C++ EVERYWHERE WITH

WEBASSEMBLY

GITHUB POWERED, WEBASSEMBLY READY, C++ DEPENDENCIES & UPGRADE MANAGER

Damien Buhl (alias daminetreg) at nxxm.github.io

WEBASSEMBLY PRIMER

WEBASSEMBLY

- Binary instruction format
- Stack based "vm"
- Enables C++ web browser & server apps

W3C OPEN STANDARD

AVAILABLE EVERYWHERE IN V1.0

PORTABLE BINARY INSTRUCTIONS

- Small files
- Fast decoding
- Reduced memory usage
- The syntheis & evolution of asm.js & PNaCl

WASM MODULE

- PE or Elf for the Web.

WASM MODULE STRUCTURE

WASM MODULE STRUCTURE

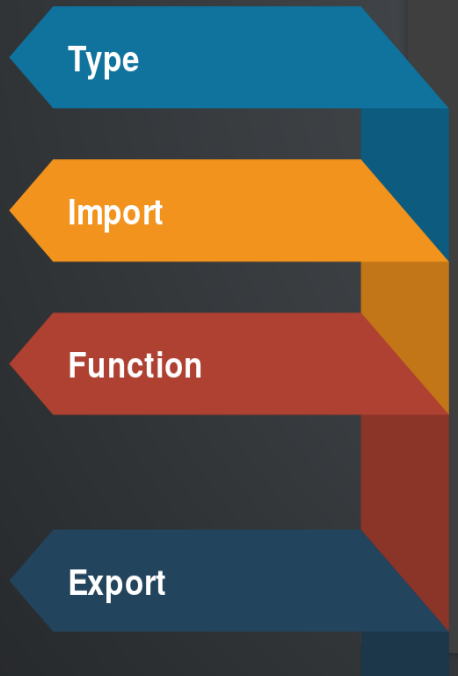
```
int main() {
  return 43;
}
```

WASM MODULE STRUCTURE

~158 bytes

```
00 61 73 6d 01 00 00 00  01 08 02 60 00 00 60 00  |.asm.......`..`.|
01 7f 03 03 02 00 01 04  05 01 70 01 01 01 05 03  |..........p.....|
01 00 02 06 15 03 7f 01  41 80 88 04 0b 7f 00 41  |........A......A|
80 88 04 0b 7f 00 41 80  08 0b 07 2d 04 05 5f 6d  |......A....-.._m|
61 69 6e 00 01 06 6d 65  6d 6f 72 79 02 00 0b 5f  |ain...memory..._|
5f 68 65 61 70 5f 62 61  73 65 03 01 0a 5f 5f 64  |_heap_base..__d|
61 74 61 5f 65 6e 64 03  02 0a 09 02 02 00 0b 04  |ata_end.........|
00 41 2b 0b 00 28 04 6e  61 6d 65 01 1a 02 00 11  |.A+..(.name.....|
5f 5f 77 61 73 6d 5f 63  61 6c 6c 5f 63 74 6f 72  |__wasm_call_ctor|
73 01 04 6d 61 69 6e 02  05 02 00 00 01 00        |s..main.......  |
```

WASM MODULE STRUCTURE




```
(module

  (type (;2;) (func (result i32)))


  (import "env" "getTotalMemory" (func (;1;) (type 2)))


  (func (;21;) (type 2) (result i32)
    i32.const 43
    return)

  (export "_main" (func 21))


  (data (i32.const 1024) "\05\c0\de\00\be\e0 ... ")
```

PORTABLE YET NATIVE

.wasm

Download

```
1  main: # @main
2    pushq %rbp
3    movq %rsp, %rbp
4    movl $43, %eax
5    movl $0, -4(%rbp)
6    popq %rbp
7    retq
```

x86_64

WASM RUNTIME

WASM RUNTIME

- Native speed, no emulation
- C++ Heap & Stack is allocated in a contiguous memory segment
- No Garbage Collection: Memory management as usual

WASM RUNTIME: NO GARBAGE COLLECTION

module import

js_getElementById(...)

Call into Browser API

WebIDL DOM API

WASM RUNTIME: WASM 2.0 ?

module import

Direct Browser API binding

WASM RUNTIME

CppCon 2014: Chad Austin "Embind and Emscripten: Blending C++11, JavaScript, and the Web Browser"

WASM RUNTIME: MULTITHREADING

- pthreads has been implemented and works
- but is based on SharedArrayBuffer...

WEB DEVELOPMENT

THE JAVASCRIPT RISE

- Javascript is a dynamically typed language
- C++ is a statically typed language
- Two irreconcilable worlds ?

JAVASCRIPT & C++

- They share one key philosophy...
- Openness, Flexibility & Extensibility

Neither Bjarne Stroustrup nor Brendan Eich wanted their language to be limited to what they could imagine.

```
Object.prototype === template <class T>
```

TYPESCRIPT

Key selling point: Javascript that scales.

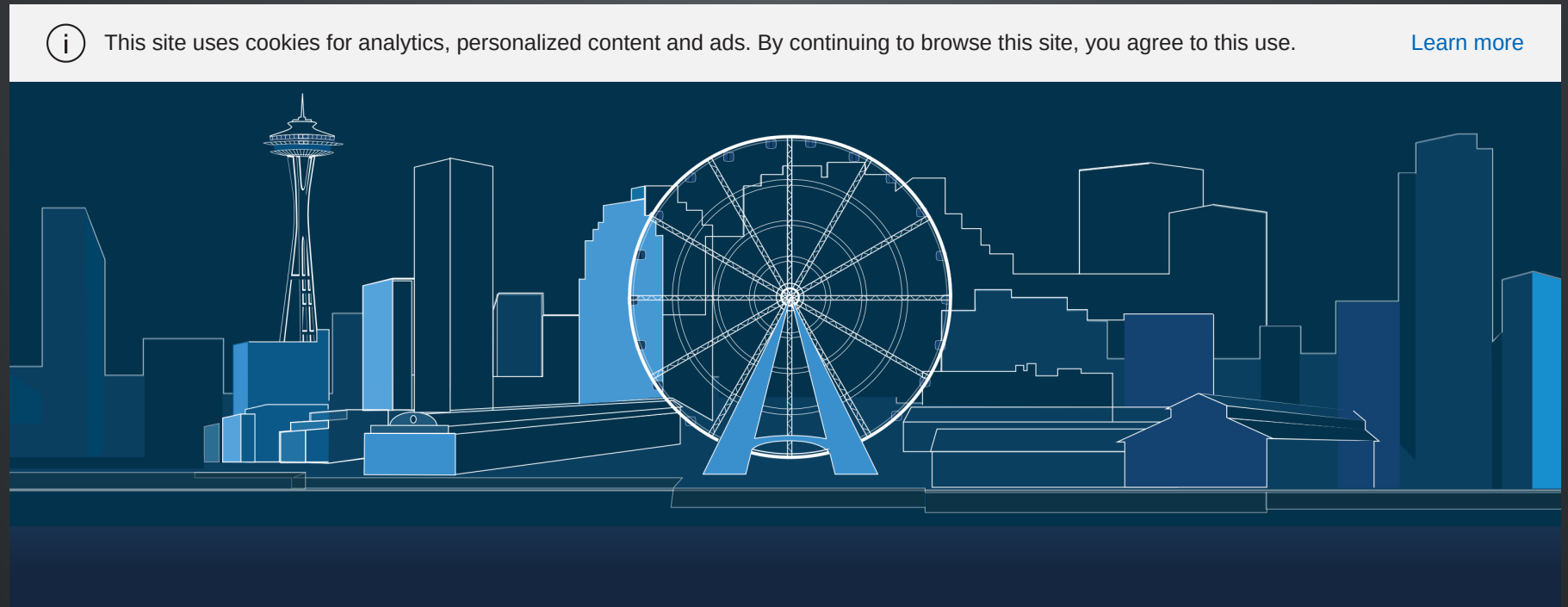
Types enable JavaScript developers to use [...] practices like static checking and code refactoring [...].

https://www.typescriptlang.org/

WAIT TYPES ? STATIC CHECKING ?

- Isn't C++ all about this ?
- Could we use it to make Web Apps that scales ?

CAN WE MAKE SOME ROOM FOR C++ ?

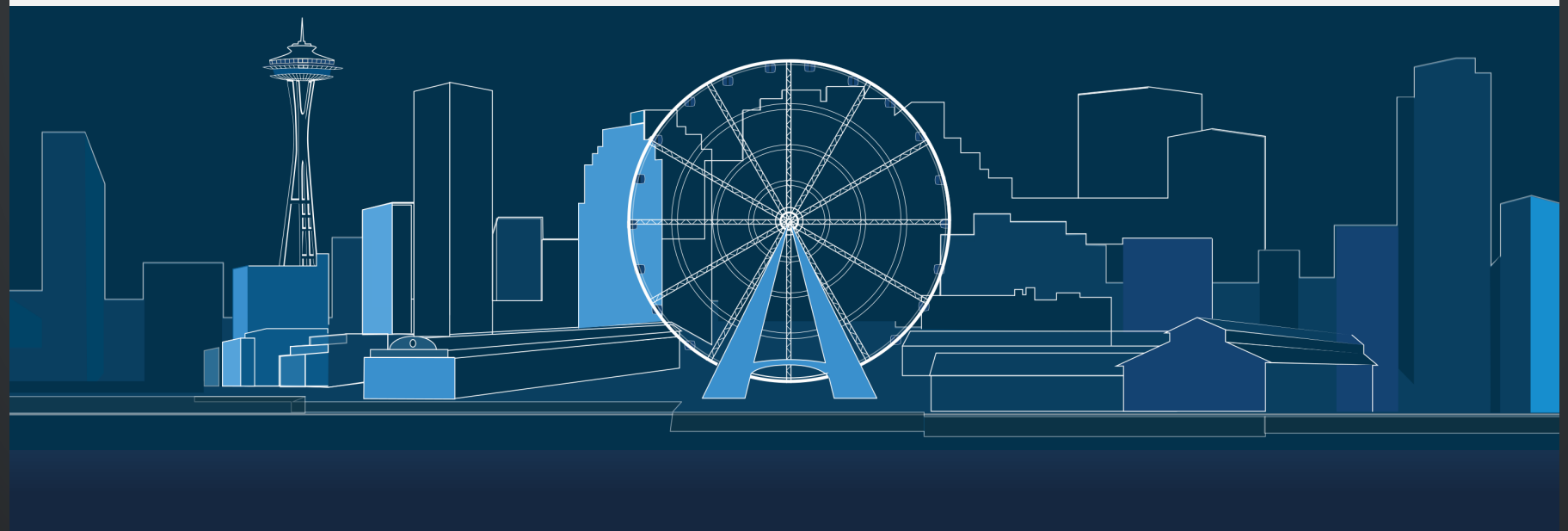


CAN WE MAKE SOME ROOM FOR C++ ?

```
<script type="text/c++">

  #include <belle/vue/dom.hxx>
  #include <belle/vue/fx.hxx>

  int main() {

    using namespace belle::vue;

    auto banner_ts = get_element_by_id("banner_ts");

    auto banner_cpp = get_element_by_id("banner_cpp");

    fx::fade_out(banner_ts, [=]() mutable {

      banner_ts.setAttribute("style", "display:none;");
      banner_cpp.setAttribute("style", "display:visible;");

      fx::fade_in(banner_cpp);

    });


    return 0;
  }
</script>
```

MAKE SOME ROOM FOR C++

This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use. Learn more



WEBSERVICES WITH WEBASSEMBLY

GETTING SOME REST

- CRUD operations on URI addressed entities

REST IN JAVASCRIPT

```
app.get('/user/:id', function(req, res) {
  res.send('user ' + req.params.id);
});


app.listen(80);
```

Does anybody care about the :id type ?

Whatever is valid : /user/43 or /user/banana

REST ARGUMENT PARSER IN C++

```cpp
namespace qi = boost::spirit::qi;
namespace ct = boost::callable_traits;
using namespace boost::mp11;

template<class F>
void GET(const std::string& route, F&& functor) {
  mp_remove<ct::args_t<F>, session&&> args_parsed;

  auto fail = false;

  auto raw = splitted_args.begin();

  tuple_for_each(args_parsed, [&](auto& result) {
    if (fail) return;

    fail = !qi::parse(raw->begin(), raw->end(), qi::auto_, result);

    ++raw;
  });

  if (!fail) {
    std::apply(functor, std::tuple_cat(std::make_tuple(session {}), args_parsed));
  }

}
```

LET'S SEE THE FOLLOWING IN ACTION

```
app.GET("/add", [&](bete::session&& session,
    std::string name, size_t min, size_t max, size_t avg) {

  datamodel->benchmarks->push_back({name, min, max, avg});

  session.response.write("Added : "s + name

      + "with min="s + std::to_string(min)
      + ", max="s + std::to_string(max)
      + ", avg="s + std::to_string(avg) + "."
      );

});

app.listen(80);
```

GET /add/added_via_REST_service/1/2/4

BUT ADDING TO WHAT ?

- To an Observable datatype
- Simulating a bit of Aspect Oriented Programming in C++

BELLE::VUE::OBSERVABLE


```
template<class T>
struct observable {
  struct proxy {

    // ...

    T * proxied;

    ~proxy () {
      for (auto& obs : observers_) {
        obs(*proxied);
      }

    }

  };

  proxy operator -> () {
     return proxy (observers_, std::addressof(proxied));
  }

  T* operator -> () const {
     return &proxied;
  }

  // ...
};
```

LET'S DEFINE AN OBSERVABLE DATAMODEL

```cpp
struct benchmark {
  std::string name{};
  size_t min{};
  size_t max{};
  size_t avg{};
};

struct datamodel_t {
  std::vector<benchmark> benchmarks;
};

BELLE_VUE_OBSERVABLE(benchmark, name, min, max, avg);
BELLE_VUE_OBSERVABLE(datamodel_t, benchmarks);
```

To record WebAssembly function calls benchmarks.

PERFORMANCE MEASUREMENT

PERFORMANCE : CALLING THE BROWSER

```cpp
void test_cpp() {
  auto element = belle::vue::get_element_by_id("insert_here");

  std::string text = fmt::format("Computing the WebAssembly answer {:d}", (43 * i));
  element.innerHTML(text);
}
```

vs

```javascript
function test_js() {
  var element = document.getElementById("insert_here");

  var string_answer = "Computing the Javascript answer " + (43 * i);
  element.innerHTML = string_answer;
}
```

Let's Measure

PERFORMANCE GAINS BY THE WEBASSEMBLY & C++ NATURE

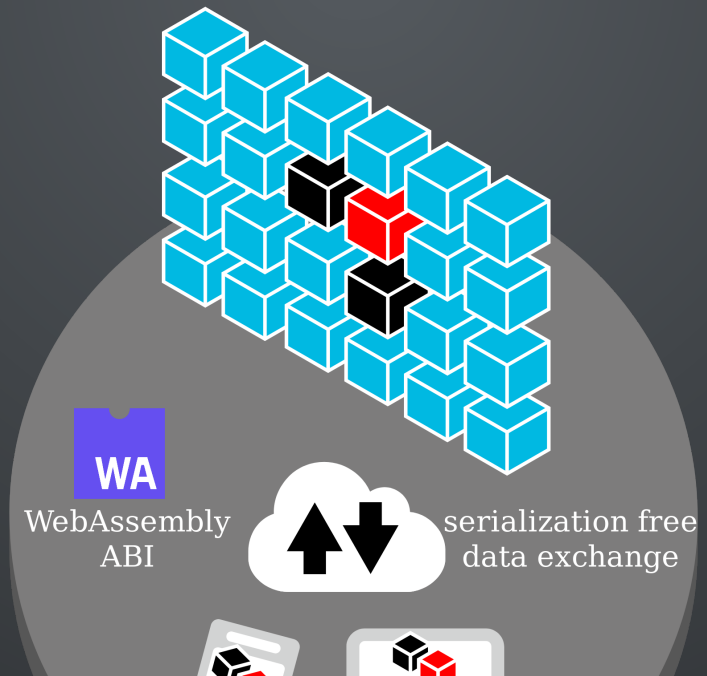
- 8-bit bytes.
- Addressable at a byte memory granularity.
- Little-endian byte ordering.
- IEEE 754-2008 32-bit and 64-bit floating point.

WHAT IF WE ABUSE STD::IS_TRIVIALLY_COPYABLE ?

WHAT IF WE ABUSE STD::IS_TRIVIALLY_COPYABLE?

```cpp
template<class Stream, class T>
inline void write(Stream& stream, const T& value) {
  using decayed_T = typename std::decay_t<T>;

  if constexpr (is_fusion_sequence_v<decayed_T>) {
    for_each_member(value, [&stream](const auto& member_value){
      write(stream, member_value);
    });

  } else if constexpr (is_container_v<decayed_T>) {
    wasmabi::write(stream, value.size());

    std::for_each(value.begin(), value.end(), [&stream](auto& v) {
      wasmabi::write(stream, v);
    });

  } else if constexpr(std::is_trivially_copyable_v<decayed_T>) {
    std::array<char, sizeof(decayed_T)> buf{'\0',};

    const char* addressof_value = reinterpret_cast<const char*>(std::addressof(value));
    std::copy(addressof_value, addressof_value + sizeof(value), buf.data());
    stream.write(buf.data(), buf.size());

  } else {
    static_assert(std::is_same<T, non_implemented_tag>::value, "Unsupported type.");
  }
}
```

FASTER LOAD TIME



THE MOST AWESOME C++17 FEATURE

TRUE HETEROGENEOUS CONTAINERS ARE NOW POSSIBLE

- std::vector<std::variant< ... >> : most of the time the best solution
- But with template variables we can go further!

STATIC TYPING DOESN'T MEAN LESS FLEXIBILITY

```
struct heterogeneous_container
{

  template<class T>
  void push_back(const T& _t) {

    items<T>[this].push_back(_t);
  }

  private:
  template<class T>
  static std::unordered_map<const heterogeneous_container*, std::vector<T>> items;
};

struct user_defined { /* ... */ };

void main() {
  heterogeneous_container c;

  c.push_back(42);
  c.push_back("Hello"s);
  c.push_back(user_defined{});
  c.push_back('A');
  c.push_back(4.00f);
}
```

AN EFFICIENT MECHANISM TO MAP TO JAVASCRIPT DYNAMICITY

- Javascript Array, Dictionaries could all be represented
- Credits to Andy G's : https://bit.ly/2Ik7GsF

JOIN US TO IMPROVE THE WEB

AVAILABLE OPEN SOURCE & COMMERCIAL NOW FOR WINDOWS, MACOS, LINUX & WEBASSEMBLY!