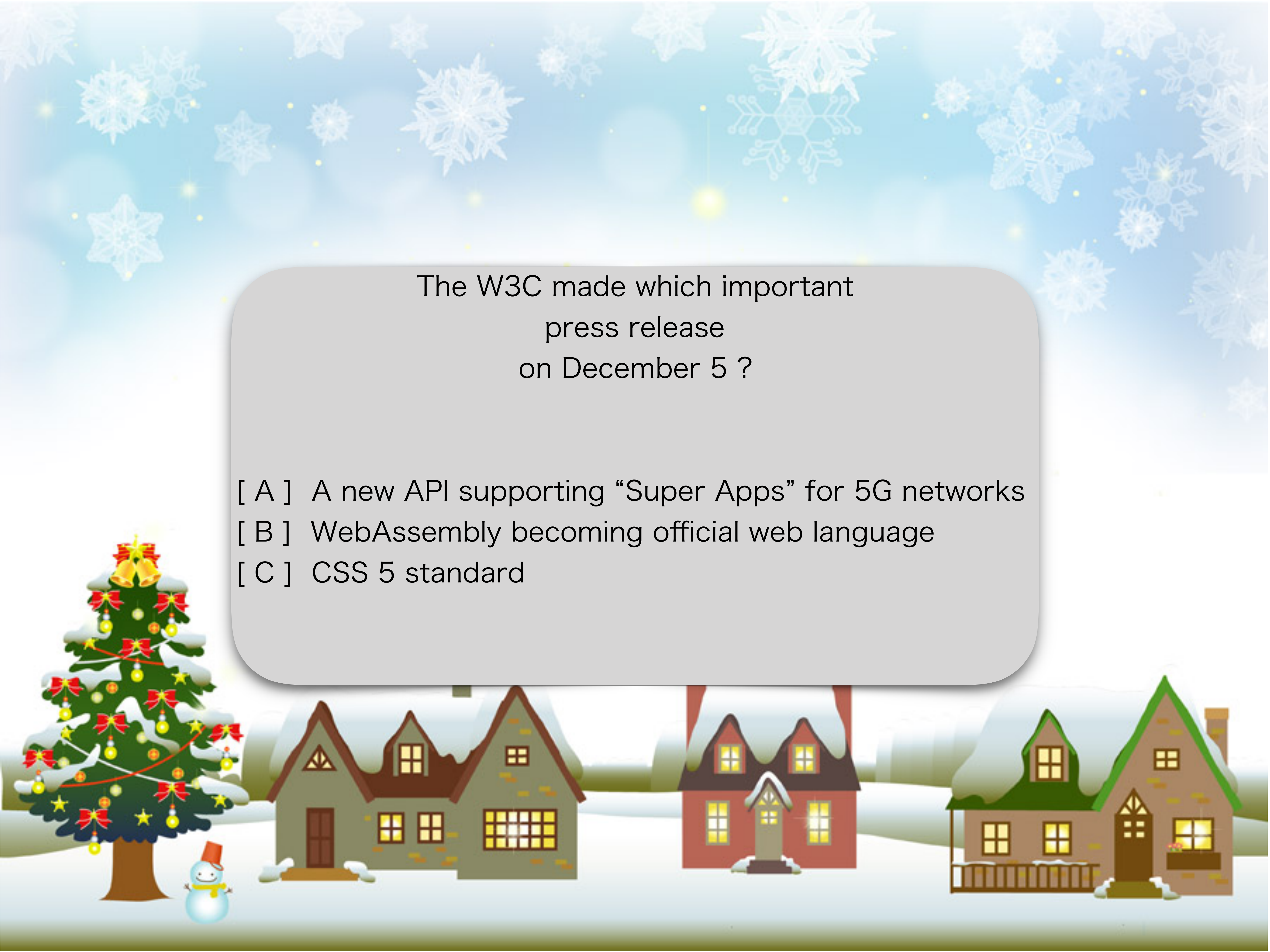
Native goes Web

WebHack Tokyo, Dec. 17 2019

Intro

Warmup Quiz



The W3C made which important
press release
on December 5 ?

[A] A new API supporting “Super Apps” for 5G networks
[B] WebAssembly becoming official web language
[C] CSS 5 standard

The W3C made which important press release on December 5 ?

Correct answer is B.

[B] WebAssembly becoming official web language

https://www.w3.org/2019/12/pressrelease-wasm-rec.html.en

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About the presentation

- Bringing mangas to your browser
- Short retrospective about native code in web
- Why WebAssembly matters
- Tool box
- WebAssembly with Rust in action
- Conclusion
- Your questions

About me

Name: Ziad Jabri

Nationality: German

Living and working in Japan since 2006.

- 22 yrs professional development experience
  - 10 yrs desktop, 5 yrs embedded
  - mobile and web since 2012
- WebFront Scrum Master & Developer at U-NEXT
  - video.unext.jp
  - U-NEXT app for PS4
- Interested in agile methodologies, DDD, DevOps and modern compiler languages like Rust

U-NEXT

Delivering mangas to your browser

U-NEXT books outline

- Major renewal Mar 2019 and integration into main service web site and mobile app
- Contents in numbers [video.unext.jp/introduction]
    - 220K+ mangas
    - 170K+ ebooks
    - 30K+ light novels
    - 70+ magazines

U-NEXT books

- Download and read in mobile app
- Open in your desktop/mobile browser and read on the go
- Read book sample in mobile browser

U-NEXT

Teaser

U-NEXT book viewer for browsers

U-NEXT manga: the background story

- started design and implementation of a prototype for a browser based magazine and manga viewer back in 2016

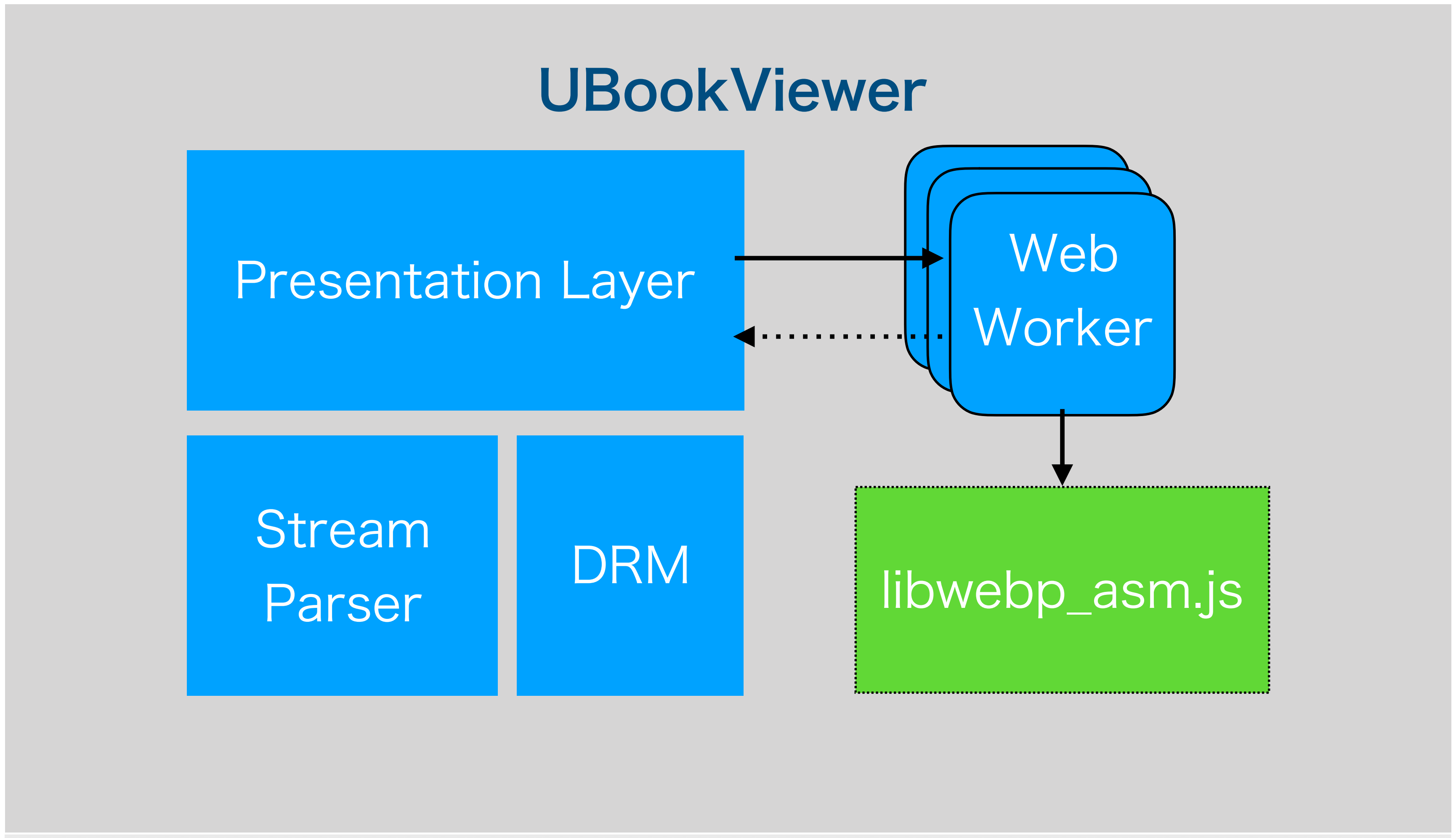
U-NEXT manga: the technical challenge

- each page is a high resolution bitmap
- possible bitmap formats:
  - PNG —> too large
  - JPG —> compression not good enough for text in images
  - WEBP —> low text artifacts and high compression rate
- problem:
  - most browsers cannot decode WEBP

The idea

- WEBP decoding in JavaScript ?
    - actually there is no decoder written from ground up in JavaScript; computational complexity to high for achieving acceptable performance
    - most decoders are written in near machine level languages e.g. C, C++, Assembler
- Experimental branch of libwebp for asm.js
  Asm.js: making C code compile to JavaScript… that's it !!!

The prototype



A short retrospective about native code in web pages

Cemetery of native code technologies

- Java Applet (1995-2015)
  - JVM parallel to browser, sandboxed APIs
- ActiveX (1996-)
  - Intel x86 DLL, full OS access, trust based
- Silverlight (2007-2012)
  - .NET core sub set, mainly for multimedia

What happened ?

- modern highly performant JS Engines (e.g. Google V8) got on par
- HTML5 / CSS3 and modern Web APIs
- mobile browsers which don’t support them
- mobile native apps killed them

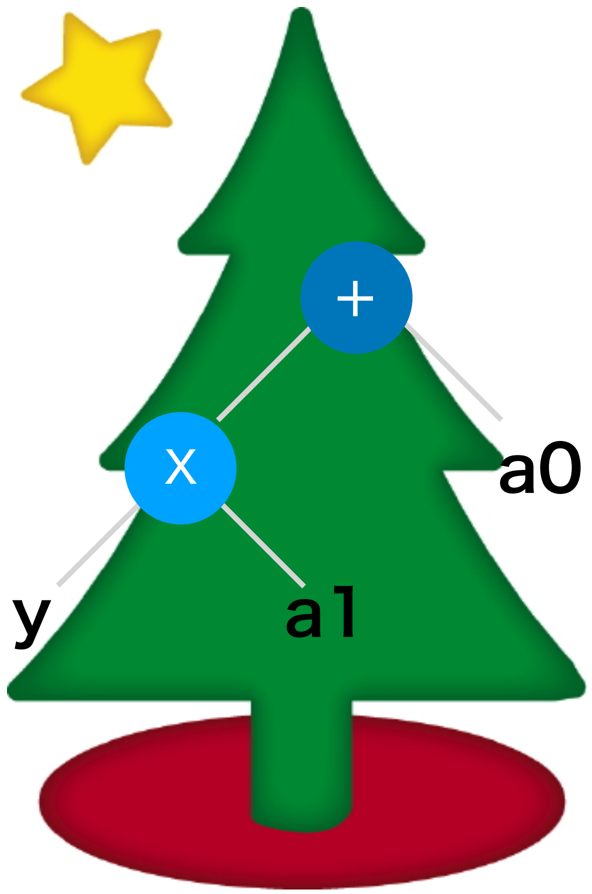
Why WebAssembly makes a difference

Web + Assembly = ?

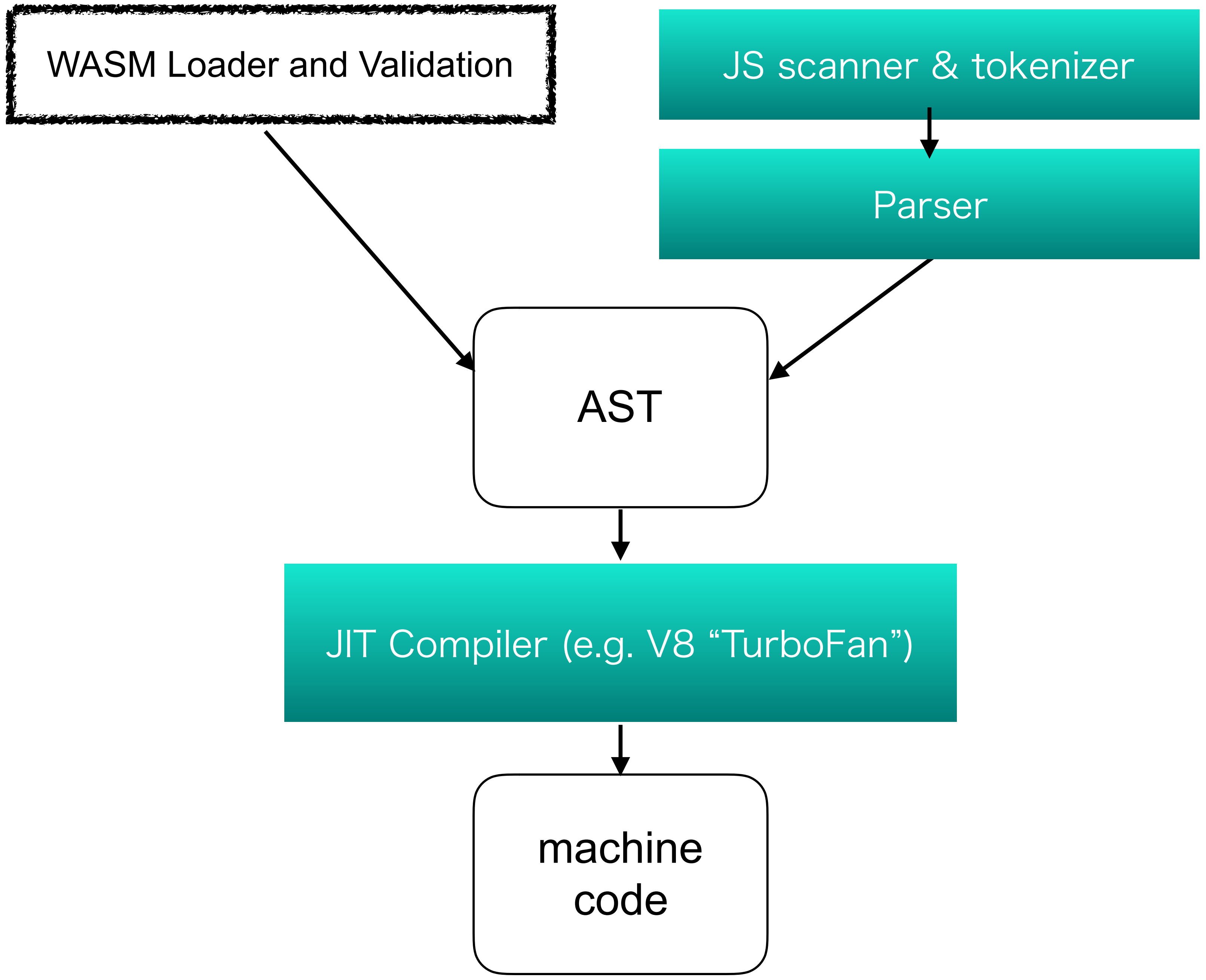
- Portable low level byte code designed for fast compilation to machine code by the WebAssembly host (browser, server, IoT)
- Each module instance executed in its own address space and isolated from other modules (sandboxed)
- Static data types and fixed argument count
- Calls from/to host

Abstract Syntax Tree

```
function poly1(a0: number, a1: number, y: number) : number {
  return a0 + a1 * y;
}
```



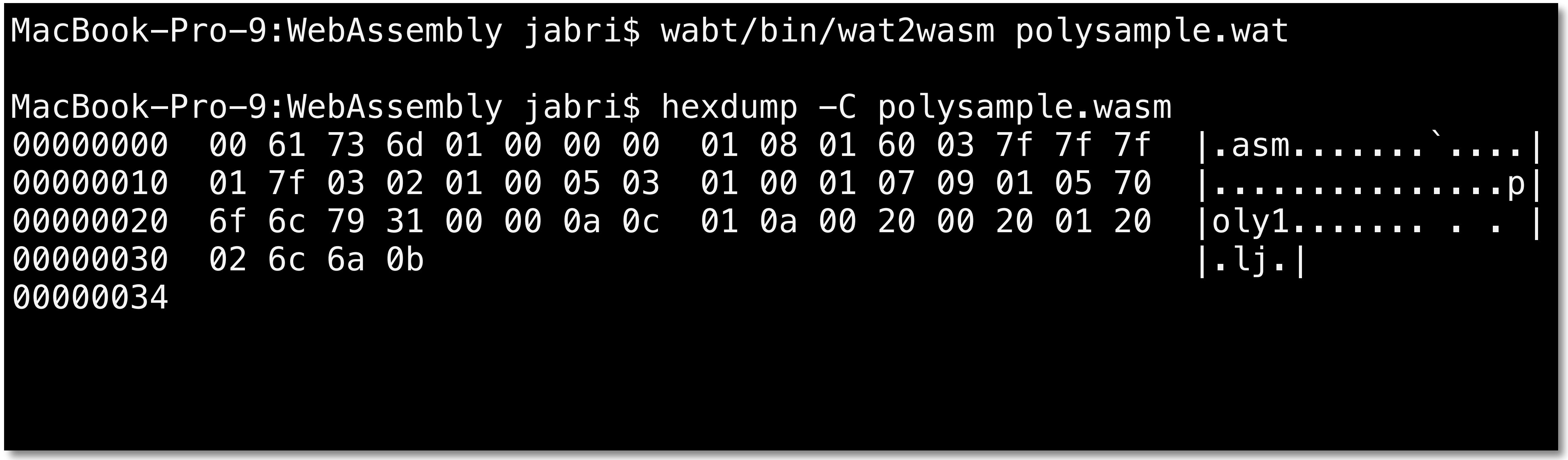
fAST lane


WASM Loader and Validation
JS scanner & tokenizer
Parser
AST
JIT Compiler (e.g. V8 “TurboFan”)
machine code

WebAssembly Text

```
(module
 (memory $mem 1)
 (func $poly1 (param $a0 i32) (param $a1 i32) (param $y i32) (result i32)
   (i32.add
     (get_local $a0)
     (i32.mul (get_local $a1) (get_local $y))
   )
 )
 (export "poly1" (func $poly1))
)
```

Building WASM binary



```
MacBook-Pro-9:WebAssembly jabri$ wabt/bin/wat2wasm polysample.wat

MacBook-Pro-9:WebAssembly jabri$ hexdump -C polysample.wasm
00000000  00 61 73 6d 01 00 00 00  01 08 01 60 03 7f 7f 7f  |.asm.......`....|
00000010  01 7f 03 02 01 00 05 03  01 00 01 07 09 01 05 70  |...............p|
00000020  6f 6c 79 31 00 00 0a 0c  01 0a 00 20 00 20 01 20  |oly1....... . . |
00000030  02 6c 6a 0b                                       |.lj.|
00000034
```

Load and call from JS

```
const importObject = { imports: {} };

WebAssembly.instantiateStreaming(fetch("./polysample.wasm"), importObject).then(
  obj => {
    const result = obj.instance.exports.poly1(2, 4, 5);
    console.log(`result is ${result}`);
  }
);
```

The tool box

WebAssembly tools

- WebAssembly Binary Toolkit
- Binaryen (library for compiler creators)
- Emscripten (C/C++ compiler to asm.js/wasm)
- Rust compiler (target: wasm32-unknown-unknown)
- Blazer (.NET on WebAssembly)

WebAssembly with Rust in action

Image Manipulation

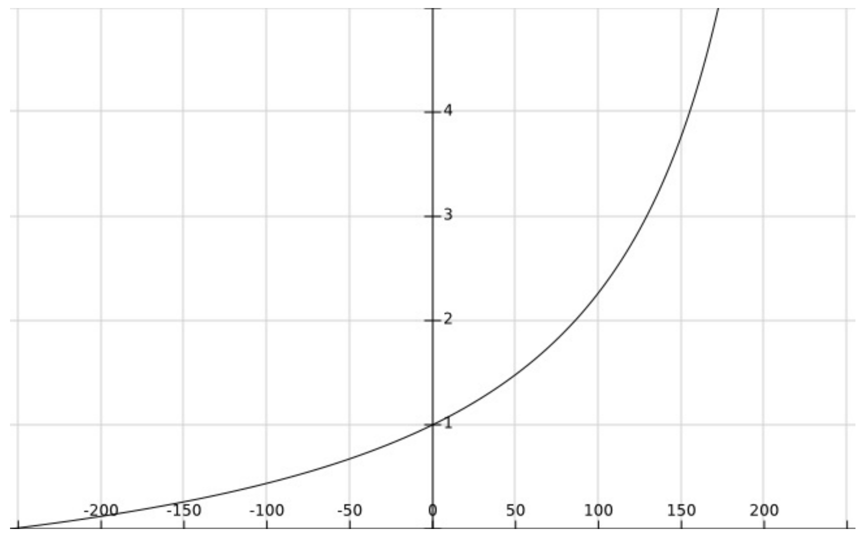
- Apply filters to your images before uploading
- Add your own watermarking on the client side
- Add support for more advanced compression formats (for instance WEBP lossless encoding)

Demo: contrast adjustment

- CanvasContext2d filters only experimental by now
- Using WebGL needs a lot of setup to be done
- Let’s try it with a Rust library compiled to WASM (github.com/WaxWayne/contrast)

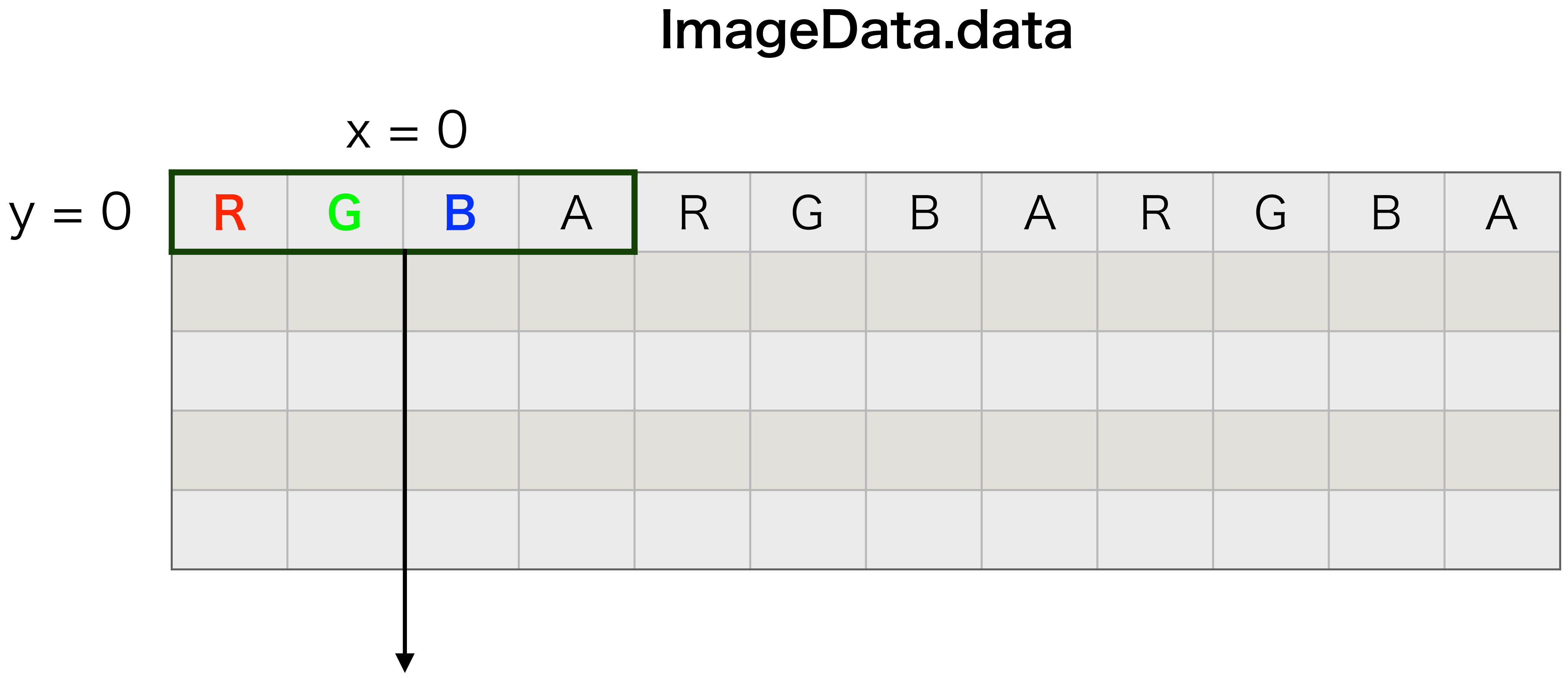
Contrast

$$f = \frac{259 * (c + 255)}{255 * (259 - c)}$$



c	f
-255	0
-128	0.33
0	1
128	3
255	129.5

Applying to canvas



R’ = ClampUint8((R - 128) * f + 128)

G’ = ClampUint8((G - 128) * f + 128)

B’ = ClampUint8((B - 128) * f + 128)

Contrast demo

Other use cases

- Implement binary protocols which are strongly typed between client / server e.g. using WebSockets.
- Advanced encryption schemes / protocols

Conclusion

Rich web apps and new collaborative landscapes

- WebAssembly enables compute intense functions to be run inside the browser with acceptable performances (e.g. HDR photo manipulations ~ 100ms)

- Web developers can tap vast resources of native libraries

- This allows new collaborations between frontend and native library developers

Your questions

Q&A memo

Andrei Krutikov commented and asked:

? had been using asm.js with Emscripten back in 2016, but lack of 64bit pointer and multithreading support made porting of existing C++ code impossible. What about WebAssembly's support for these features ?

! currently shipped version of WebAssembly does not yet support 64bit nor multithreading, but the language will most likely evolve. Those and other new language features are already on the development agenda (see Appendix)

– regarding memory larger than 4GB, this could be probably mitigated by splitting into multiple WASM modules, which communicate through the JavaScript host

? is there source level debugging support ?

! Yes, since source maps are supported. Rust source code, for instance, can be debugged in the browser.

Thank you

Appendix

- W3C press release
  https://www.w3.org/2019/12/pressrelease-wasm-rec.html.en
- Abstract Syntax Trees
  https://medium.com/basecs/leveling-up-ones-parsing-game-with-asts-d7a6fc2400ff
- A little on V8 and WebAssembly
  https://pliss2019.github.io/ben_titzer_webassembly_slides.pdf
- WebAssembly Binary Toolkit
  github.com/WebAssembly/wabt
- Rust & WebAssembly
  https://rustwasm.github.io/docs/book/
- Contrast formula
  https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-5-contrast-adjustment/
- WebAssembly Future Features
  https://github.com/WebAssembly/design/blob/master/FutureFeatures.md