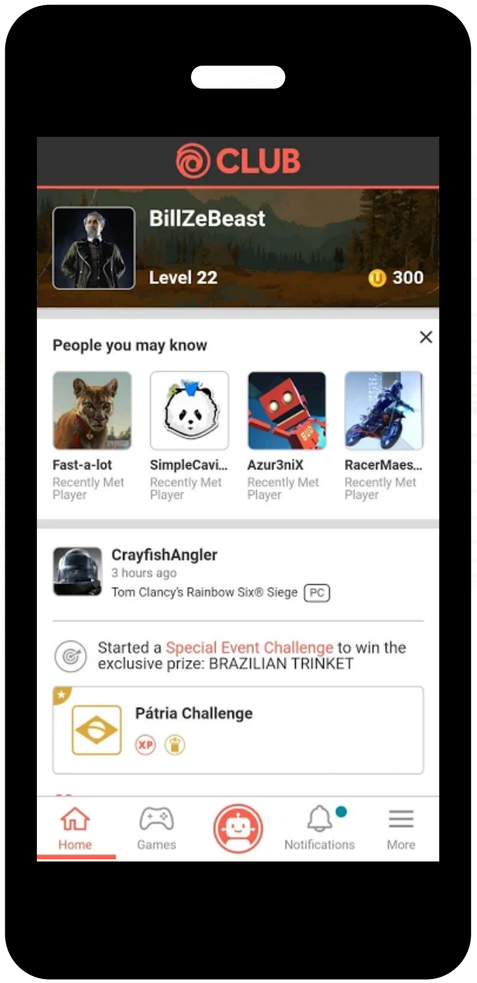
Le cœur a ses Reason(ML)

Amélie Benoit

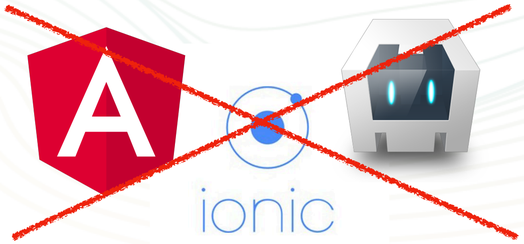
@AmelieBenoit33

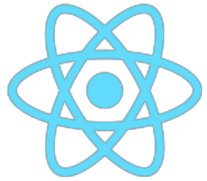

Salut ! Moi c'est **Amélie** 👋🇫🇷
👩‍💼 Développeuse Web
🇨🇦 Ubisoft Montréal
DEVOXX France

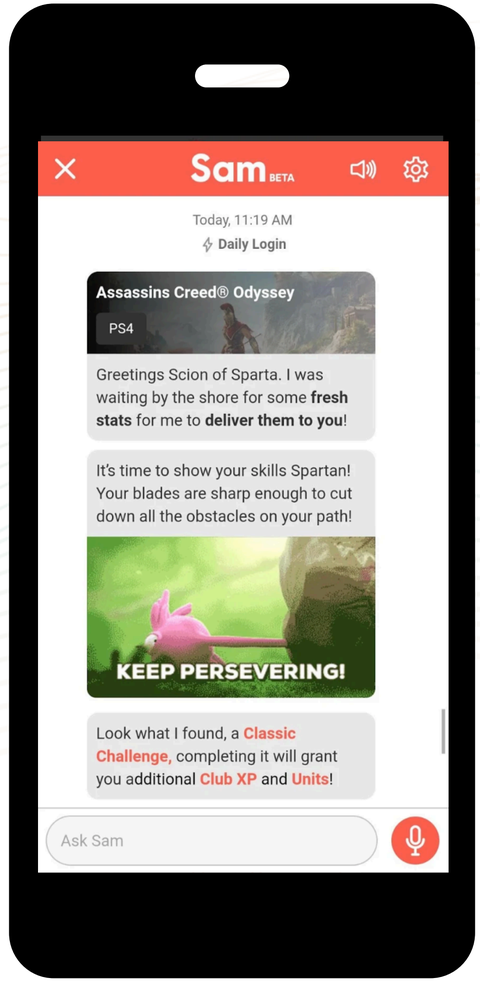
CLUB



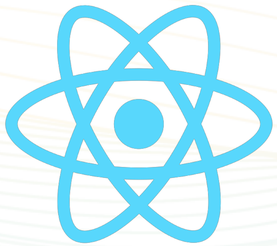
80K
USERS

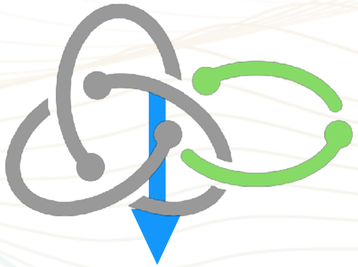


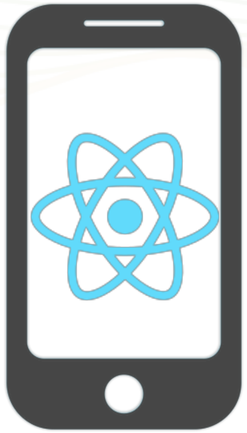




Mon parcours







« Comment je fais pour déclarer une variable ? »

- Moi, 1er jour de travail

ReasonML

Pas un nouveau langage,
mais une **extension de la syntaxe**

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

1. 100% type coverage
   Inférence de type
   Solidité

2. Les types sont inférés à la **COMPILATION**

3. **Refactos** 👌

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

Langage fortement typé VS faiblement typé

RE (langage fortement typé)

Int → Int

JS (langage faiblement typé)

Number 

???

source: Putain de code

Programmation fonctionnelle

- Immutabilité `ex. Redux state`
- Pure functions `ex. function map or filter`
- Éviter les mutations, les états partagés et les effets de bord

**Permet d'utiliser du code « impératif »
au besoin**

On retrouve des syntaxes qu'on connait

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, <, >, &&, || …)`

Et on trouve aussi de nouvelles syntaxes !

Module
```
module MyModule = { }
```

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

Alias on type
```
type level = int;
```

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

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

Named and optional parameters
```
let user = (~name, ~age=18) => { }
user(~name=“Amélie“);
user(~name=“Amélie“, ~age=28);
```

Fast Pipe (pipe first)
```
parseData(person) -> getAge     getAge(parseData(person))
                  -> isMinor    isMinor(getAge(parseData(person)))
```

Pattern matching

Variants

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

Options

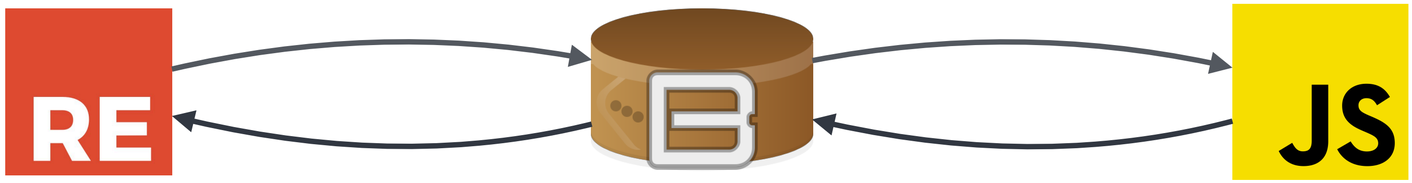
```
let driverLicence = None;
switch driverLicence {
| None => “Pas de permis“
| Some(serial) => “N°“ ++ serial
}
```

Anything !

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

Parlons compilation

BuckleScript



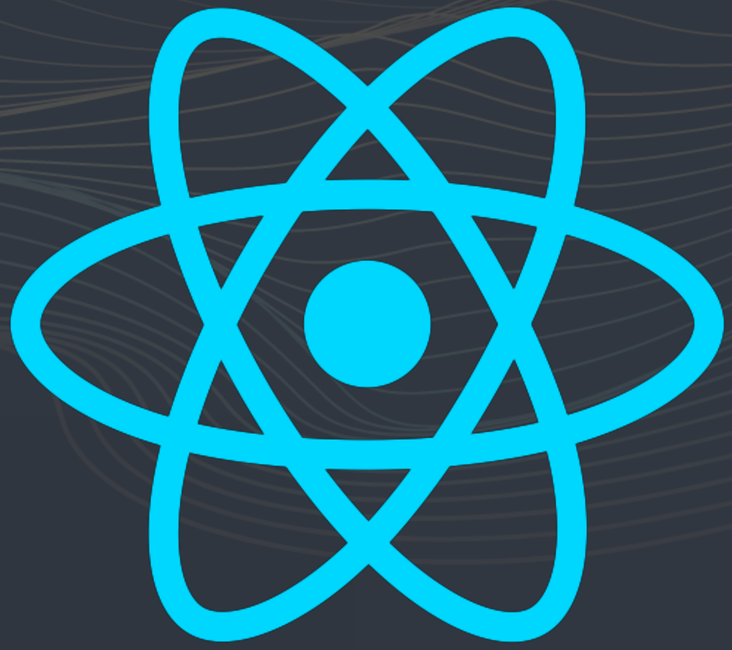
BuckleScript permet de

- Créer des bindings (interop)
- Compiler ReasonML vers 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.

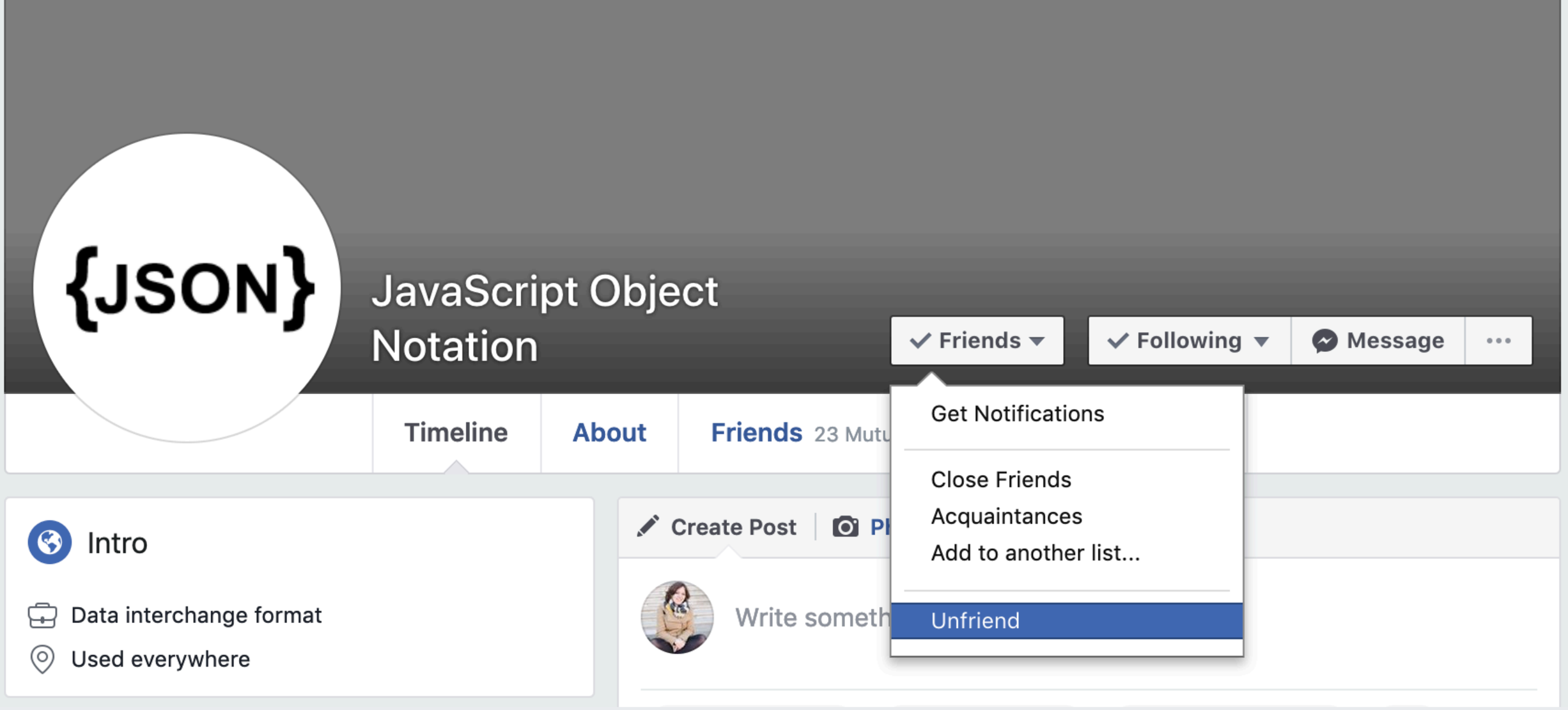
It compiles to straightforward JS files, so you can open `index.html` directly from the file system. No server needed.

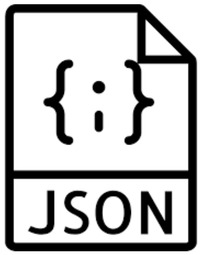
← PREVIOUS NEXT →

Demo

« WOW »

- Vous


{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

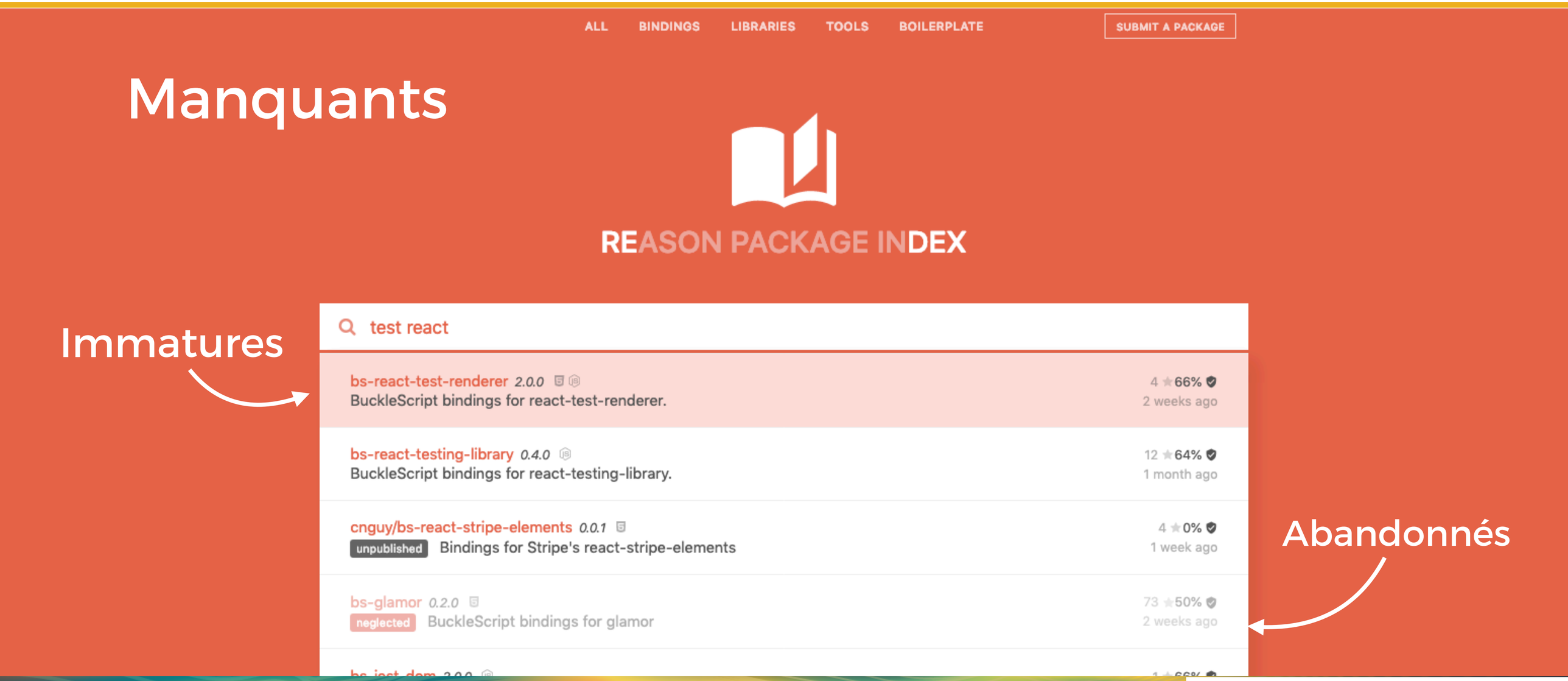



```
{
  "graphqlServer": {
    "uri": "http://localhost:4000/graphql"
  },
  "sentry": {
    "enable": false
  },
  "autoLogin": {
    "enable": true,
    "username": "user@domain.com",
    "password": "qwerty"
  },
  "servicesEnv": "uat",
  "storybook": {
    "enable": false,
    "embed": false,
  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

Manquants



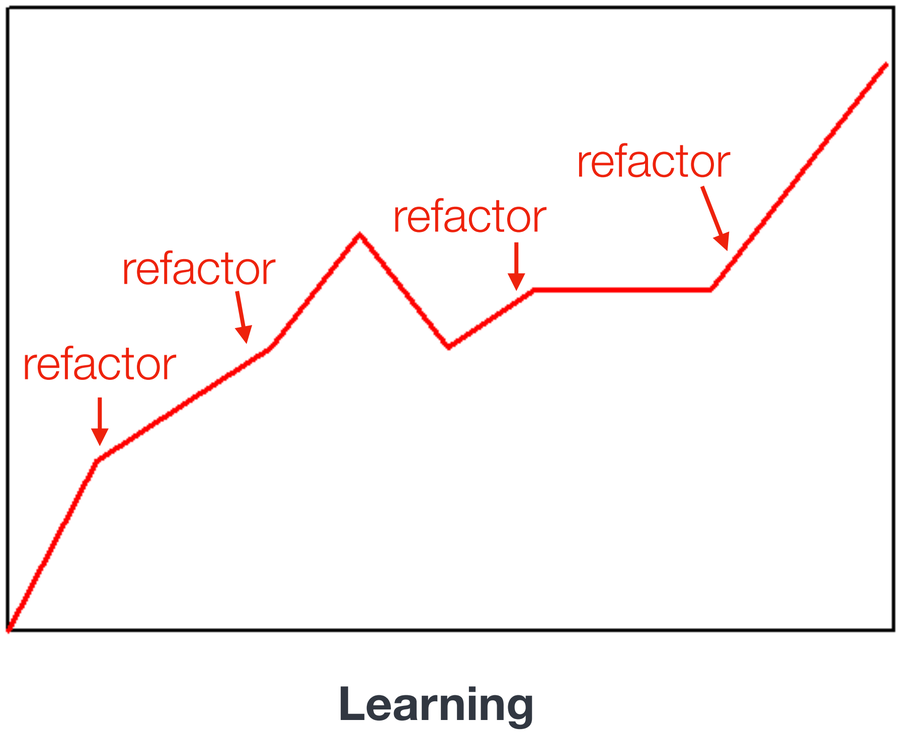
Immatures

Abandonnés

Guidelines

- Bonnes pratiques inférées par refmt
- Lecture du code source de beaucoup de librairies
- On discute **BEAUCOUP**

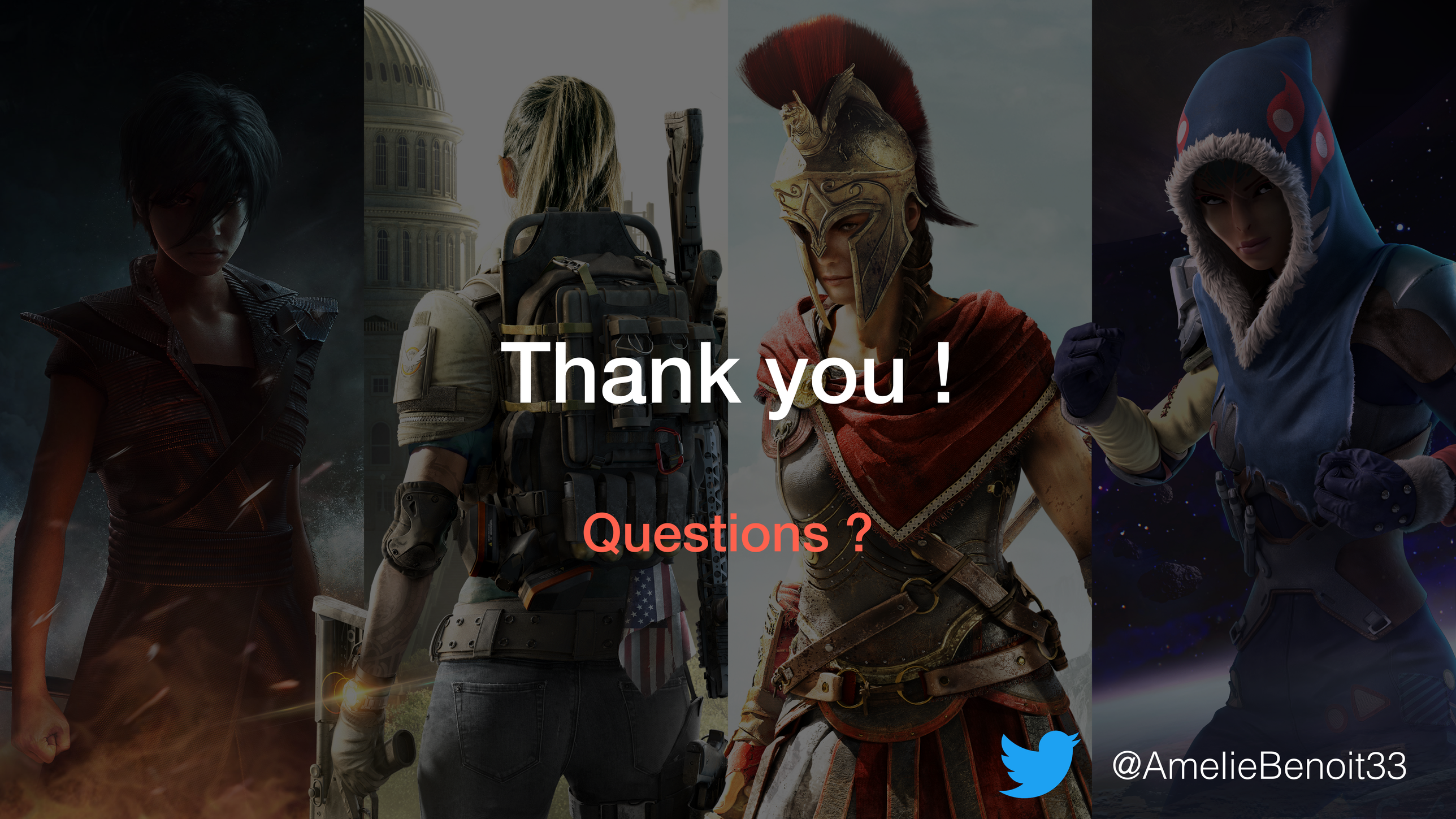
Finalement ...



- L'équipe est moteur, impliquée
- On participe à l'open-source

CLUB

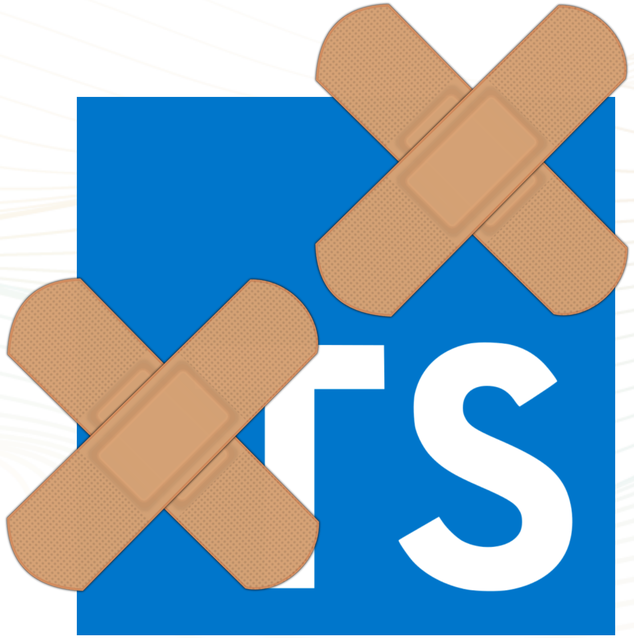
On recrute !


Thank you !
Questions ?
@AmelieBenoit33

Typescript ?

- ☑ Static typing
- ☑ DefinitelyTyped

- ☐ Temps de compilation
- ☐ Inférence de type (?)



Fonctionnel mais pas trop

Immutable data structure

Immutable.js
(or set readonly prop)

Pipe operator ?

Proposal

Typescript ou ReasonML

Expériences de développement complètement différentes.

L'important c'est d'être conscient de ses choix
et de ses implications

« Use whatever that makes you happy and productive »

- Axel Rauschmayer