ReasonML – An Overview

Jeremy Berglund

What is ReasonML?

ML != Machine Learning
ML != Markup Language

A new language

Designed to compile
to JS or replace JS

Purely functional

What is ReasonML?

- **Initiated at Facebook (creator of React)**
- **Makes strongly-typed functional programming accessible**
  - Familiar syntax
  - Incremental adoption
- **Strong type system**
  - Structural type system – 100% type coverage, type inference
  - Completely sound types – types are guaranteed to be accurate after compilation
- **Performance**
  - Dead code elimination
  - Compile-time optimizations
- **Cross platform**

Why OCaml?

- Rich history
- Reason can take advantage of existing OCaml tools
- Compile to native
- OCaml defaults to immutable / functional paradigms
- Popular in academia and with banking institutions
- Strong types
  - Immediate feedback
  - Intellisense
  - Refactoring

Types

- **Type inference**
- **Structural typing**
  - The structure of the type matters, not the name
  - TypeScript's structural typing
    - Possibility of "undefined" and "null" values leads to faulty typing
    - "any" types
    - Doesn't play well with function composition
- **100% type coverage**
- **Completely "sound" types**
  - Once the code is compiled, its guaranteed to have no type errors
  - No null pointer exceptions ever!

Installation

```
npm i -g bs-platform
```

let bindings

```
let hello_world = "Hello world";

let greet = (greeting: string) => {
    print_endline(greeting);
};

greet(hello_world);
```

Lists and Arrays

```
let immutable_arr = ["😀", "🔥", "👍"];
let head = List.hd(immutable_arr);

let mutable_arr = [|"😀", "🔥", "👍"|];
mutable_arr[0] = "😯";
```

Records

```
type person = {
  first: string,
  last: string,
  age: int,
};

let record = {first: "Obi-Wan", last: "Kenobi", age: 38};
```

Functions

```
let add = (x) ⇒ (y) ⇒ (z) ⇒ x + y + z;

let add1 = add(1);
let result = add1(2)(3);
```

Functions

```
let add = (~x, ~y, ~z) ⇒ x + y + z;

let curriedAdd = add(~y=10, ~z=5);
let result = curriedAdd(2); /* 17 */
```

Pattern Matching

```
let fizzbuzz = i ⇒
  switch (i mod 3, i mod 5) {
  | (0, 0) ⇒ "FizzBuzz"
  | (0, _) ⇒ "Fizz"
  | (_, 0) ⇒ "Buzz"
  | _ ⇒ string_of_int(i)
  };

for(i in 1 to 100) {
    print_string(fizzbuzz(i) ++ " ");
}
/* 1  2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz
   11 Fizz 13 14 FizzBuzz 16 17 ... */
```

Variant Types

```
/* Variant Types */
type social_media_account =
  | None
  | Facebook(string, string)
  | Twitter(string, string, string);

let print_account_details =
  fun
  | Facebook(first, last) ⇒ print_endline(first ++ last)
  | Twitter(first, last, handle) ⇒ print_endline(first ++ last ++ "(" ++ handle ++ ")")
  | None ⇒ print_endline("Not logged in");

let my_account = Twitter("Jeremy", "Berglund", "@jeremyberglund");
print_account_details(my_account); /* Jeremy Berglund (@jeremyberglund) */

print_account_details(None); /* "Not logged in" */
```

Piping

```
let isEven = i ⇒ i mod 2 == 0;

let mul = (a, b) ⇒ a * b;
let double = mul(2);

let arr = [6, 2, 1, 4, 8]
    |> List.filter(isEven) /* [6, 2, 4, 8] */
    |> List.map(double) /* [12, 4, 8, 16] */
    |> List.map(string_of_int) /* ["12", "4", "8", "16"] */
```

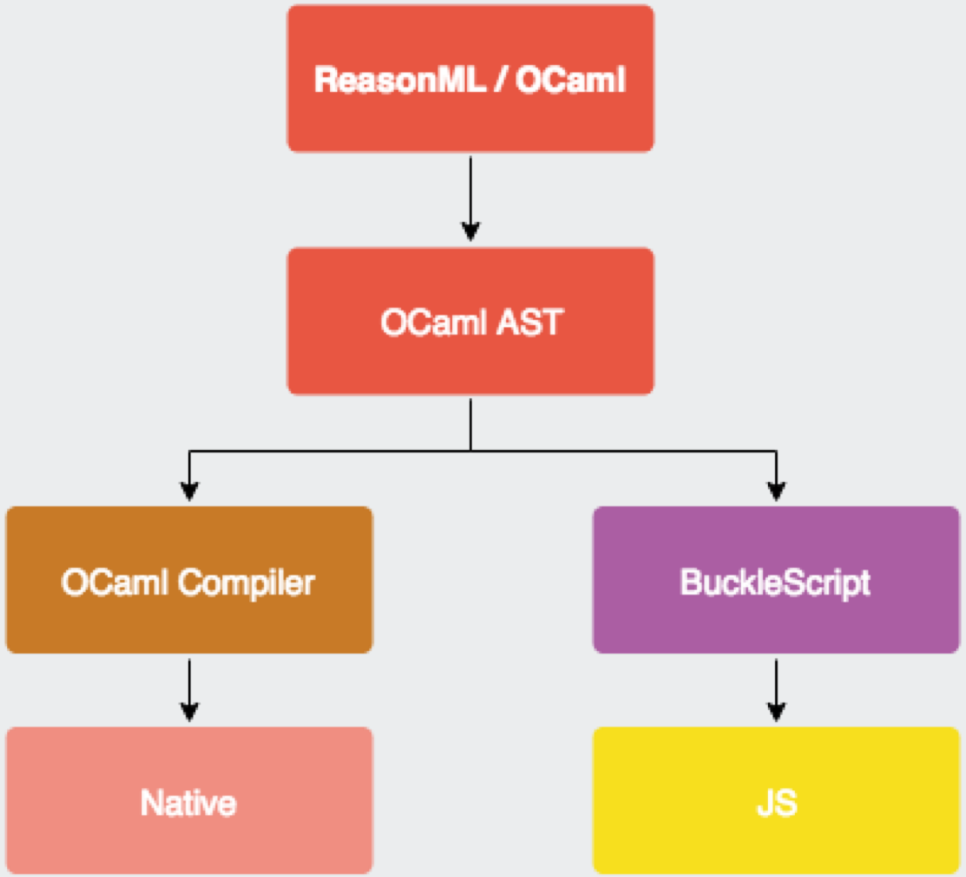
Piping

```
let mul = (a, b) ⇒ a * b;
let double = mul(2);
let repeat = str ⇒ str ++ str;

let nonsense_fn = arg ⇒
 print_string(repeat(string_of_int(double(arg))));

let better_nonsense_fn = arg ⇒ arg
  |> double
  |> string_of_int
  |> repeat
  |> print_string;
```

BuckleScript

- Developed by Bloomberg
- Takes the OCaml AST and outputs readable and performant JS

BuckleScript



Bucklescript

- Developed by Bloomberg
- Takes the OCaml AST and outputs readable and performant JS
- Dead code elimination
- Compile time optimizations
- Small footprint

Avg. Node Startup	Bucklescript Startup
100ms	2ms

Bucklescript Performance

```
let test = () ⇒ {
  let m = ref(IntMap.empty);
  let count = 1000000;

  for(i in 0 to count) {
    m := IntMap.add(i, i, m^);
  };
  for(i in 0 to count) {
    ignore(IntMap.find, i, m^)
  };
};
```

Bucklescript Performance

```
function test() {
    var m = /* Empty */0;
    for(var i = 0; i⩽ 1000000; ++i){
      m = add(i, i, m);
    }
    for(var i$1 = 0; i$1⩽ 1000000; ++i$1){
      find(i$1, m);
    }
    return /* () */0;
  }

  test(/* () */0);
```

900 Bytes - ~1100ms execution time

Bucklescript Performance

```
'use strict'
var Immutable = require('immutable')
var Map = Immutable.Map
var m = new Map()
var test = function() {
  var count = 1000000
  for (var i = 0; i < count; ++i) {
    m = m.set(i, i)
  }
  for (var j = 0; j < count; ++j) {
    m.get(j)
  }
}

test()
```

55KB - ~3500ms execution time

BuckleScript

- Developed by Bloomberg
- Takes the OCaml AST and outputs readable and performant JS
- Dead code elimination
- Compile time optimizations
- Small footprint
- Interop with existing JS code
  - Reuse code
  - Incremental adoption

JavaScript Interop

```
/* raw javascript */
[%bs.raw {|
    const add = (a, b) ⇒ a + b;
|}];

/* Importing JS packages */
[@bs.val] external leftPad : (string, int, option(int)) ⇒ string = "leftpad";

leftPad("foo", 6);
```

BuckleScript

- Developed by Bloomberg
- Takes the OCaml AST and outputs readable and performant JS
- Dead code elimination
- Compile time optimizations
- Small footprint
- Interop with existing JS code
  - Reuse code
  - Incremental adoption

ReasonReact

- Write React in ReasonML
- Bindings around the JS React library
- Complete types
- Pure functions for life cycle methods

Case study: Facebook's Messenger.com

- **1.5 billion users**
- **Incremental adoption**
- **75%+ converted to ReasonML**
- **Build times drastically decreased**
  - Incremental builds: <100ms
  - Full build: 2s
- **10 bugs in a year**
- **Refactoring time went from days to minutes**

Links

- Reason https://reasonml.github.io/
- BuckleScript https://bucklescript.github.io/en/
- REPL https://staging.rtop.khoanguyen.me/
- Reason Language Server https://github.com/jaredly/reason-language-server
- Reprocessing https://github.com/Schmavery/reprocessing

ReasonML – An Overview