Intro to WebAssembly

Irfaan Khalid, Matt Leon | CS 3892

Introduction

Hi!

I’m Irfaan.

Fun fact: I’ve rapped in front of over 1000 people before - twice!

Hey!

I’m Matt.

Fun fact: I’ve contributed to VS Code and Parcel on GitHub.

Agenda

Overview	*What are we talking about?*
Use Cases	*When is WebAssembly useful?*
Caveats	*When isn’t WebAssembly useful?*
Technical Demo	*How does it actually work?*
Conclusion	*What have we learned?*

Overview

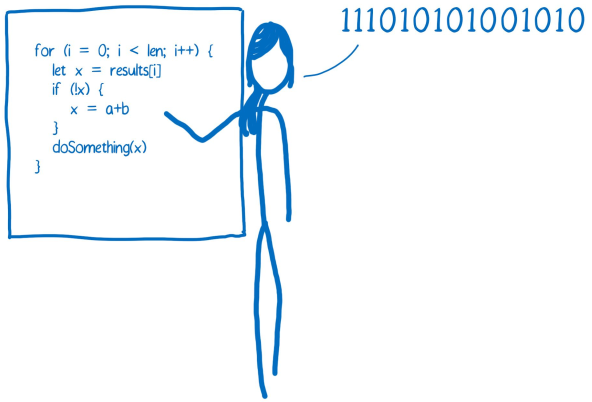
What are we talking about?

What is *WebAssembly*?

- ▷ Efficient binary format
- ▷ Developed by industry leaders
- ▷ Executes at native speed
- ▷ Extremely portable
- ▷ Uses native instructions

What is *JavaScript*?

- Interpreted language designed for the web
- JIT compiled for performance
- Follows event loop
- Single-threaded
- Recently:
  - Optimizing JIT
  - Asynchronous I/O

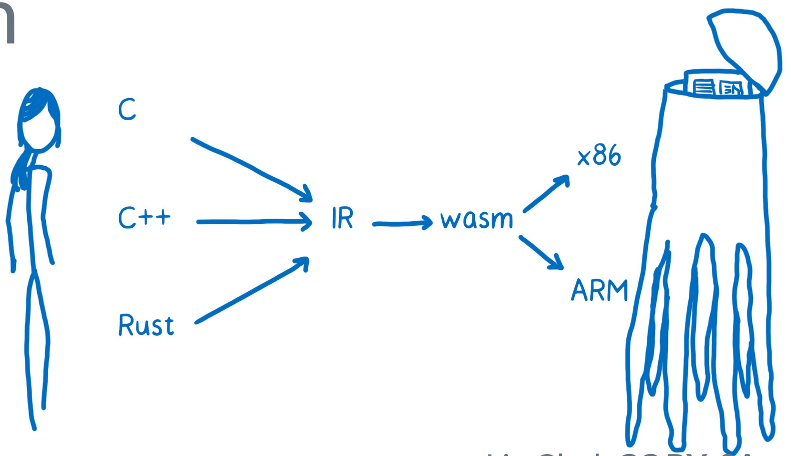


```
for (i = 0; i < len; i++) {
  let x = results[i]
  if (!x) {
    x = a+b
  }
  doSomething(x)
}
```

Lin Clark **CC BY-SA**

What is *WebAssembly*? (V2!)

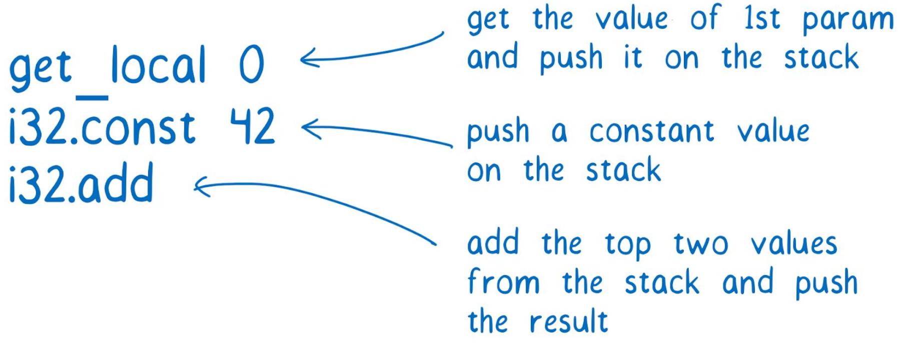
- Intermediate representation for LLVM langs
- Intersection of common machine instructions
- Stack Machine execution
- Compiled & optimized beforehand
- Easy for browser to parse



Lin Clark **CC BY-SA**

Textual representation

```
(module
    (func $add (param $lhs i32) (param $rhs i32) (result i32)
      get_local $lhs
      get_local $rhs
      i32.add)
    (export “add” (func $add))
)
```



```
get_local 0
i32.const 42
i32.add
```

get the value of 1st param and push it on the stack

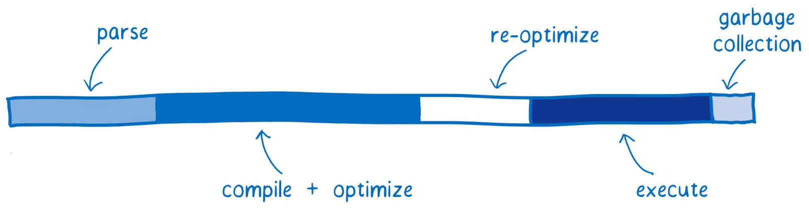
push a constant value on the stack

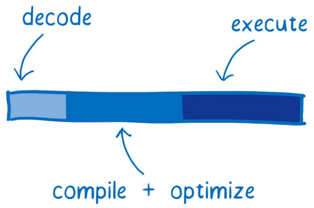
add the top two values from the stack and push the result

Use Cases

When is WebAssembly useful?

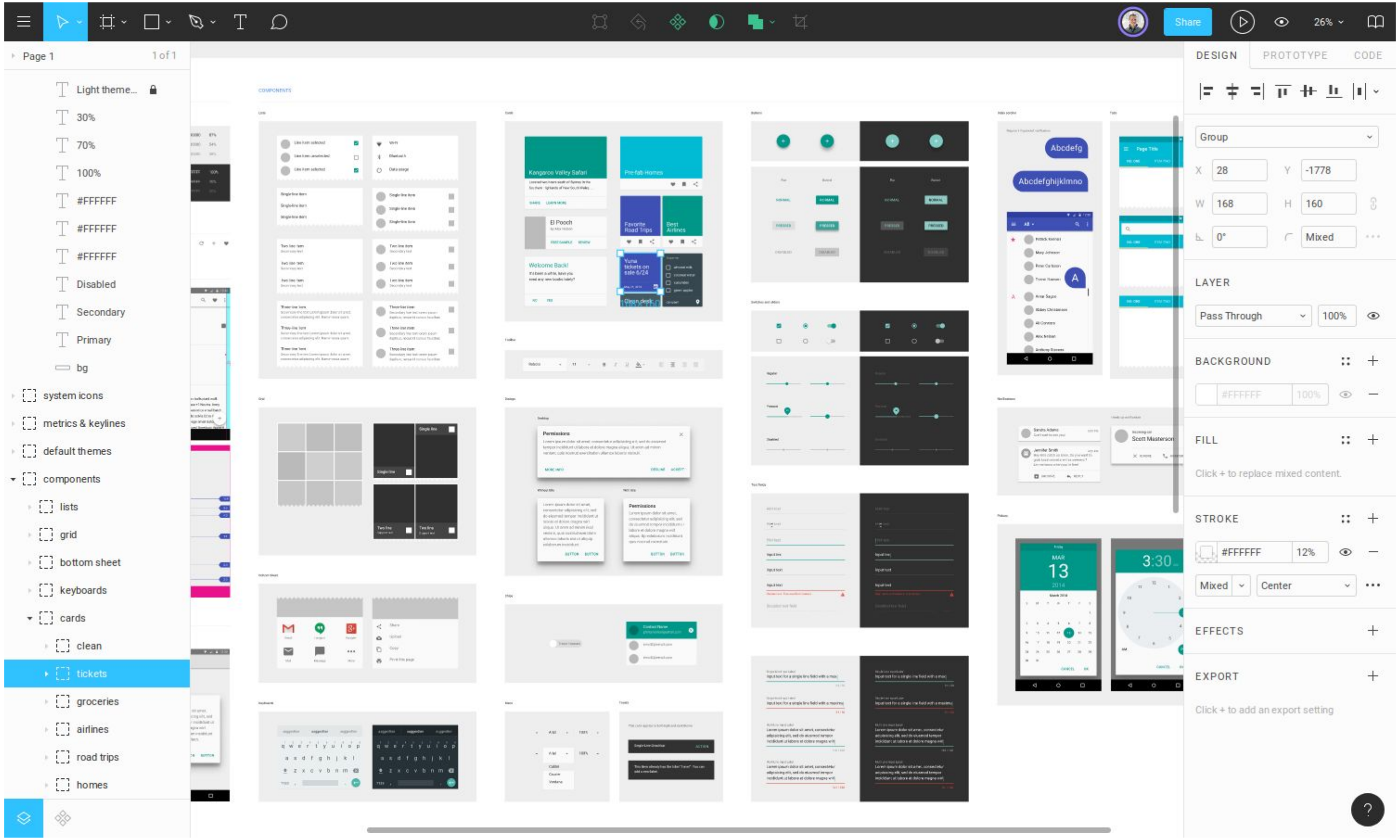
Figma | Computational Complexity





- Interface design tool
- Computationally complex
  - Application loading
  - Graphical rendering
- Users may load many large designs
- Written in C++

Lin Clark **CC BY-SA**



X 28, Y -1778, W 168, H 160, 0°, Mixed

Layer: Pass Through, 100%

Background: #FFFFFF, 100%

Stroke: #FFFFFF, 12%, Mixed, Center

Figma's load time is 3x faster using WASM over JS

React | Virtual DOM

- Computationally complex
- React 16+ API is ideally concurrent
- WebAssembly multithreading (WIP!)
- JavaScript used to update DOM, WebAssembly included to speed things up
- Probably written in Rust

~~Asynchronous I/O~~

- ~~Threads allow for multiple requests in parallel~~
- ~~May use backend languages in frontend~~

Use JavaScript

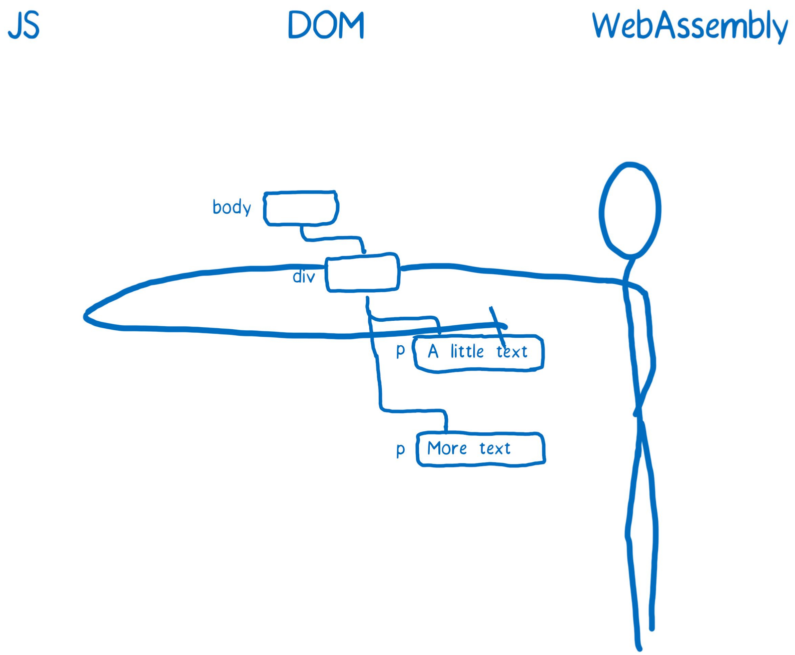
Caveats

When isn't WebAssembly useful?

Sharing is *hard*...

- Numbers are natively supported
- More complex data types require more work
- WebAssembly heap stored as JavaScript ArrayBuffer

Slow Invocation



- ▷ JavaScript functions can be called from WASM and vice-versa
- ▷ WebAssembly functions can't be called by the browser's JIT runtime directly and require an intermediate step
- ▷ Firefox eliminated extra step, but other browsers still require as much as 100x as long to invoke WASM functions
- ▷ DOM manipulation still has to be from JavaScript, with long invocation times

Technical Demo

How does it actually work?

Observations about WASM

(+) Easy to interface with web tech

(+) Makes native code portable

(+) Quick for heavy computations

(-) Difficult to set up build tools (WSL)

(-) Requires low-level awareness

Conclusion

What have we learned?

WebAssembly is...

- An **efficient binary format**
- Great at making native code **portable**
- Incredibly **fast**
- Relatively **young**

Thanks!

Any questions?

Irfaan Khalid https://irfaan.me

Matthew Leon https://mleon.dev