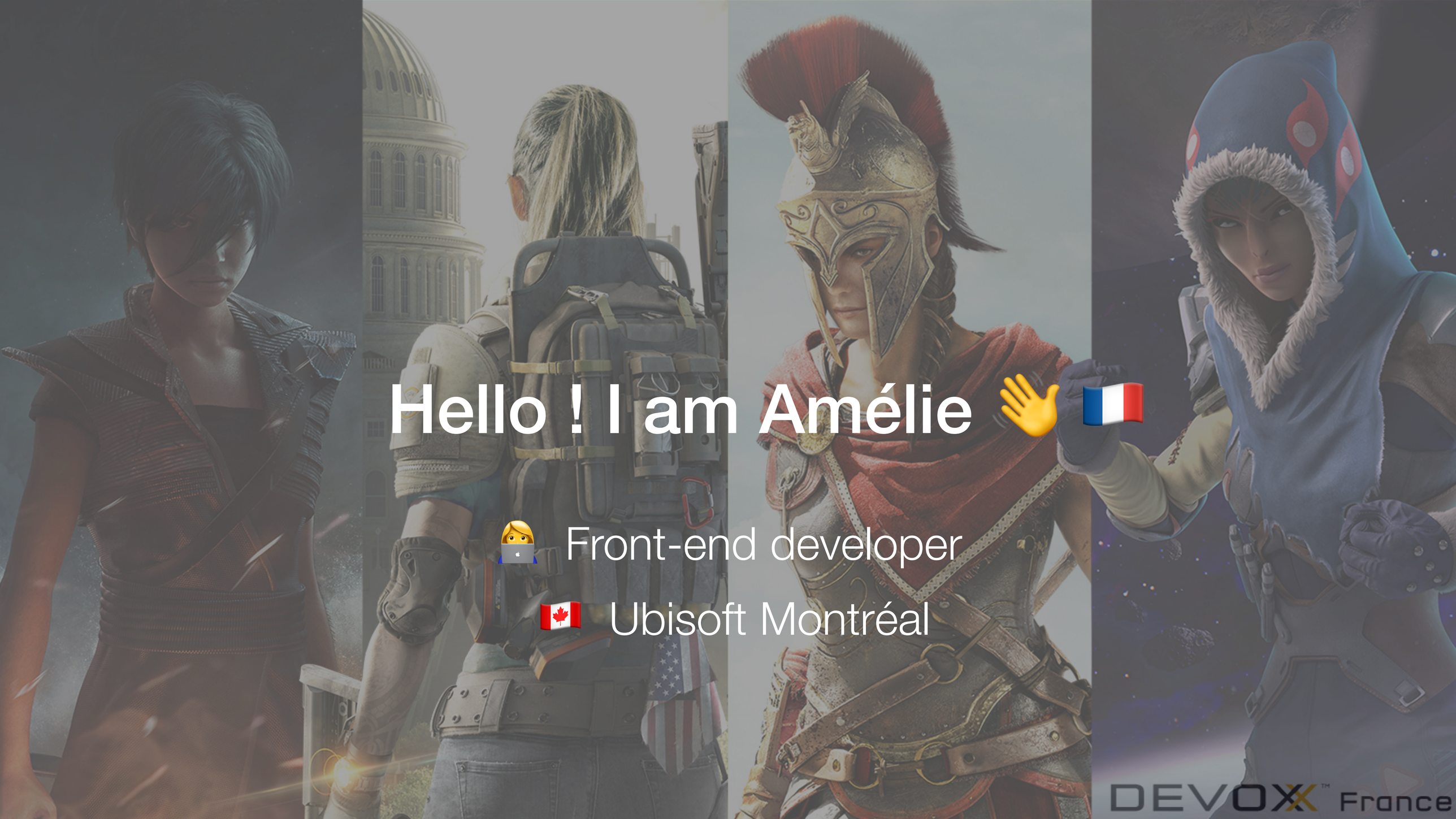
The Heart has its Reason(ML)

Amélie Benoit

@AmelieBenoit33



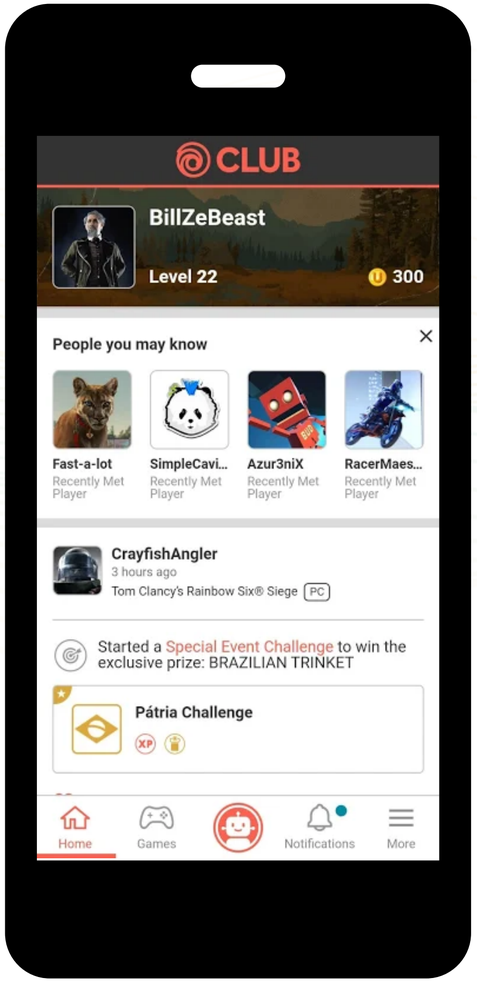
Hello ! I am Amélie 👋🇫🇷

👩‍💻 Front-end developer

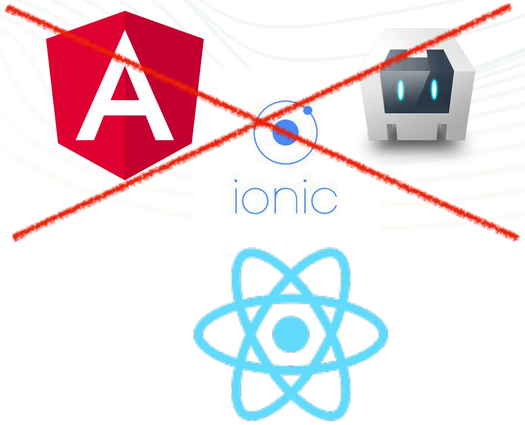
🇨🇦 Ubisoft Montréal

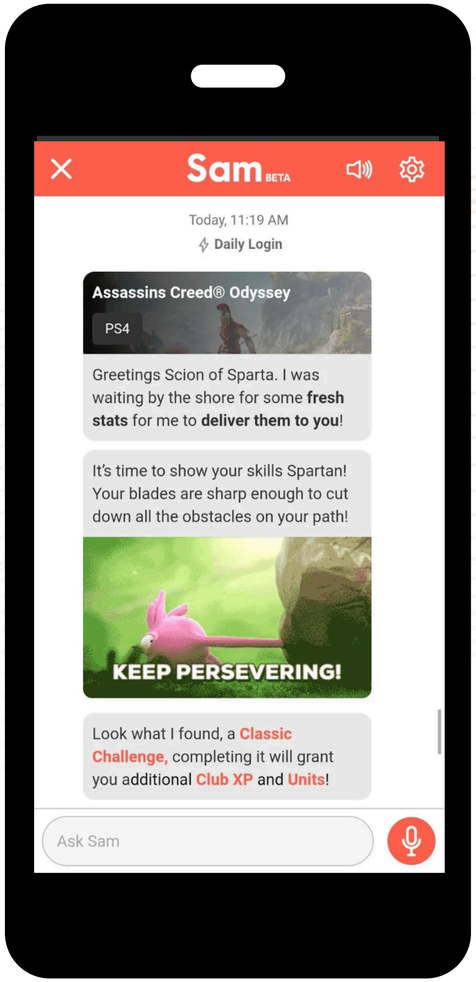
DEVOXX™ France

CLUB

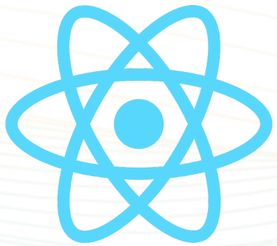


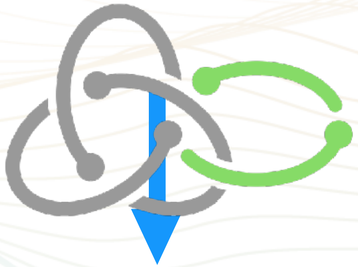
115K
USERS

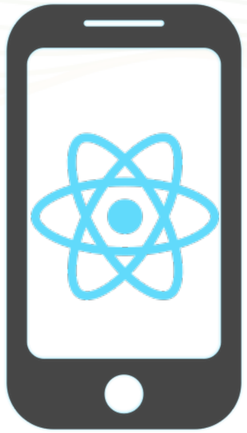




My Journey







How do I even declare a variable ?

- Me, 1st day of work

A new language (again)...

ReasonML

Not a new language
more a **syntax extension**

Sponsored and used for **Messenger.com** @Facebook

1. 100% type coverage
   Type inference
   Solid, static typing

2. Types are inferred at the **COMPILATION**

3. **Refactors** 👌

4. undefined is a function
   cannot read property ‘x’ of undefined

Langage fortement typé VS faiblement typé

RE (language strongly typed)

JS (language weakly typed)

Functional programming

Base principles

- Immutability `ex. Redux state`
- Pure functions `ex. function map or filter`
- Avoid mutations, shared states and side effects

allows to use imperative code when you really need it

You already know some of the syntax !

Declaration	`let myVar = “hello”;`
Conditional	`if() { }` `else { }`
Types	string, int, float, array, list …
Object / Record	`{x:30, y: 20}` `{…point, y: 30}`
Destructuring	`let {x, y} = data`
Function & Block	`let myFunction = (x, y) => { x ++ y }`
Boolean	Same as in Javascript (`true`, `false`, `<`, `>`, `&&`, `||` …)

And some new syntaxes / types ...

Module

```
module MyModule = { }
```

Type

```
let myInt:int = 5;
```

Alias on type

```
type level = int;
```

Option

```
type option('a) = None | Some(‘a)
```

Variant

```
type animal =
| Dog
| Cat;
```

Named and optional parameters

```
let user = (~name, ~age=18) => { }
user(~name=“Amélie“);
user(~name=“Amélie“, ~age=18);
```

Fast Pipe (pipe first)

```
parseData(person) -> getAge    getAge(parseData(person))
                  -> isMinor   isMinor(getAge(parseData(person)))
```

Pattern matching

Variants

```
let toutou = Dog;
switch toutou {
| Dog => “Woof“
| Cat => “Meow“
}
```

Options

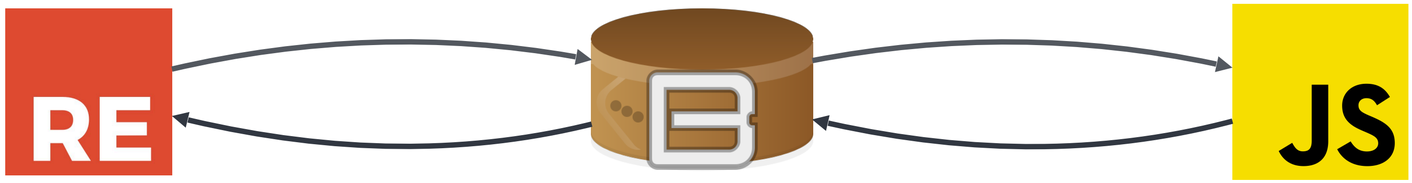
```
let driverLicence = None;
switch driverLicence {
| None => “No Licence !“
| Some(serial) => “N°“ ++ serial
}
```

Anything !

```
let myPet = Dog;
let isBig = true;
switch (myPet, isBig) {
| (Dog, true) => “Big Dog !“
| (Dog, false) => “Small Dog“
| (Cat, _) => “A cat“
}
```

Let’s talk compilation

BuckleScript



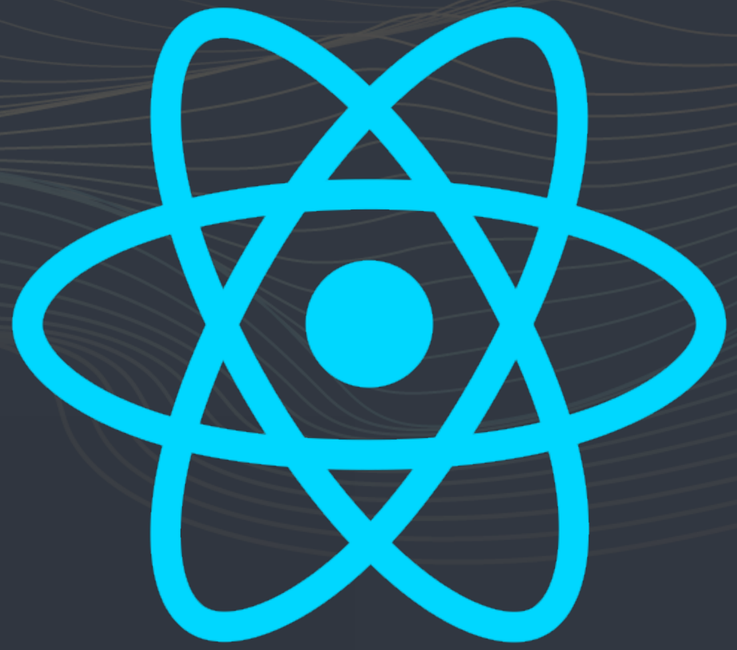
BuckleScript allows you to

- Create bindings (interop)
- Compile ReasonML to Javascript

« Simple, small and blazing fast build workflow. »

ReasonReact

Safer, simpler way to build React components in Reason



Installation

EDIT

Note: for general Reason + BuckleScript editor setup, see here.

Bsb

```
npm install -g bs-platform
bsb -init my-react-app -theme react
cd my-react-app && npm install && npm start
# in another tab
npm run webpack
```

BuckleScript's bsb build system has an `init` command that generates a project template. The `react` theme offers a lightweight solution optimized for low learning overhead and ease of integration into an existing project.

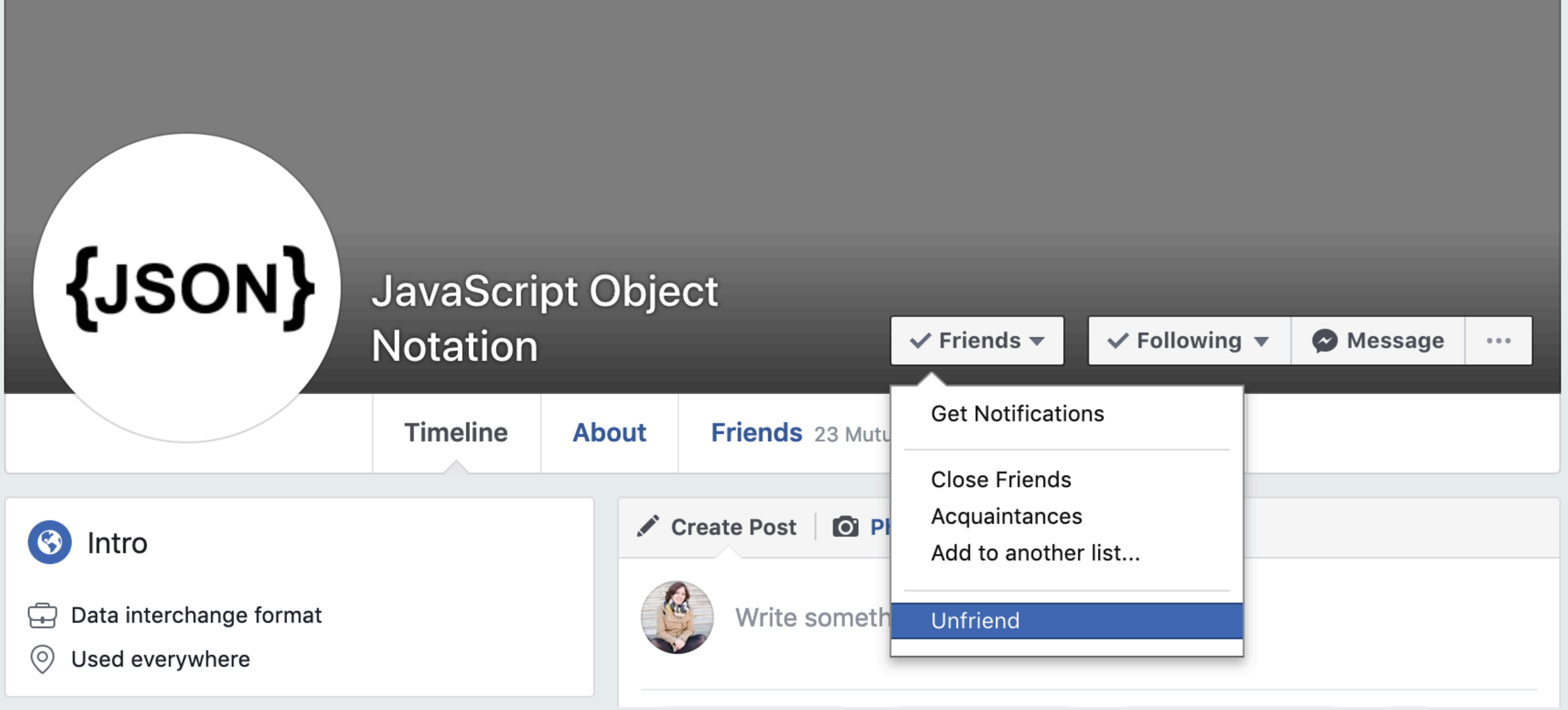
It compiles to straightforward JS files, so you can open `index.html` directly from the file system. No server needed.

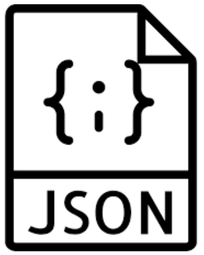
← PREVIOUS NEXT →

Demo

« WOW »

- You


{JSON}
JavaScript Object Notation
Friends
Following
Message
Get Notifications
Close Friends
Acquaintances
Add to another list...
Unfriend
Timeline
About
Friends 23 Mutu
Intro
Data interchange format
Used everywhere
Create Post
Write someth

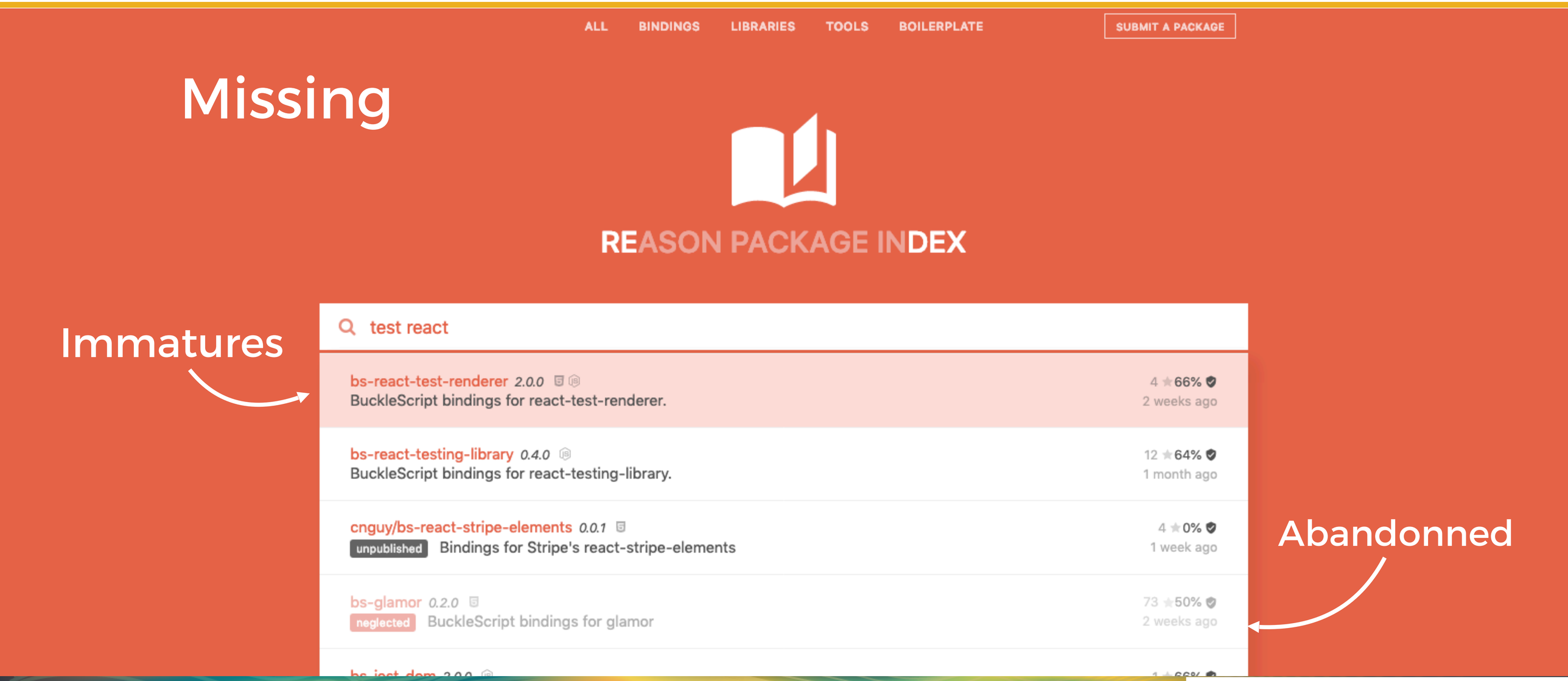



```
{
  "graphqlServer": {
    "uri": "http://localhost:4000/graphql"
  },
  "sentry": {
    "enable": false
  },
  "autoLogin": {
    "enable": true,
    "username": "user@domain.com",
    "password": "qwerty"
  },
  "servicesEnv": "uat",
  "storybook": {
    "enable": false,
    "embed": false,
    "host": "localhost"
  }
}
```

```
module Decode = {
  let graphqlServer = json =>
    Json.Decode.{uri: json |> field("uri", string)};

  let storybook = json =>
    Json.Decode.{
      enable: json |> field("enable", bool),
      embed: json |> field("embed", bool),
      host: json |> field("host", string),
    };

    sentry = json => Json.Decode.{enable: json |> field("enable", bool)};

    autoLogin = json =>
    on.Decode.{
    enable: json |> field("enable", bool),
    username: json |> field("username", string),
    password: json |> field("password", string),


    configuration = json =>
    itch (
    Json.Decode.{
      graphqlServer: json |> field("graphqlServer", graphqlServer),
      sentry: json |> field("sentry", sentry),
      servicesEnv:
        json |> field("servicesEnv", string) |> Service.toEnvironment,
      autoLogin: json |> field("autoLogin", autoLogin),
      storybook: json |> field("storybook", storybook),
    }
    {
    conf => conf
    exception ex =>
    Js.Exn.raiseError @@ {j|Error while parsing config file: $ex|j} |> ignore;
    raise @@ ex;
  };
};
```

Bindings JS → Reason

Missing



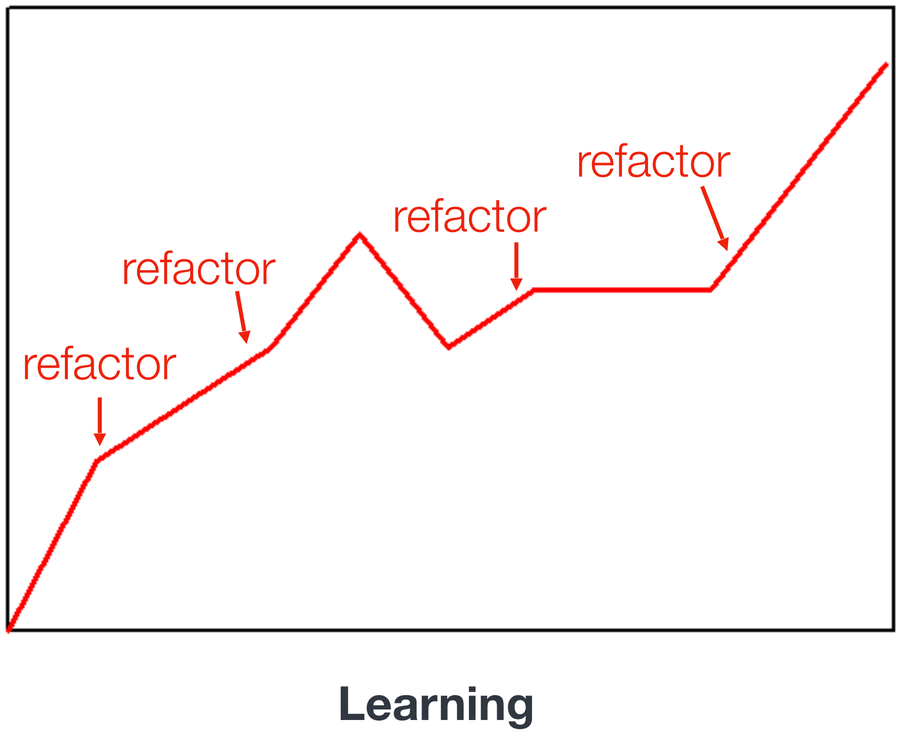
Immatures

Abandonned

Guidelines

- Good practices inferred from refmt
- We read a lot of source code from other libraries
- We discuss **A LOT**

In the end ...



- Team is highly motivated
- (Almost) no bug on the first features we built

Thank you !

Questions ?

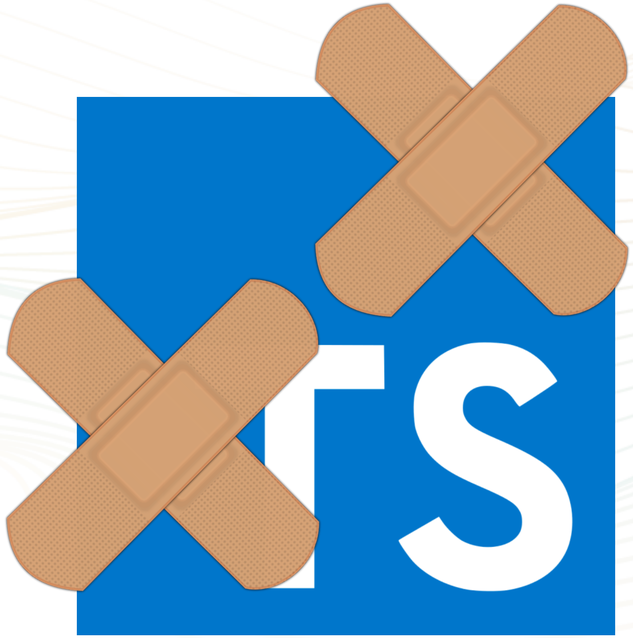
github.com/abenoit/devoxx-reasonml

@AmelieBenoit33

Typescript ?

- [x] Static typing
- [x] DefinitelyTyped

- [ ] Compilation time
- [ ] Type inference (?)



Can be functional somehow

Immutable data structure

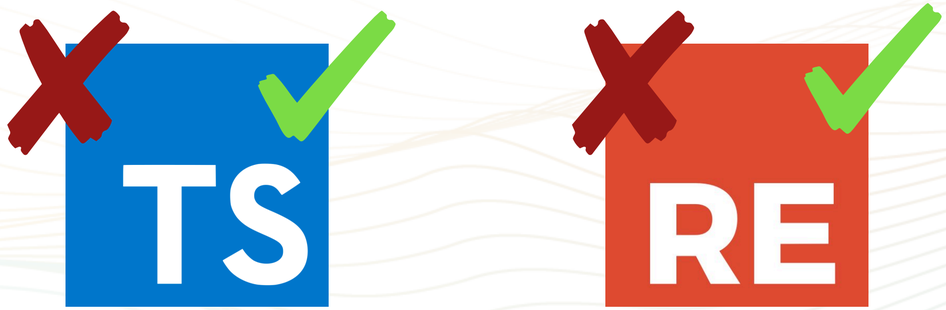
Immutable.js
(or set readonly prop)

Pipe operator ?

Proposal

Typescript or ReasonML

Development experience completely different



What matters is to be aware of the choices we make and what it involves

« Use whatever that makes you happy and productive »

- Axel Rauschmayer