Web Assembly

Sam Sartor
February 14, 2018
Mines Linux Users Group
Client Code
What is a website?

A website is really just a fancy document, temporarily downloaded from some remote computer system.

Indeed, Sir Tim Berners-Lee created the world-wide-web as a way of sharing research papers.

These documents contain text, images, and styling information.
Interactive Websites

Is it possible to make websites interactive?

We would have to include some instructions along with the document. We need some kind of scripting language for webpages.
JavaScript
The year is 1995. Netscape Communications Corporation is dying. In a frantic attempt to one-up Microsoft, the company decides to embed a scripting language into the Netscape browser.

They give Brendan Eich 10 days to make a prototype. Eich dreams of Scheme. He wants his new language to be elegant, fast, and pure. But it is too late. The lawyers at Netscape have made a deal with Sun Microsystems.

It will be known as JavaScript.
This page uses 800 thousand lines of JavaScript
Emscripten
We need C

- People start writing games in JavaScript
- Why can’t we use the Unity game engine?
- It’s written in C
- Browsers don’t run C
- Browsers only run JavaScript
- What if we compiled C to JavaScript?
Introducing...

It compiles things to JavaScript!
C

```c
int factorial(int n) {
    if (n == 0)
        return 1;
    else
        return n *
            factorial(n-1);
}
```

asm.js

```asm
function __Z9factoriali($0) {
    $0 = $0|0;
    var $1 = 0, $10 = 0, $2 = 0, $3 = 0, $4 = 0, $5 = 0, $6
    = 0, $7 = 0, $8 = 0, $9 = 0, label = 0, sp = 0;
    sp = STACKTOP;
    STACKTOP = STACKTOP + 16|0; if ((STACKTOP|0) >=
        (STACK_MAX|0)) abortStackOverflow(16|0);
    $2 = $0;
    $3 = $2;
    $4 = ($3|0)==(0);
    if ($4) {
        $1 = 1;
    } else {
        $5 = $2;
        $6 = $2;
        $7 = (($6) - 1)|0;
        $8 = (__Z9factoriali($7)|0);
        $9 = Math_imul($5, $8)|0;
        $1 = $9;
    }
    $10 = $1;
    STACKTOP = sp;return ($10|0);
}
```
That was a Bad Idea

JavaScript was designed for...
✓ Crazy people
○ Humans
✗ Computers

Why compile low-level → high-level?

Why don’t we have *machine code for the web*?
Web Assembly
Which Direction?

emscripten can compile your code to Web Assembly!
### WASM Example

<table>
<thead>
<tr>
<th>C</th>
<th>WASM Text</th>
<th>WASM Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>int factorial(int n) {</td>
<td>get_local 0</td>
<td>20 00</td>
</tr>
<tr>
<td>if (n == 0)</td>
<td>i64.eqz</td>
<td>50</td>
</tr>
<tr>
<td>return 1;</td>
<td>if i64</td>
<td>04 7E</td>
</tr>
<tr>
<td>else</td>
<td>i64.const 1</td>
<td>42 01</td>
</tr>
<tr>
<td>return n *</td>
<td>else</td>
<td>05</td>
</tr>
<tr>
<td>factorial(n-1);</td>
<td>get_local 0</td>
<td>20 00</td>
</tr>
<tr>
<td>}</td>
<td>get_local 0</td>
<td>20 00</td>
</tr>
<tr>
<td></td>
<td>i64.const 1</td>
<td>42 01</td>
</tr>
<tr>
<td></td>
<td>i64.sub</td>
<td>7D</td>
</tr>
<tr>
<td></td>
<td>call 0</td>
<td>10 00</td>
</tr>
<tr>
<td></td>
<td>i64.mul</td>
<td>7E</td>
</tr>
<tr>
<td></td>
<td>end</td>
<td>0B</td>
</tr>
</tbody>
</table>
Practical WASM
C/C++
We can use the emscripten sdk to compile C/C++ to JavaScript and WASM.

Emscripten can build a test HTML webpage, JavaScript file, or a pure wasm library.

$ emcc demo.c -s WASM=1 -o demo.html
$ emcc demo.c -s WASM=1 -o demo.js
$ emcc demo.c -s WASM=1 -s SIDE_MODULE=1 -o demo.wasm
Rust
Rust is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.

```rust
fn main() {
    let number = 13;
    let fact = match number {
        1 => "is one",
        2 | 3 | 5 | 7 => "is prime",
        13..19 => "is a teen",
        _ => "ain't special",
    };

    println!("{} {}!", number, fact);
}
```
Rust Toolchains

Rust can be easily compiled to JavaScript and WebAssembly!

You can install a WASM toolchain by running

```
$ rustup toolchain add wasm32-**
```

The available toolchains are:

- `wasm32-unknown-unknown`  compile to pure WASM
- `wasm32-unknown-emscripten`  compile to a WASM executable
- `asmjs-unknown-emscripten`  compile to JavaScript
Rust comes with a build system called **Cargo**.

$ cargo run  
   Finished helloworld [unoptimized + debuginfo] target(s) in 0.0 secs  
   Running `target/debug/helloworld`  
Hello, world!

**Cargo can target WASM!**

$ cargo build --target wasm32-unknown-emscripten  
   Compiling helloworld v0.1.0 (file:///home/sam/Code/helloworld)  
   Finished dev [unoptimized + debuginfo] target(s) in 1.26 sec

$ cd target/wasm32-unknown-emscripten/debug/  
$ node helloworld.js  
Hello, world!
The whole point is to make interactive websites. How can we edit HTML through Rust? How do we make Rust web development feasible?

Introducing...
stdweb provides JavaScript’s abilities within Rust:

```rust
let print_hello = |name: String| {
    println!( "Hello, {}!", name );
};

js! { 
    var print_hello = @{print_hello};
    print_hello("Bob");
}
```

We can connect to buttons and other page elements:

```rust
let button = document().query_selector("#button").unwrap();
button.add_event_listener(move |_: ClickEvent| 
    js! { @{button}.style = "display: none;" } }
);
We can install the cargo-web tool to make Rust WASM even easier!

```bash
$ cargo web start
  Finished release [optimized] target(s) in 0.0 secs
  You can access the web server at `http://127.0.0.1:8000`.
```
parcel
We can use Parcel to bundle JS and WASM into the same project.

In `main.js`:

```javascript
import {add} from './add.rs';
console.log(add(2, 3));
```

In `add.rs`:

```rust
#[no_mangle]
pub fn add(a: i32, b: i32) -> i32 {
    a + b
}
```
Putting it all Together
Conclusion

- Faster websites
- Written in any language
- With desktop libraries
- With native graphics
- With modern tools
Questions?
This presentation was from the **Mines Linux Users Group**. A mostly-complete archive of our presentations can be found online at https://lug.mines.edu.

Individual authors may have certain copyright or licensing restrictions on their presentations. Please be certain to contact the original author to obtain permission to reuse or distribute these slides.