Bridging Languages in ICU4X

How Diplomat brings i18n to the Web and Beyond

Who am I?

- Tyler Knowlton
  - B.S. Computer Science: Game Design
  - Pursuing M.S. in Scientific Computing and Applied Mathematics
- Google Summer of Code Contributor
  - ~500 hours contributing to Diplomat
- Talking to YOU about Diplomat

Sample Problem

Imagine a library with a large scope

Sample Problem

~1.5.0 **icu_collator** NO DEFAULT FEATURES

API for comparing strings according to language-dependent ...

~1.5.0 **icu_collections** NO DEFAULT FEATURES

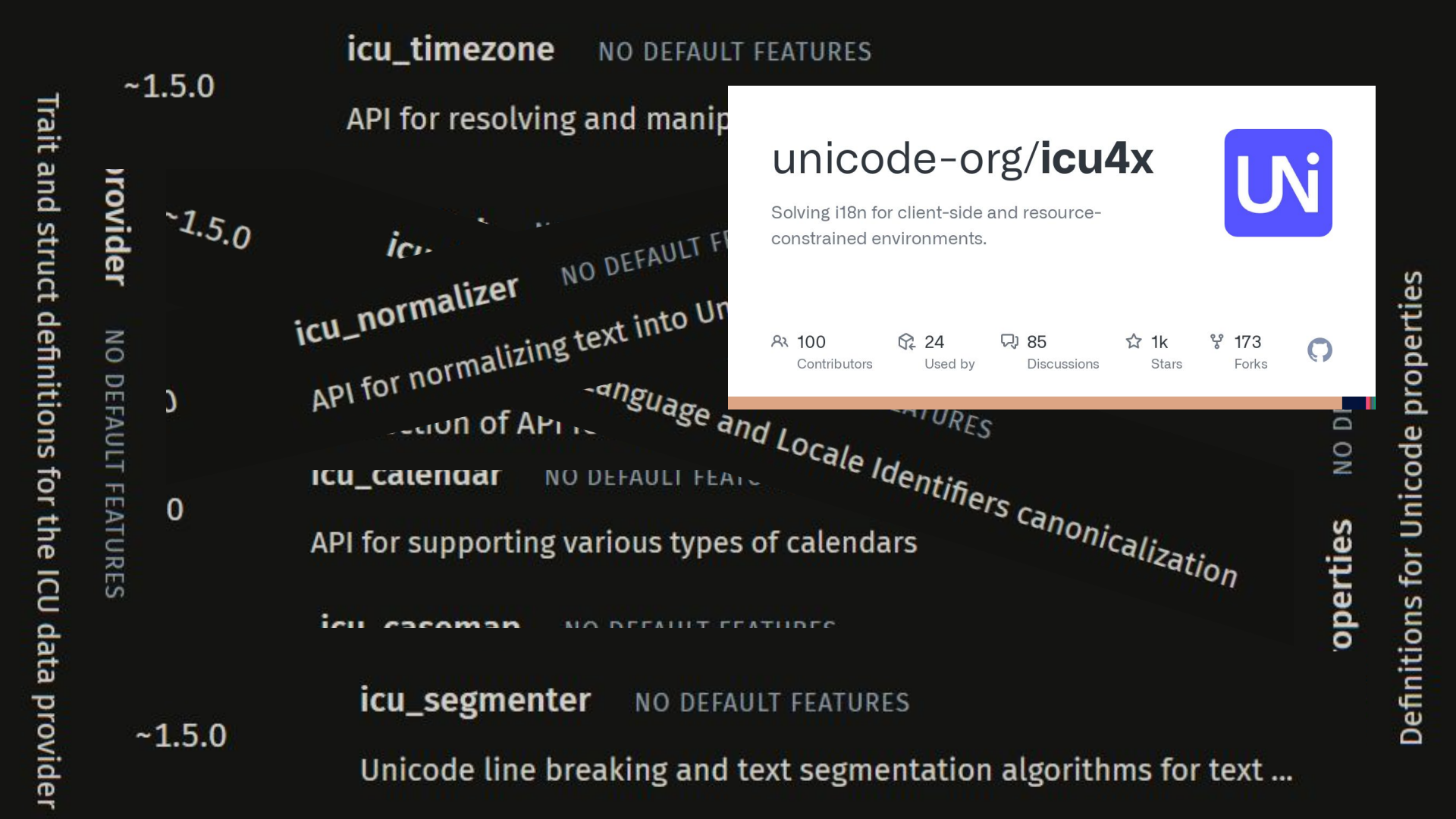
Collection of API for use in ICU libraries.

~1.5.0 **icu_calendar** NO DEFAULT FEATURES

API for supporting various types of calendars

~1.5.0 **icu_casemap** NO DEFAULT FEATURES

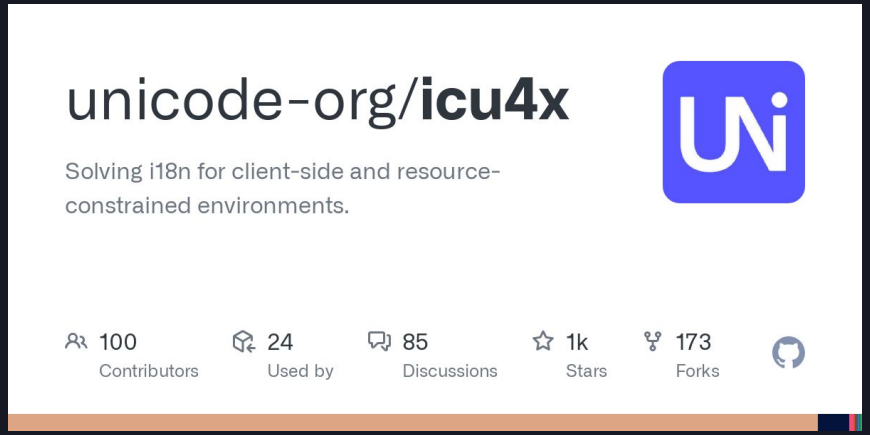
Unicode case mapping and folding algorithms


icu_timezone NO DEFAULT FEATURES
~1.5.0
API for resolving and manip
unicode-org/icu4x
Solving i18n for client-side and resource-constrained environments.
100 Contributors
24 Used by
85 Discussions
1k Stars
173 Forks
icu_normalizer NO DEFAULT F
API for normalizing text into Un
anguage and Locale Identifiers canonicalization
icu_calendar
API for supporting various types of calendars
icu_segmenter NO DEFAULT FEATURES
~1.5.0
Unicode line breaking and text segmentation algorithms for text ...
Trait and struct definitions for the ICU data provider
Definitions for Unicode properties

ICU4X

Library of smaller libraries, all in Rust

Scale creates portability problem



Portability

C (or Rust) provides a simple option: FFI bindings

Another cost: ABI binding generation

```rust
#[no_mangle]
pub extern fn "C" foo() {
    println!("Hello world!");
}
```

```c
#include "<lib/external_def.h>"

int main() {
    foo();
}
```

Challenges with ABIs: Bindings

```
#[no_mangle]
pub extern "C" fn addition(a : i8, b : i8) -> i8 {
    a + b
}
```

```
const fs = require('node:fs');

const wasmRead =
fs.readFileSync("./target/wasm32-unknown-unknown/debug/testlib
.wasm");

function addition(wasmModule, a, b) {
    return wasmModule.addition(a, b);
}

WebAssembly.instantiate(wasmRead).then(
    (wasmModule) => {
        console.log(addition(wasmModule.instance.exports, 1,
10000));
    }
);
```

```
#include <stdint.h>
#include <stdio.h>

extern "C" int8_t addition(int8_t a,
int8_t b);

int main() {
    printf("%i\n", addition(1, 5));
    return 0;
}
```

```
package dev.diplomattest.somelib;
import com.sun.jna.Callback
import com.sun.jna.Library
import com.sun.jna.Native
import com.sun.jna.Structure


internal interface TestLib: Library {
    fun addition(a : Byte, b: Byte) : Byte
}

class TestLibrary {
    companion object {
        internal var libClass : TestLib::class.java
        internal val lib: TestLib = Native.load("testlib", libClass)

        fun add(a : Byte, b: Byte) : Byte {
            return lib.addition(a, b)
        }
    }
}

fun main() {
    println(TestLibrary.add(0, 10))
}
```

```
    @meta.RecordUse()
    @ffi.Native<ffi.Int8 Function(ffi.Int8,
    ffi.Int8)>(isLeaf: true, symbol: 'addition')
    external int _icu4x_FixedDecimal_new_mv1(int a, int
    b);

    final class TestLib implements ffi.Finalizable {
        static int addition(int a, int b) {
            return addition(a, b);
        }
    }

    void main() {
        print(TestLib.addition(0, 10));
    }
```

Challenges with ABIs: ABI Quirks

```rust
#[repr(C)]
pub struct ReturnStruct {
    i : i32,
    j : i32,
    k : i32
}

#[no_mangle]
pub extern "C" fn get_struct() -> ReturnStruct {
    ReturnStruct { i: 42, j: 1, k: 102 }
}
```

```c
struct ReturnStruct {
    int32_t i;
    int32_t j;
    int32_t k;
};

struct ReturnStruct get_struct();
```

Challenges with ABIs: ABI Quirks

```
class ReturnStruct {
    constructor(i, j, k) {
        this.i = i; this.j = j; this.k = k;
    }
}

function getStruct(wasmModule) {
    const structBuffer = new DataView(wasmModule.memory.buffer, 0, 12);

    wasmModule.get_struct(structBuffer);

    let i = structBuffer.getInt32(0, true);
    let j = structBuffer.getInt32(4, true);
    let k = structBuffer.getInt32(8, true);
    return new ReturnStruct(i, j, k);
}
```

Bindgen Tools

Why not use a tool for wrapping? cxx, emscripten, wasm-bindgen, etc.

Each works for one language

Dependencies (and work) stacks with more languages

```
#[cxx::bridge]
mod ffi {
    ...
}
```

```
#[wasm_bindgen]
pub fn some_function(...) {
    ...
}
```

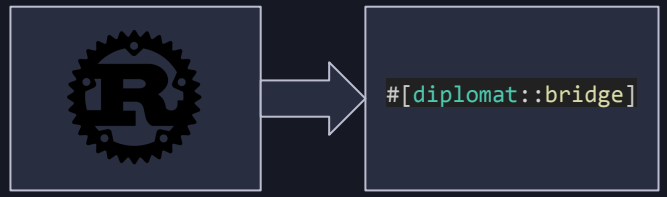
Diplomat!

Translate to many languages

Design goals:

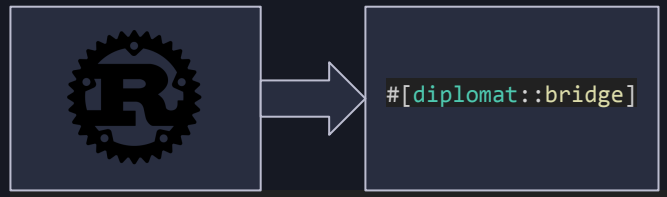
- ONE source of truth
- Extensible by language
- Bindings AND definitions

One Source of Truth: The Bridge



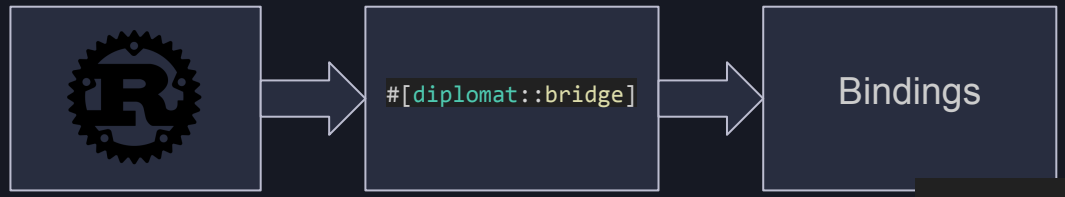

```rust
#[diplomat::bridge]
mod ffi {
    pub struct SomeNamespace;
    impl SomeNamespace {
        pub fn do_a_thing() {
            println!("doing thing");
        }
    }
}
```

One Source of Truth: The Bridge




```rust
#[diplomat::bridge]
mod ffi {
    #[diplomat::opaque]
    #[diplomat::rust_link(icu::fixed_decimal::FixedDecimal, Struct)]
    pub struct FixedDecimal(pub icu::fixed_decimal::FixedDecimal);
    impl FixedDecimal {
        #[diplomat::attr(auto, constructor)]
        pub fn new(v: i32) -> Box<FixedDecimal> {
            Box::new(FixedDecimal(icu::fixed_decimal::FixedDecimal::from(v)))
        }
    }
}
```

Extensible: Diplomat Backends




```
(func $icu4x_FixedDecimal_new_mv1 (param
i32) (result i32))
```

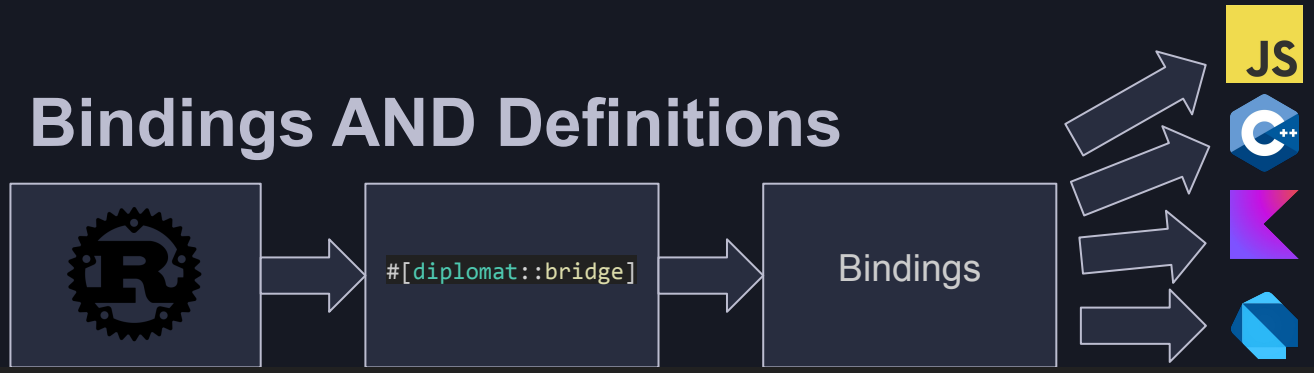
```
typedef struct FixedDecimal FixedDecimal;

FixedDecimal*
icu4x_FixedDecimal_new_mv1(int32_t v);
```

```
@ffi.Native<ffi.Pointer<ffi.Opaque>
Function(ffi.Int32)>(isLeaf: true, symbol:
'icu4x_FixedDecimal_new_mv1')
external ffi.Pointer<ffi.Opaque>
_icu4x_FixedDecimal_new_mv1(int v);
```

```
internal interface FixedDecimalLib: Library {
    fun icu4x_FixedDecimal_new_mv1(v: Int):
Pointer
}
```

Bindings AND Definitions




```
export class FixedDecimal {
    #ptr = null;
    constructor() {
        this.#ptr =
wasm.icu4x_FixedDecimal_new_mv1(v);
    }
}
```

```
final class FixedDecimal implements ffi.Finalizable {
  final ffi.Pointer<ffi.Opaque> _ffi;
  FixedDecimal._fromFfi(this._ffi);
  factory FixedDecimal(int v) {
    final result = _icu4x_FixedDecimal_new_mv1(v);
    return FixedDecimal._fromFfi(result);
  }
}
```

```
class FixedDecimal {
  public:
  inline static std::unique_ptr<FixedDecimal>
new_(int32_t v) {
      auto result = icu4x_FixedDecimal_new_mv1(v);
      return std::unique_ptr<FixedDecimal>(result);
  }
}
```

```
class FixedDecimal internal constructor(internal val
handle: Pointer) {
  companion object {
    fun new_(v: Int): FixedDecimal {
      return
FixedDecimal(lib.icu4x_FixedDecimal_new_mv1(v))
    }
  }
}
```

Results for ICU4X

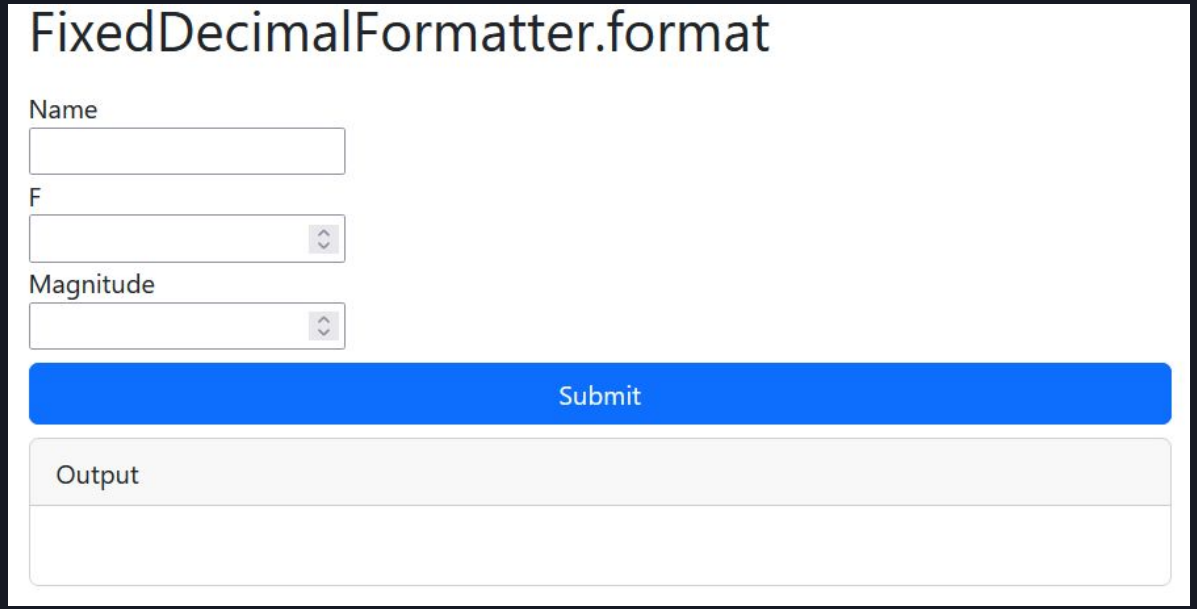
ICU4X requires only one definition crate:

Immediate support for ALL of Diplomat's supported languages

Minimal tweaking, some minor maintenance (as ICU4X changes and Diplomat is still in development)

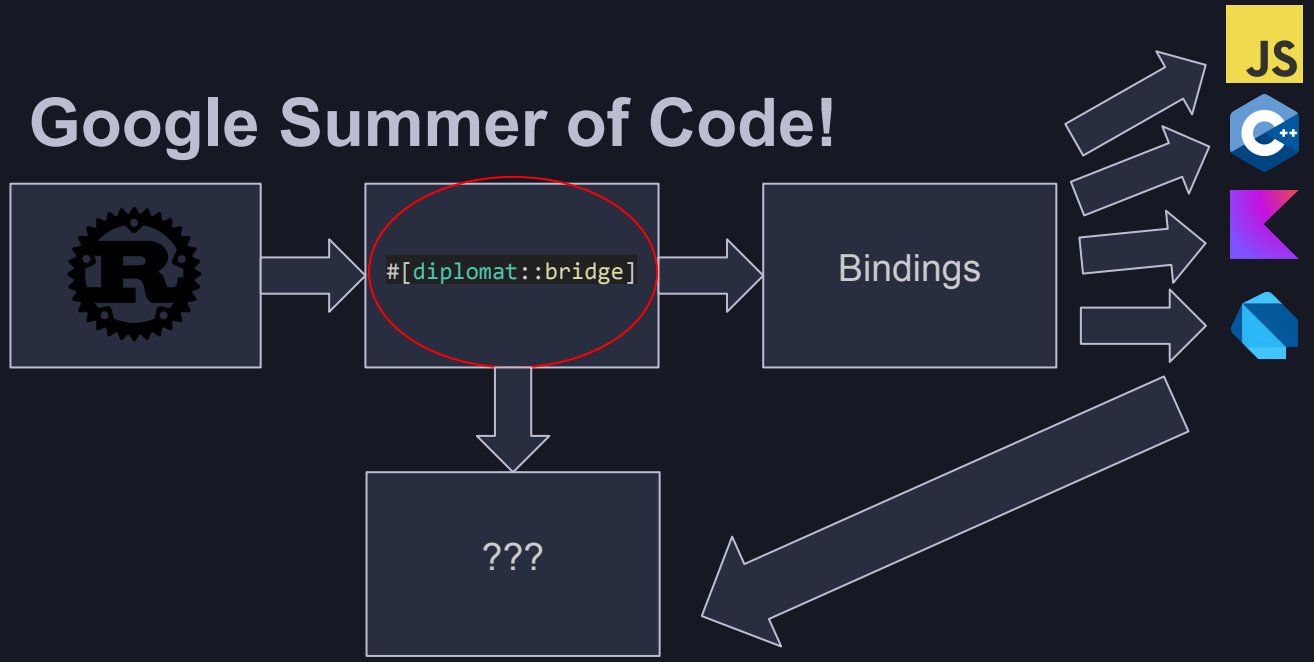
Plus…

ICU4X on the Web!



Live Demo: https://unicode-org.github.io/icu4x/wasm-demo/

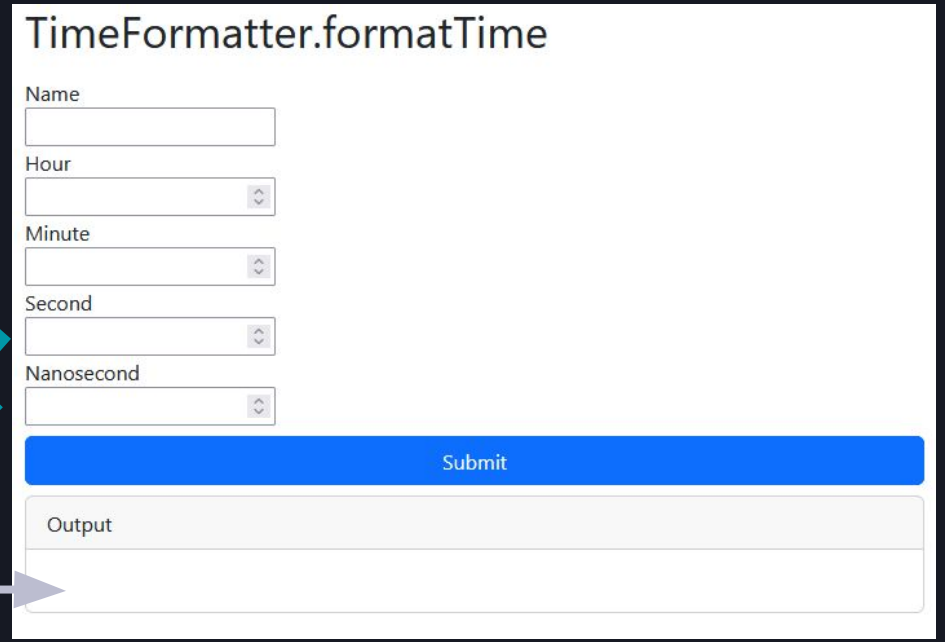
Google Summer of Code!



Diplomat and the Web (demo_gen)

```
pub fn format_time(&self,
value: &Time, write: &mut
diplomat_runtime::DiplomatWr
ite);
```

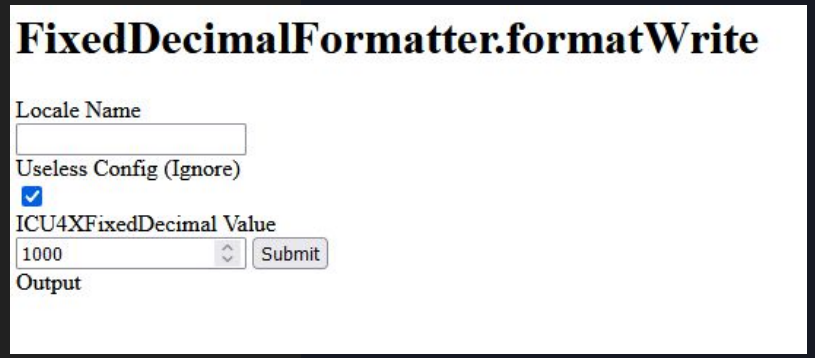
```
impl Time {
        pub fn create(
            hour: u8,
            minute: u8,
            second: u8,
            nanosecond: u32,
        ) -> Result<Box<Time>,
CalendarError>
}
```



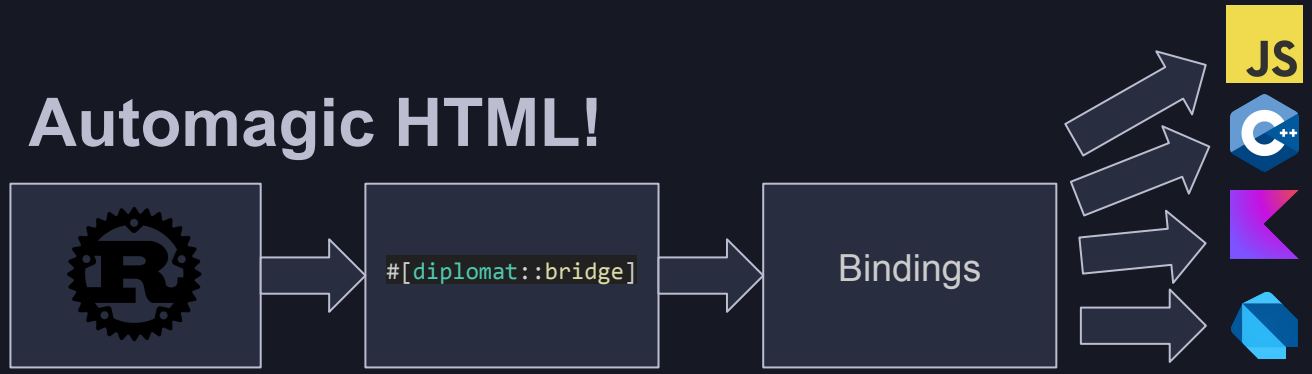
Object Schema

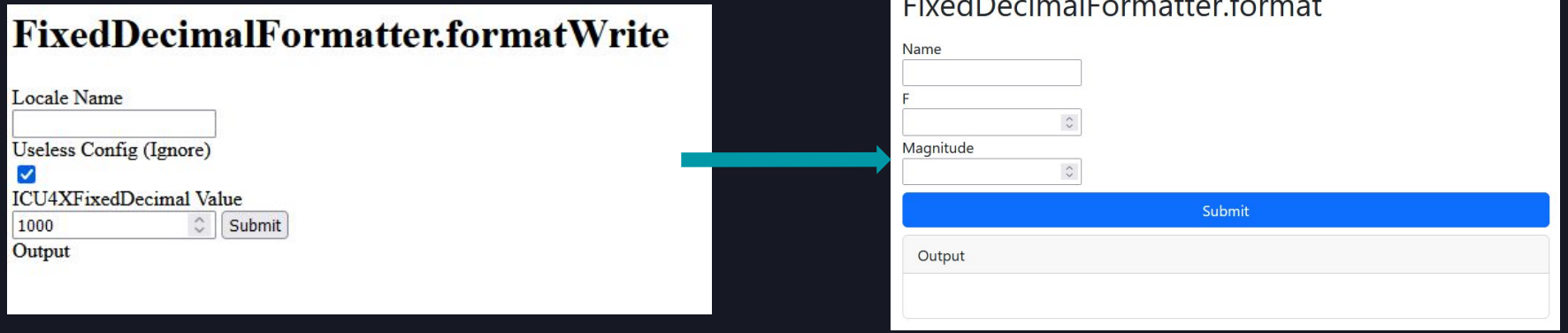
Provide metadata for generated JS functions

```
export const RenderInfo = {
  "FixedDecimalFormatter.formatWrite": {
    func: FixedDecimalFormatterDemo.formatWrite,
    funcName: "FixedDecimalFormatter.formatWrite",
    parameters: [
      {
        name: "Locale Name",
        type: "string"
      },
      // ...
    ]
  },
};
```



Automagic HTML!

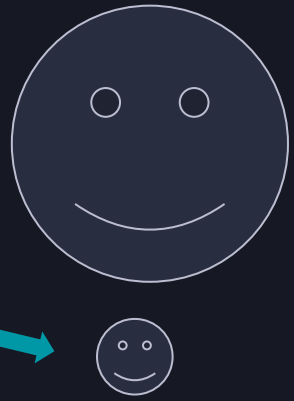




Open to contributions!

Looking for users!

This could be YOU



…and your Rust library

Always looking for contributors to expand on our backends!

The End

Thank you to:

- Shane Carr
- Manish Goregaokar
- Robert Bastian
- Elango Cheran
- Organizers of UTW 2024
- And YOU!

Questions?

Outline

- What is Diplomat? (5 minutes)
  - History/Problem Statement
    - Sample problem: ICU4X, porting Rust library to many languages
  - Solution Statement: Diplomat
- Diplomat's design (7 minutes)
  - No action-at-a-distance
    - "What you see is what you mean"
  - Ready to use
  - Uses Rust for definition
    - No IDLs
    - Seamless Integration of API and code
    - Easily extensible
  - Should pepper with some visual examples (how a bridge gets converted to C, JS, etc.)
- Generating Web Demos Automatically (10 minutes)
  - Live Demo
  - How the demo (roughly) works
    - Diplomat's design allows us to do this naturally
    - Diagrams!
  - Current challenges
- Call to action (3 minutes)
  - What we're working on for the future

Notes