

Improving web app with Rust and WebAssembly

ILYA BARYSHNIKOV

2019.04.20

ABOUT ME

- I work in Align Technology
- previously, JavaScript developer
- I use Rust and WebAssembly at work

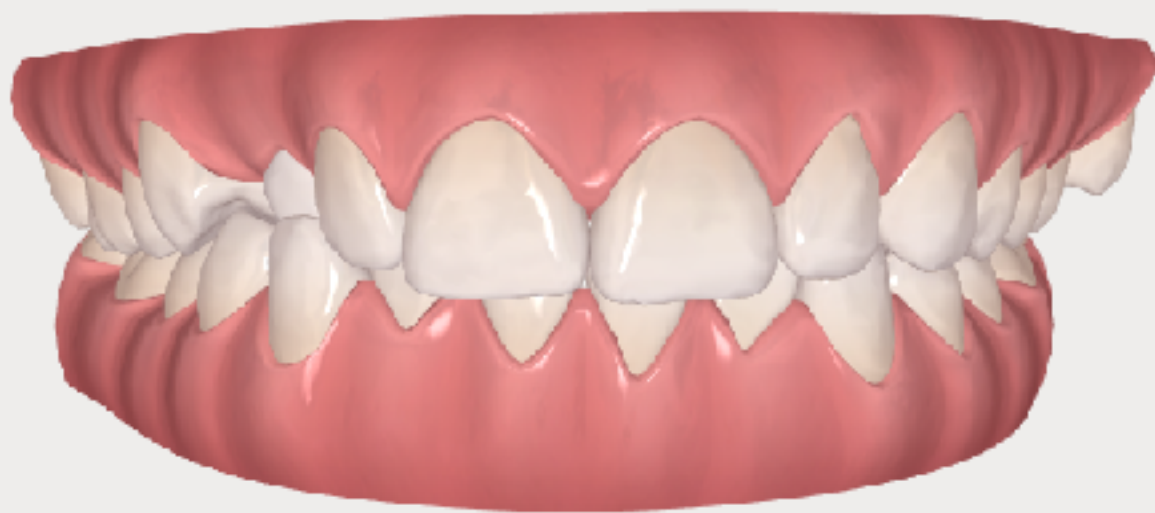


WE USE RUST AND WEBASSEMBLY



DON'T NOT BREAK





LIGHTS

```
1 use wasm_bindgen::prelude::*;
2
3 #[wasm_bindgen(js_name = doMath)]
4 pub fn do_math(
5     indices: Vec<u32>,
6     vertices: Vec<f32>,
7     uv: Vec<f32>,
8 ) -> Vec<f32> {
9     let normals = get_results(&indices, &vertices, &uv);
10    normals
11 }
```

```
46  /**
47   * @param {Uint32Array} indices
48   * @param {Float32Array} vertices
49   * @param {Float32Array} uv
50   * @returns {Float32Array}
51  */
52  export function doMath(indices, vertices, uv) {
53      const ptr0 = passArray32ToWasm(indices);
54      const len0 = WASM_VECTOR_LEN;
55      const ptr1 = passArrayF32ToWasm(vertices);
56      const len1 = WASM_VECTOR_LEN;
57      const ptr2 = passArrayF32ToWasm(uv);
58      const len2 = WASM_VECTOR_LEN;
59      const retptr = globalArgumentPtr();
60      wasm.doMath(retptr, ptr0, len0, ptr1, len1, ptr2, len2);
61      const mem = getUint32Memory();
62      const rustptr = mem[retptr / 4];
63      const rustlen = mem[retptr / 4 + 1];
64
65      const realRet = getArrayF32FromWasm(rustptr, rustlen).slice();
66      wasm.__wbindgen_free(rustptr, rustlen * 4);
67      return realRet;
68  }
```

```
1 use wasm_bindgen::prelude::*;
```

```
2  
3 #[wasm_bindgen]
```

```
4 extern "C" {
```

```
5     pub type Callback;
```

```
6  
7     #[wasm_bindgen(method, structural, js_name = loadResults)]
```

```
8     pub fn load_results(  
9         this: &Callback,  
10         normals: Vec<f32>,  
11         binormals: Vec<f32>,  
12     );  
13 }  
14  
15 #[wasm_bindgen(js_name = doMath)]  
16 pub fn do_math(  
17     indices: Vec<u32>,  
18     vertices: Vec<f32>,  
19     uv: Vec<f32>,  
20     callback: Callback,  
21 ) {  
22     let (normals, binormals) = get_results(&indices, &vertices, &uv);  
23     callback.load_results(normals, binormals);  
24 }
```

```
23 export function __wbg_loadResults_b18649d3fb51362a(arg0, arg1, arg2, arg3, arg4) {  
24     let varg1 = getArrayF32FromWasm(arg1, arg2);  
25  
26     varg1 = varg1.slice();  
27     wasm.__wbindgen_free(arg1, arg2 * 4);  
28  
29     let varg3 = getArrayF32FromWasm(arg3, arg4);  
30  
31     varg3 = varg3.slice();  
32     wasm.__wbindgen_free(arg3, arg4 * 4);  
33  
34     getObject(arg0).loadResults(varg1, varg3);  
35 }
```

DATA DESCRIPTION

In: 3 arrays, 60 000 total length

Out: 2 arrays, 30 000 total length

WHAT I MEASURED

JS: computation

WASM: computation + data transfer
overhead

WHAT I MEASURED

JS: computation

WASM: computation + data transfer overhead

Looks unfair...

WHAT I MEASURED

JS: computation

WASM: computation + data transfer overhead

Looks unfair... but still faster!

PERFORMANCE GAINS

x2

development build

x1.3

production build

PERFORMANCE GAINS

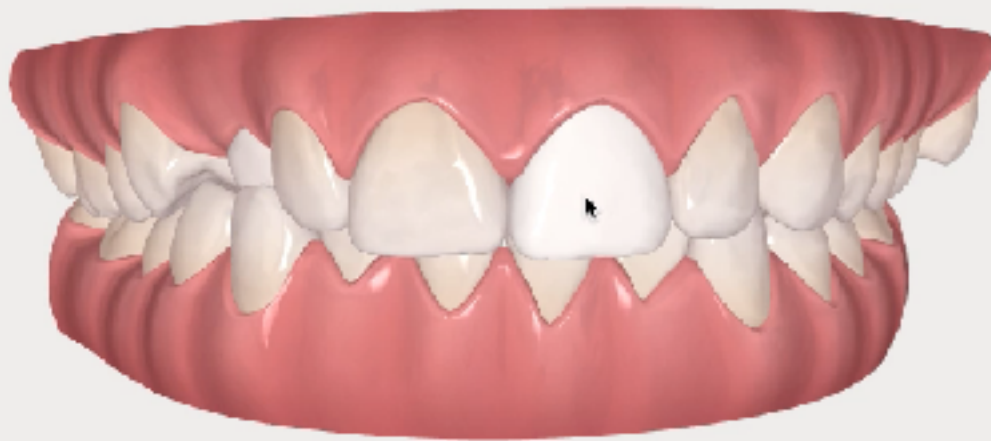
- Most desktop browsers are around x1.3
- Safari on ipad is x0.8
- Edge is x6

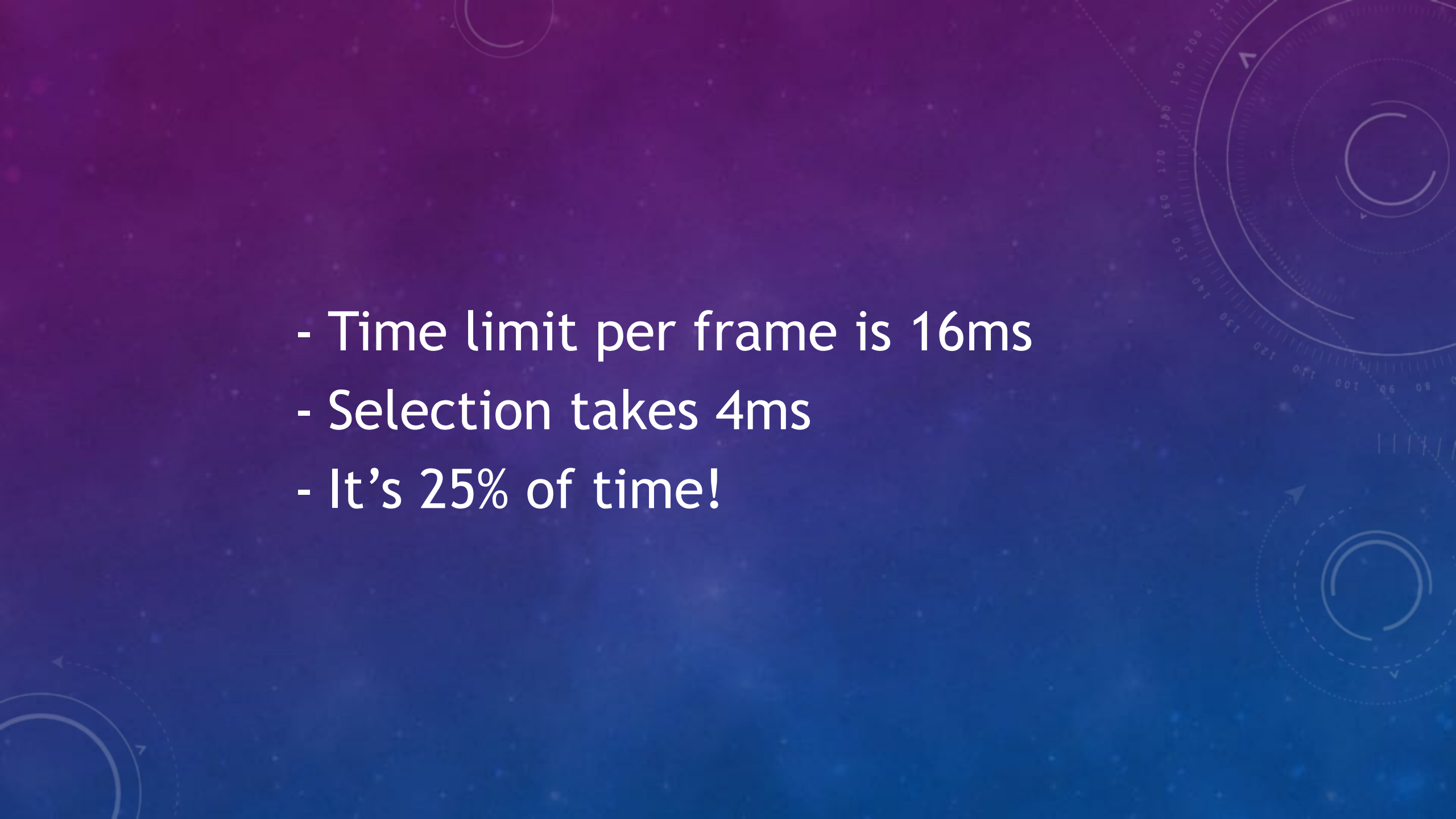
HOW TO IMPROVE

- Keep state on the wasm side
- Create JS typed arrays directly from WebAssembly memory

But it's a small case,
it takes only 2ms

SELECTING OBJECTS



- 
- Time limit per frame is 16ms
 - Selection takes 4ms
 - It's 25% of time!

REAL IMPACT

- Better FPS in some cases
- Less GC
- Deliver more features!

DOWNSIDERS

- Higher complexity
- Browser support



wasm2js

```
1  let wasm;
2
3  import('pkg/math_module.js').then(result => {
4    wasm = result;
5  });
6
7  import { doMathInJs } from './fallback.js';
8
9  export function doMath(indices, vertices, uv) {
10    if (wasm) {
11      let normals;
12      let binormals;
13
14      const callback = {
15        loadResults(normalsOut, binormalsOut) {
16          normals = normalsOut;
17          binormals = binormalsOut;
18        }
19      };
20
21      wasm.doMath(indices, vertices, uv, callback);
22
23      return { normals, binormals };
24    } else {
25      return doMathInJs(indices, vertices, uv);
26    }
27  }
```



wasm-pack

QUALITY CONTROL

- wasm-pack test

QUALITY CONTROL

- wasm-pack test
- cargo test

QUALITY CONTROL

- wasm-pack test
- cargo test
- cargo2junit

QUALITY CONTROL

- wasm-pack test
- cargo test
- cargo2junit
- cargo-audit

QUALITY CONTROL

- wasm-pack test
- cargo test
- cargo2junit
- cargo-audit
- headless_chrome

FUTURE

- better wasm2js
- gloo - modular web toolkit
- wasm-bindgen and stdweb compatibility, initial support released a few days ago!
- Web IDL Bindings proposal
- WASI (WebAssembly System Interface)

Thanks

!

RUSTCON ASIA 2019

Links:

