ReasonML

Building type-safe react applications

```
type remoteData =
  | NotAsked
  | Pending
  | Succcess(data)
  | Error(string)

let render = (state) =>
  switch(state) {
  | NotAsked => "Let's learn about ReasonML!"
  | Pending => "Presentation starting..."
  | Success(data) => hd -> data.slide
  | Error(msg) => "Uh oh!" ++ msg
  }
```

October 29, 2019

Don't get me wrong I *love* JavaScript ヽ(^。^)ノ

What exactly is *ReasonML*?

How can it benefit developer and user experience?

story time!

1973
ML *(Metalanguage)*
Robin Milner

React!

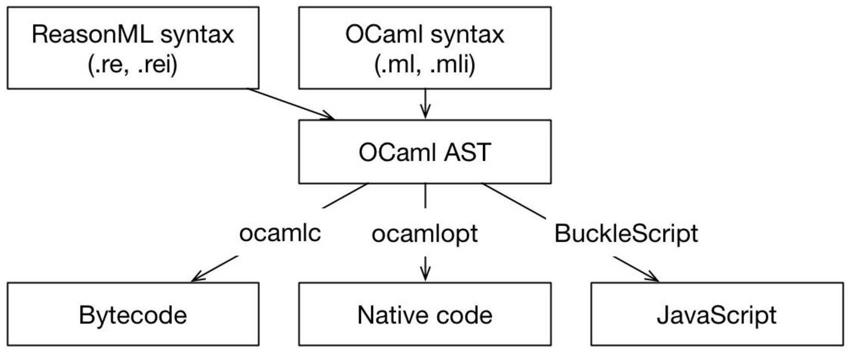
2013 React Open Sourced

2010 React Prototype

|

2013 React Open Sourced

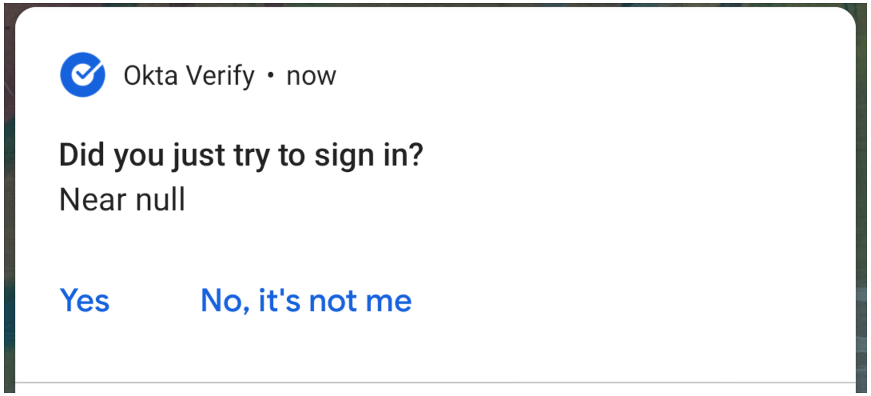
2016 ReasonML Born!


ReasonML syntax
(.re, .rei)
OCaml syntax
(.ml, .mli)
OCaml AST
ocamlc
ocamlopt
BuckleScript
Bytecode
Native code
JavaScript

- immutability first

- list, record, and tuple!

- ADT (sum/variant types)!


Okta Verify • now

Did you just try to sign in?
Near null

Yes
No, it's not me

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

type location = option(remoteData)

let currentLocation =
  switch(location) {
  | None => "your current location is unknown"
  | Some(location) => "your current location is " ++ location
  }
```

"each of types" vs "one of types"

"this *and* that" vs "this *or* that"

each of types
this *and* that

```
const person = {
  name: "Joe",
  age: 65
};

//each of types: person describes type of string and int
```

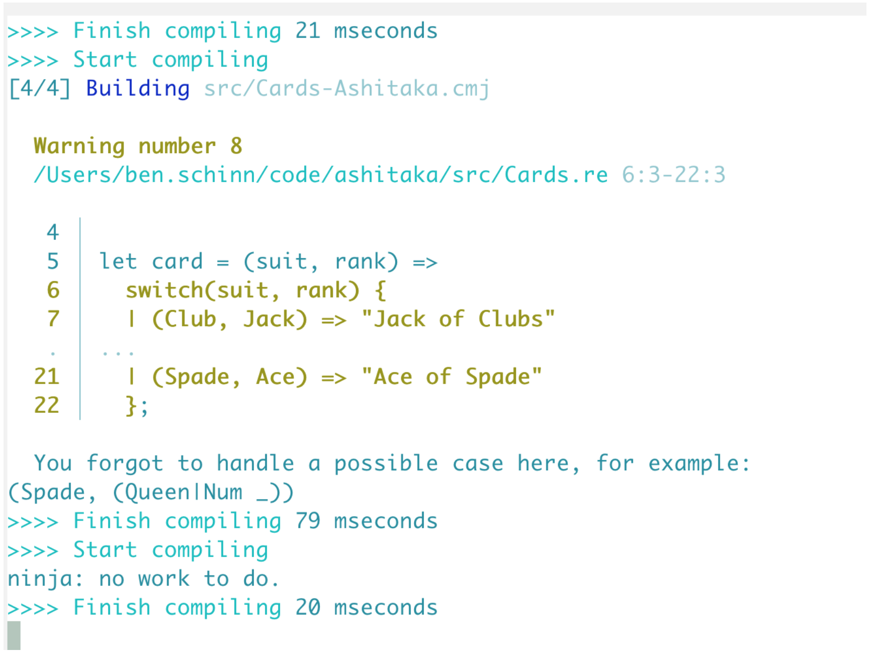
one of types
this *or* that

```
type suit =           type rank =
| Club                | Jack
| Diamond             | Queen
| Heart               | Ace
| Spade               | Num(int)
```

pattern matching!

```
let card =
switch(suit, rank) => {
| (Club, Jack) => <Card suit="Club" rank="Jack" />
| (Club, Queen) => <Card suit="Club" rank="Queen" />
| (Club, Num(num)) => <Card suit="Club" rank=(num -> string_of_int) />
...
_ => <InvalidCard />
}
```

pattern matching!



```
>>>> Finish compiling 21 mseconds
>>>> Start compiling
[4/4] Building src/Cards-Ashitaka.cmj

  Warning number 8
  /Users/ben.schinn/code/ashitaka/src/Cards.re 6:3-22:3

   4 │
   5 │ let card = (suit, rank) =>
   6 │   switch(suit, rank) {
   7 │   | (Club, Jack) => "Jack of Clubs"
   . │ ...
  21 │   | (Spade, Ace) => "Ace of Spade"
  22 │   };

  You forgot to handle a possible case here, for example:
(Spade, (Queen|Num _))
>>>> Finish compiling 79 mseconds
>>>> Start compiling
ninja: no work to do.
>>>> Finish compiling 20 mseconds
```

```
type remoteData =
  | NotAsked
  | Pending
  | Succcess(data)
  | Error(data)

let initialState = {
  loading: false,
  data: NotAsked,
  ...
}
```

Gradual Adoption

State Management First

Gradual Adoption

UI First

State Management Gradual Adoption POC

```
let webhooksLogs = (state: t, action) => {
  if(state == None) {
    defaultState
  } else {
    switch(action -> type_) {
    | "webhooksLogs/get"  =>
        state_(~loading=true, ~webhooksLogs=None, ~error=None, ~links=None)
    | "webhooksLogs/error" =>
        state_(~loading=false, ~webhooksLogs=None, ~error=Some(action -> payload), ~links=None)
    | "webhooksLogs/success" =>
        state_(
          ~loading=false,
          ~webhooksLogs=Some(action -> payload -> webhook_logs),
          ~links=Some(action -> payload -> links_),
          ~error=None,
        )
    | _ => state |> resolveState
    }
  }
};
```

State Management Gradual Adoption POC

```
let counter = (state, action) =>
  if(state == None) {
    initialState
  } else {
    switch(action -> tag) {
    | Increment => state_(~counter=state -> counter + 1)
    | Decrement => state_(~counter=state -> counter - 1)
    | _ => state
    }
  }
```

Proposal:

Identify where we can have big returns when we invest in ReasonML. Gradually adopt Reason in Ashitaka.