Presentation given at FOSDEM 2016 on January 31st 2016 in Brussels, Belgium.
For a deeper dive in LLVM-WebAssembly-isms, complete with role-playing references, consult our LLVM developer meeting keynote. The FOSDEM presentation has fewer LLVM-ism, and more up-to-date information.

¶ JF narrates:
WebAssembly is a tale of four browser vendors, seeking new languages and capabilities while staying fast, secure and portable. The old JavaScript wizard still has many spells under its belt, but it seeks a companion on its quest to reach VM utopia. WebAssembly is that companion.

Presenters:

- Dan Gohman, Mozilla.
- JF Bastien, Google.

Note to the reader: we use the following markers throughout the speaker notes:

⏩ denotes when we click to animate inside slides.
¶ denotes when we change who's talking.

¶ Dan: Let's meet our hero!

- **Background:**
  - Born from the union of Emscripten/asm.js and PNaCl;
  - Cousin of JavaScript;
  - Godparents Chromium, Mozilla, Edge, WebKit;
  - Designed on Github.
  - Born under W3C Community Group.
- **We're driving towards a Minimal Viable Product:**
  - Release something simple first.
  - Evolve new features, as the Web does.

- **Basic attributes:**
  - **Strength:** leveraging the power of the Web.
  - **Dexterity:** portability.
  - **Constitution:** designed for security-in-depth from the start.
  - **Intelligence:** very low! WebAssembly is a low-level language. It's up to the compiler on the developer's machine to optimize, a WebAssembly VM only does basic things such as instruction selection and register allocation.
  - **Wisdom:** you want to know what code runs on your computer. WebAssembly makes it normal to view-source on its binaries.

    - **Charisma:** it's *of the Web*, the internet loves it.

¶ JF goes for the remaining __attribute__((*))!

- **Others:**
    - **Secure:** Internet-proof. Security is our thing. Learn more on security from JF's CppCon talk.
    - **Initiative:** the announcement seems to have taken folks by surprise (more than we expected!). Either we rolled high or had a large base modifier.
    - **Speed:** near-native. PNaCl and asm.js have shown code could go pretty fast, often on par with native, and we think WebAssembly can surpass either.
    - **Languages:** C/C++ to start with, but we want to be polyglots.
    - **Ideals:** speed, security, portability, ephemerality. Note the last 3 are the Web's ideals, WebAssembly adds speed.
    - **Bonds:** the Web should be able to do everything native code can.

Bring the latest and greatest C++14 to the web, portably and efficiently (see goals). WebAssembly also wants to adventure in the non-Web parts of the world. It should also work on servers and tiny IoT devices.

¶ JF con't.

Let's dive into the details.

¶ Dan: Say I'm a real experienced C++ programmer. "I know how these things go, you take away my pointers!"

¶ JF takes action.
What's the outlook?

- "linear memory" is virtual memory. Cache locality.
- Pointers for serious.
- Function pointers too, in a way.
- Even pointer to member functions if you're into that type of thing.
- vtables.
- APIs on the Web, based on SDK + helper libraries that "just work":
    - OpenGL → WebGL (GLES3 will soon be in Firefox and Chrome).
    - SDL → WebAudio.
    - C++/POSIX filesystems: whatever the web has… It's still improving in that area.
    - Networking: WebSocket / WebRTC.

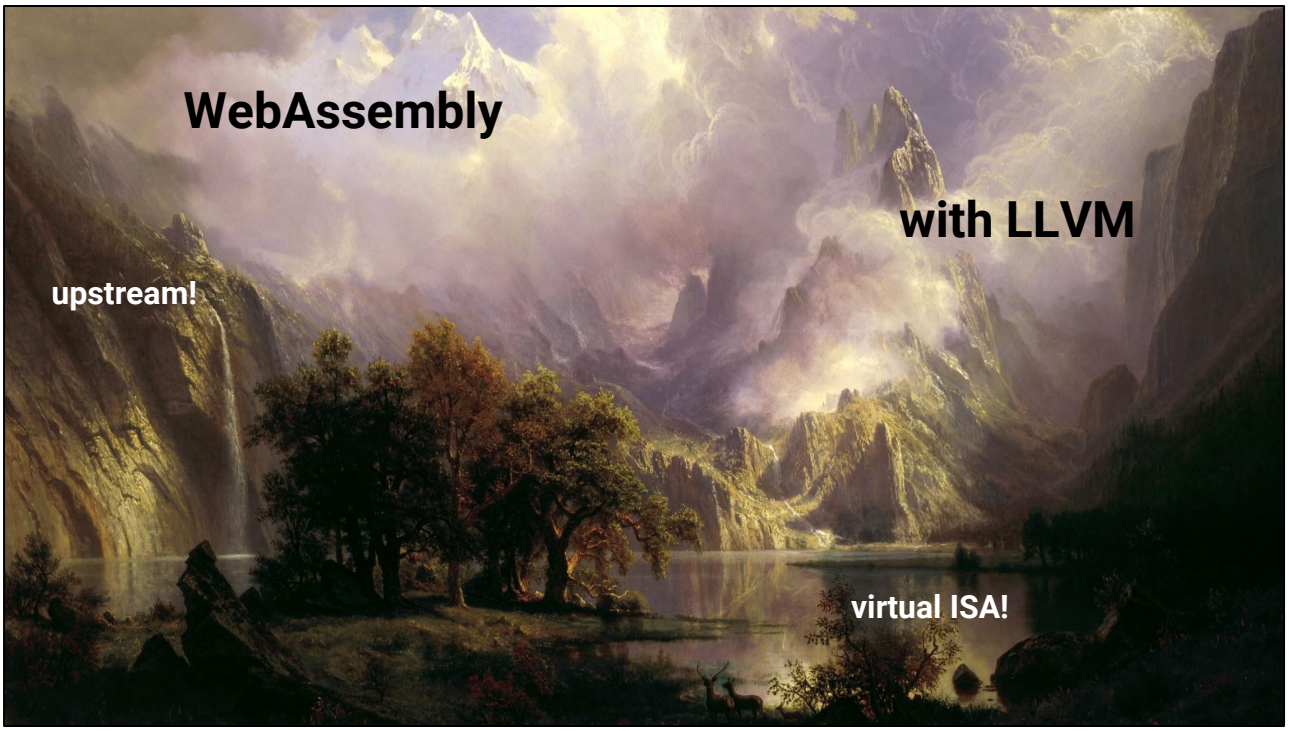
¶ Dan continues:

Look forward to predictable performance.

Expected to quickly add the following:

- Threads/atomics/shared memory.
- SIMD.
- Zero-cost EH.
- Dynamic linking.

This will round out a fairly complete C++ platform.



But how do you do this?

Learned from Emscripten / PNaCl mistakes.
WebAssembly is developed straight in upstream LLVM under `lib/Target/WebAssembly`

¶ JF jumps in!

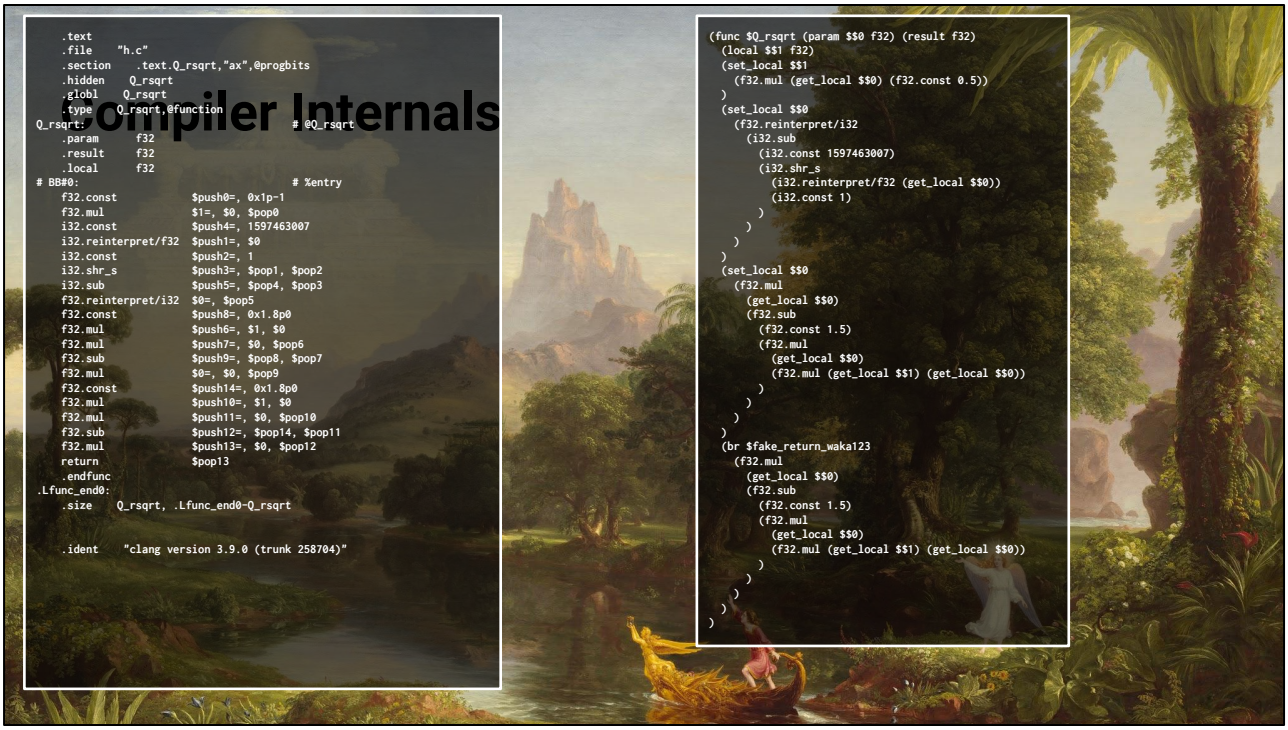
Virtual ISA: what's that? We talk about it a lot like it a big mythical beast.

- Compile on the developer's machine, then lower to different ISAs on the user's machine. There's a second compiler that runs once we know what the architecture is. In browsers it'll share code with the JavaScript engine. There are 10 implementations of this in progress, 3 of the prototypes use LLVM!
- Browser as an OS (or in general "embedder": WebAssembly isn't just a browser thing).
- We must make certain assumptions about the target for code layout and performance:
  - 8-bit bytes.
  - Two's complement integers.
  - IEEE 754-2008 32-bit and 64-bit floating point.
  - Developers choose either 32-bit or 64-bit pointers.
  - Little-endian machine.
  - ...and more.

Undefined behavior, implementation defined, unspecified, etc. WHAT DOES IT MEAN? C++ developers all know of UB, they're afraid that it'll snatch their children in the night and that demons will fly out of their nose.

- Clang and LLVM already eliminate a lot of the leeway C++ gives compilers.
- Progressive refining of undefined behavior.
- WebAssembly nails most of the rest down.
    - Dividing by zero traps.
    - Out of bounds accesses trap.
    - A malicious program can't harm the user directly (developers should still be careful with the data their WebAssembly modules contains).
- Some still remains, for example threads can still race but offer limited local nondeterminism.

Key takeaway: Developers can really think of this as an ISA.



Expression trees allow us to reassociate code and use recent values.
We go even further, expressing control flow as ASTs instead of CFGs. This moves some of the restructuring to developer's machine instead of being the user's device.
We're also thinking that making `statement == expression` might work out well for our goals.
Why? Encoding size.
▸▸
Look at those beautiful s-expressions, we could just feed them to a LISP and `eval` everything and C++ would just execute out of that environment and...

¶ Dan jumps in.

The JF appears to be confused. ▸▸
The output looks like assembly!
Syntax is just syntax, these are interchangeable.

We get to design our own instruction set :-)

Instruction selection and register allocation are simple because we have a simple virtual ISA:

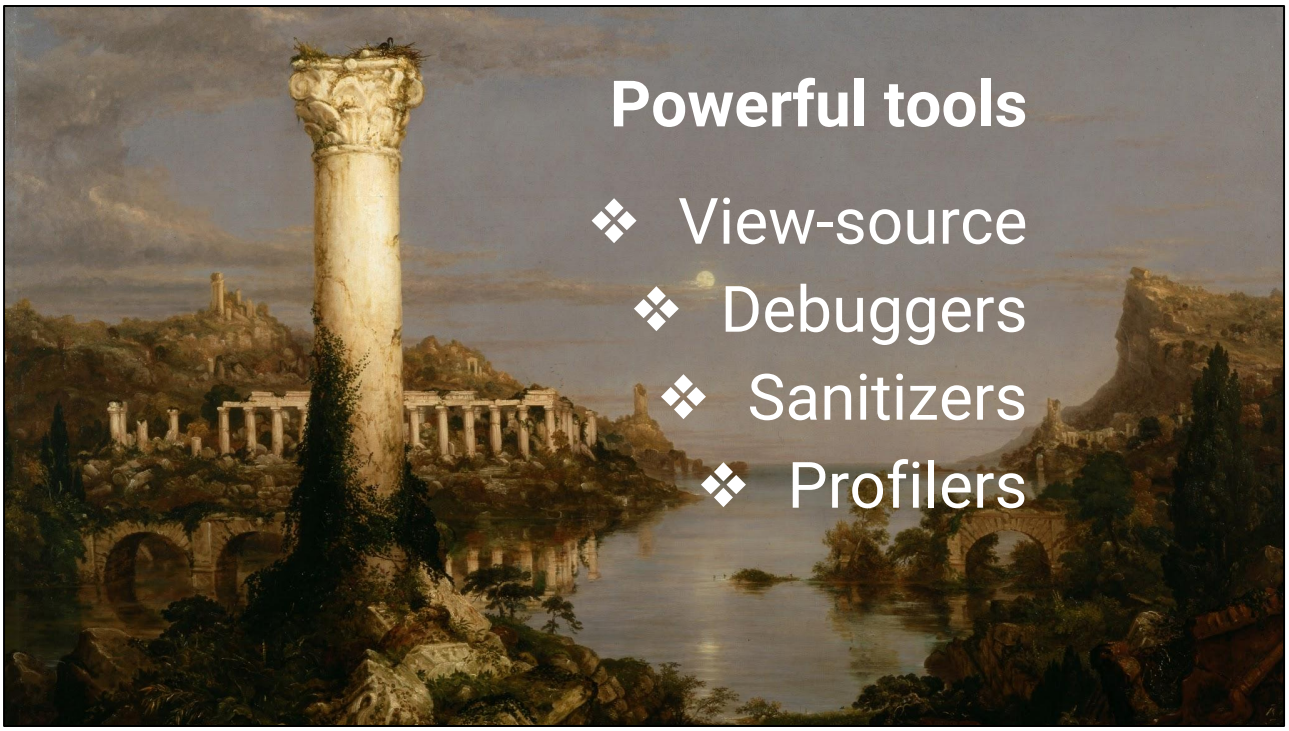
- The WebAssembly instructions are very similar to a compiler IR.
- WebAssembly has an infinite register set, we can color them to reduce interference.

¶ **JF** Of course, most developers don't want to write assembly by hand.

¶ **Dan** That's what frontends are for.
Using clang's out of the box feature set for most things.
Get rid of C++ overhead which other targets are stuck with, e.g. create C++ ABI v2.
For example, some tweaks to the standard C++ name mangling algorithm to greatly reduce the length of common symbol names.

¶ **JF stumbles onto the battlefield.**

Other static languages are also a great target for WebAssembly.
This is a battleground for optimizations.



We're developing LLVM as the first WebAssembly C/C++ compiler. Not to be distasteful in the LLVM dev room but we hope other compilers play along as well.

▸▸ Can do cool tricks with powerful tools, the first we'll build is a toolchain.
Choose your own adventure; we'll build a toolchain but anyone else can build their own.
Remember: everything is open source and built in the open.

¶ Dan tools in. /rimshot

We want a great C++ development story, and that means we're planning for all the tools that modern C++ developers depend on:

▸▸ View source!
▸▸ Debuggers, integrated in the browser and outside of the browser.
▸▸ Sanitizers.
▸▸ Profilers.

Reiterate the “soon after” features from slide 3:

- Threads/atomics/shared memory.
- SIMD.
- Zero-cost EH.
- Dynamic linking.

Dan challenges us: this is a good platform but what’s going to take it to the next level? Do we know what’s out there?

⏩ There are many cool things we’re hoping to enable:

- Rust
- Go
- C#
- Python
- And anything else you fancy!

¶ JF peers into the future with his crystal ball. We’ve got this!

More features, more levelups! How are we going to do these other languages?

- ⏩ JIT-compilation: WebAssembly is a new ISA that compilers would need to target.
- ⏩ GC
- Finer-grained memory control.
- Asynchronous signals.

- ⏩ Many things are possible!

Our adventurer is now back at the tavern, enjoying a good brew… What's the next quest?

All the browsers have started implementing WebAssembly, and they're all open source. Check it out!

We have many other adventures ahead of us.
¶ JF and Dan: Will you join us?