YEW

Rust + WebAssembly framework

Denis Kolodin

Tech Lead at Exonum, Bitfury Group

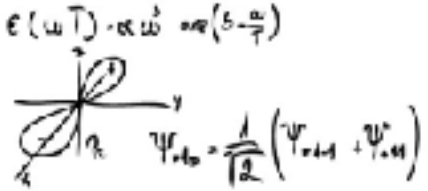
Agenda

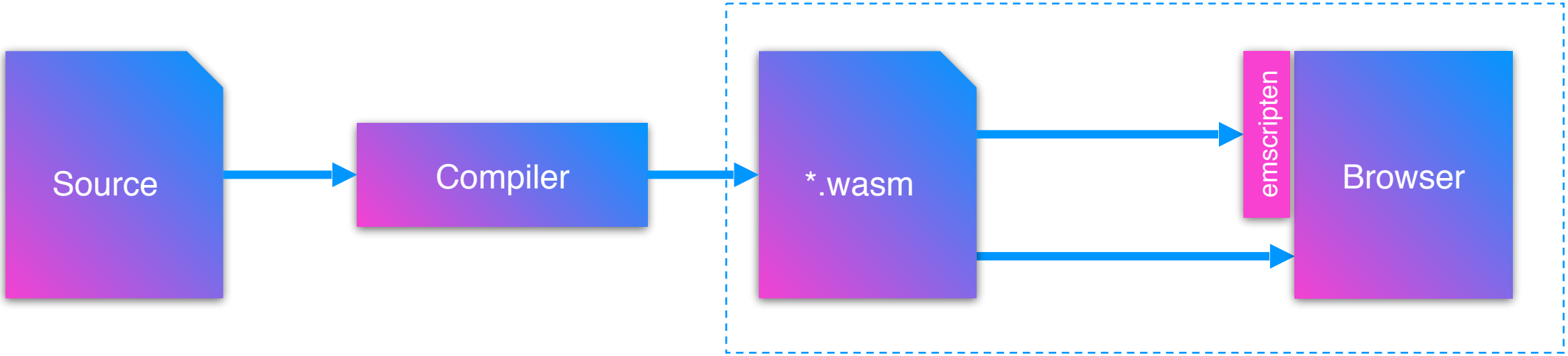
- What is Yew?
- Architectural aspects
- Features
- Extra benefits of Rust
- How to start?

WHAT IS YEW?

WebAssembly is the future!

- **Complex algorithms**
- A standard
- Many programming languages
- JavaScript remains necessary to a start WASM module





Yew – ready-to-use wasm framework

- Created **16 Dec 2017**
- Collected more than **>4k** stars on GitHub
- Open-source (MIT/Apache)
- Written in **Rust**
- Supports pure ***wasm32-unknown-unknown*** target
- Many examples!

DenisKolodin/yew

ARCHITECTURE

Elm/Redux like state changing

```
type alias Model =
    { value : Int
    }

type Msg
    = Increment
    | Decrement
```

```
case msg of
    Increment ->
      { value = model.value + 1 }
    Decrement ->
      { value = model.value - 1 }
```

model is immutable!

Rust implementation

```rust
struct Model {
    value: i64,
}

enum Msg {
    Increment,
    Decrement,
}
```

```rust
match msg {
    Msg::Increment => {
        self.value += 1;
    }
    Msg::Decrement => {
        self.value -= 1;
    }
}
```

model is **mut**able!

Why mutable is better than immutable?

```
fn update(&self, msg: Msg) -> Self {
    match msg {
        Msg::Increment => {
            Self {
                value: self.value + 1,
            }
        }
        Msg::Decrement => {
            Self {
                value: self.value - 1,
            }
        }
    }
}
```

Clone if even no changes

Copying

```
fn update(&mut self, msg: Msg) {
    match msg {
        Msg::Increment => {
            self.value += 1;
        }
        Msg::Decrement => {
            self.value -= 1;
        }
    }
}
```

Reusing

Why we need to keep the old model?

For rendering, to check the model has changed, but...

Change by comparison is **not effective**!

```
fn update(&mut self, msg: Self::Msg, _: &mut Env<Context, Self>) -> ShouldRender {
    match msg {
        Msg::UpdateValue(value) => {
            self.value = value;
            false
        }
        Msg::Commit => {
            self.name = self.value.clone();
            true
        }
    }
}

fn view(&self) -> Html<Context, Self> {
    html! {
        <div>
            <input oninput=|e: InputData| Msg::UpdateValue(e.value), />
            <button onclick=|_| Msg::Commit,>{ "Apply" }</button>
            <p>{ self.name.chars().rev().collect::<String>() }</p>
        </div>
    }
}
```

Model had changed…

but the template has not!

What if your model has hidden fields?

Developer knows better when it should be rendered!

FEATURES

Start from scratch and move to stdweb

```
const JS: &'static [u8] = b"\
    var handler_fn_ptr = $0;\
    var app_system = $1;\
    var method = UTF8ToString($2);\
    var url = UTF8ToString($3);\
    var body = UTF8ToString($4);\
    var header_len = $5;\
    var header_key_ptr = $6;\
    var header_value_ptr = $7;\
    var timeout = $8;\
    var handler_data_ptr = $9;\
    var handler_vtable_ptr = $10;\
    var xhr = new XMLHttpRequest();\
    var error_fn = function(error_sig) { return function() {\
        Runtime.dynCall('viiiiiiii', handler_fn_ptr, [error_sig,
app_system, handler_data_ptr, handler_vtable_ptr, 0, 0, 0, 0]);\
    } };\
    xhr.addEventListener('timeout', error_fn(1));\
    xhr.addEventListener('error', error_fn(2));\
    xhr.addEventListener('load', function() {\
        var stack = Runtime.stackSave();\
        var status_code = xhr.status;\
        var status_text = allocate(\
            intArrayFromString(xhr.statusText), 'i8',
ALLOC_STACK\
        );\
"; // and many more...
```

koute/stdweb

```
let handle = js! {
    var callback = @{callback};
    var action = function() {
        callback();
    };
    var delay = @{ms};
    return {
        interval_id: setInterval(action, delay),
        callback: callback,
    };
};
```

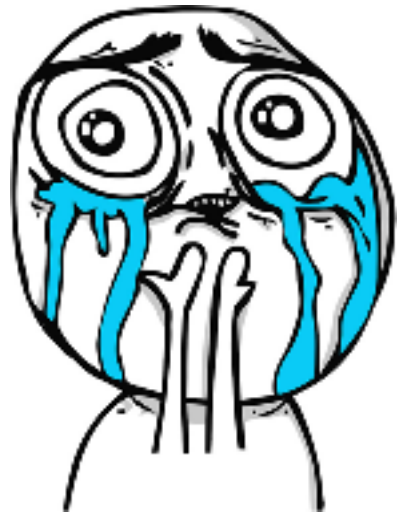
Templates in pure code are boring!

```
fn view(&self) -> Html<Context, Self> {
    nav("nav", ("menu"), vec![
        button("button", (), ("onclick", || Msg::Clicked)),
        tag("section", ("ontop"), vec![
            p("My text...")
        ])
    ])
}
```

JSX-like templates with Rust macro

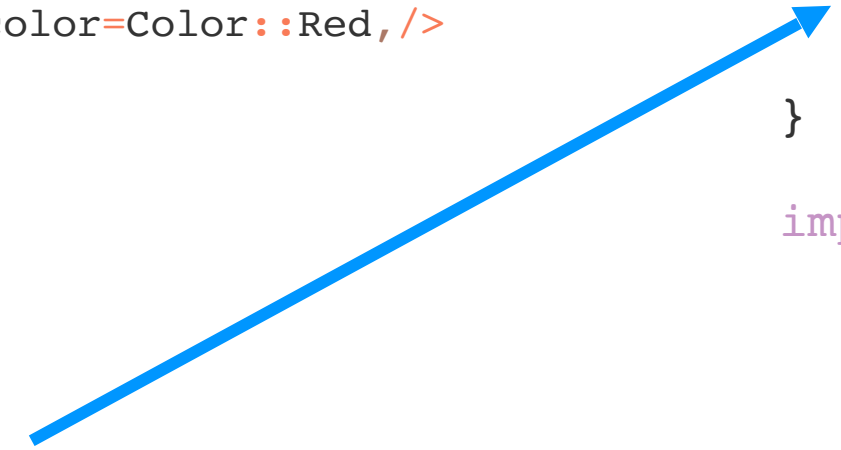
```
impl Renderable<Context, Model> for Model {
    fn view(&self) -> Html<Context, Self> {
        html! {
            <div>
                <nav class="menu",>
                    <button onclick=|_| Msg::Increment,>{ "Increment" }</button>
                    <button onclick=|_| Msg::Decrement,>{ "Decrement" }</button>
                </nav>
                <p>{ self.value }</p>
                <p>{ Local::now() }</p>
            </div>
        }
    }
}
```

Think of this as pure code templates **on steroids!**



Components: crazy strong typed feature

```
html! {
    <div class="counters",>
        <Counter: initial=100,/>
        <Counter: color=Color::Red,/>
    </div>
}
```



Properties is a simple **struct**

```
#[derive(PartialEq, Clone)]
pub struct Properties {
    pub initial: u32,
    pub color: Color,
    pub onclick: Option<Callback<u32>>,
}

impl Default for Properties {
    fn default() -> Self {
        Properties {
            initial: 0,
            color: Color::Green,
            onclick: None,
        }
    }
}
```

Typed scopes

```rust
pub struct Button {
    title: String,
    button_clicked: Option<Callback<()>>,
}
```

```rust
use button::Button as MyButton;

html! {
    <nav class="menu",>
        <MyButton: color=Color::Red,/>
        <MyButton: button_clicked=
            |_| ParentMsg::DoIt,/>
    </nav>
}
```

```rust
impl<CTX: 'static> Component<CTX> for Button {
    fn update(&mut self, msg: Self::Msg, _: &mut Env<CTX, Self>)
        -> ShouldRender
    {
        match msg {
            Msg::Clicked => {
                if let Some(ref mut callback) = self.button_clicked {
                    callback.emit(());
                }
            }
        }
        false
    }
}
```

Matches the
parent's message type

Other features

Easy serialization

```
Msg::Store => {
    context.local_storage.store(KEY, Json(&model.clients));
}
Msg::Restore => {
     if let Json(Ok(clients)) = context.local_storage.restore(KEY)
{
          model.clients = clients;
    }
}
```

Fragments

```
html! {
    <>
        <tr><td>{ "Row" }</td></tr>
        <tr><td>{ "Row" }</td></tr>
        <tr><td>{ "Row" }</td></tr>
    </>
}
```

EXTRA BENEFITS

Services: interact with the world outside

- TimeoutService
- IntervalService
- FetchService
- WebSocketService

- Custom services

Services are Rust crates!
https://crates.io

Context: declare the requirements

```rust
struct Context {
    web: FetchService,
    ws: WebSocketService,
    aws: AwsService,
}

impl WithAws for Context {
    fn upload_to_s3(&mut self, data: Vec<u8>)
        -> UploadTask {
        unimplemented!("do something here...");
    }
}
```

Implementation

Requirement

```rust
trait WithAws {
    fn upload_to_s3(&mut self, data: Vec<u8>)
        -> UploadTask;
}

impl<T: WithAws> Component<T> for MyComponent {
    fn update(&mut self, msg: Self::Msg,
        context: &mut Env<Context, Self>)
        -> ShouldRender {
        match msg {
            Msg::UploadIt(data) => {
                self.upload_task =
                    context.upload_to_s3(data);
                false
            }
        }
    }
}
```

Compiler won’t let you be wrong

```rust
#[derive(PartialEq, Clone)]
pub struct Props {
    pub limit: u32,
    pub onsignal: Option<Callback<()>>,
}
```

code

What if you set onsgnal property?

i

Compiler won’t let you be wrong

```
impl<CTX: AwsS3Api + 'static> Component<CTX> for Button {
    type Msg = Msg;
    type Properties = Props;
}
```

code

What if your context doesn’t implement AwsS3Api trait?

START HERE!

Rust-to-wasm compilation

Install Rust

curl https://sh.rustup.rs -sSf | sh

Install cargo-web

cargo install cargo-web

Create a project

cargo new --bin my-project

Run!

cargo web start --target wasm32-unknown-unknown

emscripten might come in handy!

```
[package]
name = "my-project"
version = "0.1.0"

[dependencies]
yew = "0.3.0"
```

Current restrictions

- Some crates doesn’t support target `wasm32-unknown-unknown`
  - No threads!
  - No system API
- WASM debugger exists, maybe…

The future of the framework

- Integration with JS *(but don't use it in the core)*
- Typed CSS
- Ready-to-use components
- Improve performance for some cases
- More core contributors
- Docs, tutorials, books

THANK YOU!