WASM for dummies

A very short primer in a series of acts
#FOSDEMFlashbacks

The very first web browser, Credits: CERN

Divya_Mohan02
ACT I: JavaScript
JavaScript: The Gen Z (undefeated) compilation target

- Developed in 1995 by Brendan Eich
- Low entry level barrier leading to ubiquity
- Undefeated champion of compilation targets ever since
- Non-exhaustive list of competitors
  - ActiveX
  - Adobe Flash
  - JavaBeans

Divya_Mohan02
Cons

- Not designed to be a compilation target
- Weakly typed
- Eventually faster
  - We’ll cover this in the next slide!
About time

Credit: https://blog.devgenius.io/a-primer-on-webassembly-834150fdd7ae

Divya_Mohan02
ACT II: asm.js
Asm.js: The origin story

- Subset of JavaScript
- Spec describes sandboxed VMs for memory-unsafe languages
- Low-level efficient compiler target
- Implemented by Mozilla
Cons

- Not standardized
  - Informal spec
  - Vendor implementations were customized
- Still limited to things that were expressible in JavaScript
ACT III: WebAssembly
What is it?

- Binary instruction format for stack-based virtual machines
- Designed to be a portable compilation target
  - On the web
  - Off it, as well!
- Strongly typed
Will it replace JavaScript?
Demo time!
Why is it better?

- Designed to be a compilation target
  - Allows for more languages to be brought into the Web
- Standardized across the four major browsers
- Faster than JavaScript
- Predictable performance across application
Where are we now?

- Still at v1
- MVP released in 2017
  - A lot of things are still in the pipeline for implementation
- Experimental projects/offerings popping up for:
  - Server-side implementation (like NodeJs for JavaScript)
  - Combining cloud native tech to form,
    - Application frameworks running on Kubernetes (e.g. Atmo)
    - Scheduling wasm modules on cloud native frameworks (e.g. Krustlet)

Divya_Mohan02
Questions?