REASONML

What is it and why should we care?

JOE GRAHAM

@josgraha

github.com/josgraha/js-nyc-reason-talk

REASON - SYNTAX

OCAML - SEMANTICS

Syntax and Toolchain for OCAML

REASON – WHAT IS IT?

- New syntax for Ocaml
- Compiler and workflow
- Docs, libs, utils
- Npm friendly
- ES/JS Friendly syntax

HOW DOES IT WORK?

Syntax

Reason - ES6/ES-Next friendly, familiar syntax

Semantics

OCaml provides a typesafe OO/FP (meta) language

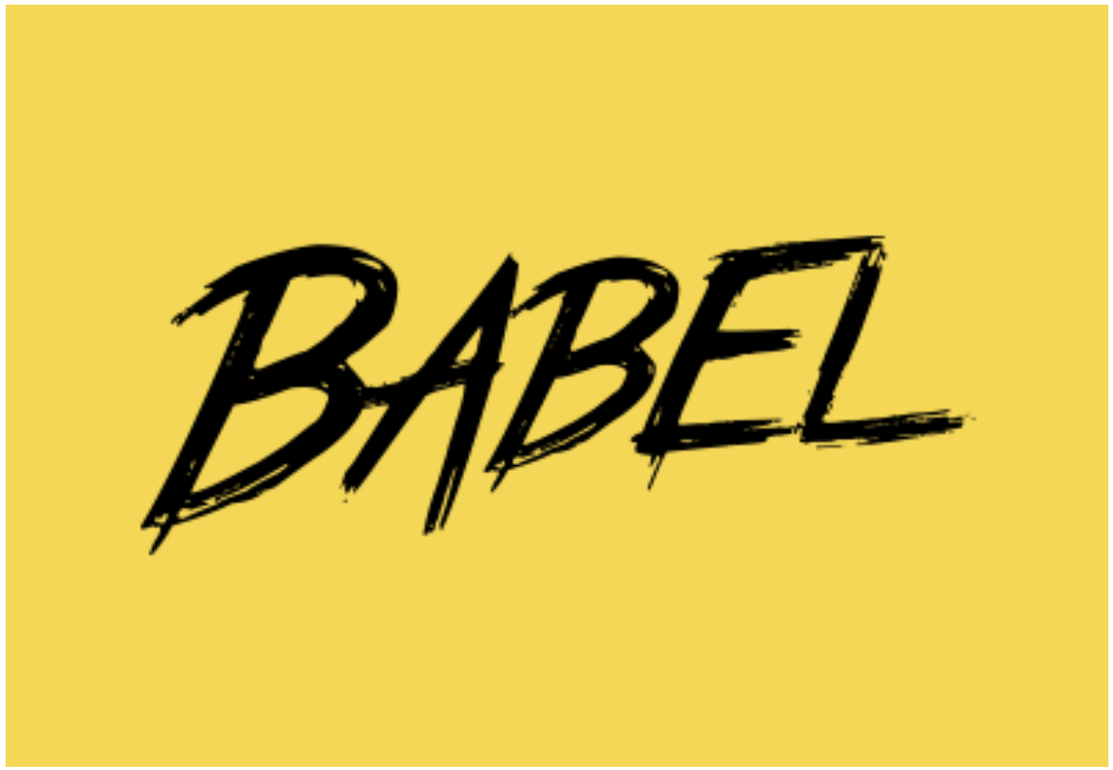
Compiler

Bucklescript compiles Ocaml/Reason to JavaScript

WHY? AN EXAMPLE TOOLCHAIN

FLOW – ADDS TYPES

- OCaml is a typed language
- Flow is written in Ocaml (cheat!)
- Type lookups
- Type inference (annotate when necessary)
- Refactoring… what?



BABEL – ES TO JS COMPILER

- Bucklescript 10x faster than TypeScript
- Human readable JS output
- Can compile to bytecode (native) - not just JS

PRETTIER - FORMAT

- Taken from reason-fmt (cheat!)
- Reason-Fmt has fewer config options
- Eject long legacy of craft

ESLINT – NO NAUGHTY STUFF

- Compiler!
- Type safety
- Type infererence
- Monads … (IO, Maybe, Either)
- Deprecation
- Reason tries to tame the MetaLanguage
  (see Cheng Lou talk)

WHAT DOES IT HAVE TO OFFER?

- Destruction of NULL
- History
- Community
- Solid type system (30+ years of compiler research)
- Incremental conversion
- Performance and Size (compiler does crazy optimization)
- Vastly improved bundle compaction
- Features not even on ES-Next-Next radar
- Simpler toolchain (bsb to js then webpack?) …

REASON VS JS TOOLCHAIN

- All of the tools get “absorbed” into the toolchain
- Example: reason app

```
{  The name of the package.
  "name": "reason-talk-demo",
  "version": "0.1.0",
  "scripts": {
    "build": "bsb -make-world",
    "start": "bsb -make-world -w",
    "clean": "bsb -clean-world"
  },
  "keywords": [
    "BuckleScript"
  ],
  "author": "",
  "license": "MIT",
  "devDependencies": {
    "bs-platform": "^2.1.0"
  }
}
```

REASON VS REACT TOOLCHAIN

- Add reason-react to deps
- CRA - reason scripts
- Example: reason-react

```
{
  "name": "reason-talk-react-demo",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.2.0",
    "react-dom": "^16.2.0",
    "reason-scripts": "0.8.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject",
    "prepare": "npm link bs-platform"
  },
  "devDependencies": {
    "bs-jest": "^0.2.0",
    "reason-react": "^0.3.0"
  }
}
```

REASON – BASICS

PRIMITIVES

Strings	`"Hello"`
Characters	`x'`
Integers	`23 , -23`
Floats	`23.0 , -23.0`
Arrays	`[|1, 2, 3|]`
Records	`type player = {score: int}; {score: 100}`
Comments	`/* Comment here */`

MATH

Integer Division/Multiplication	`2 / 23 * 1`
Float Division/Multiplication	`2.0 /. 23.0 *. 1.0`
Float Exponentiation	`2.0 ** 2.0`

OPERATIONS

Immutable Lists	`[1, 2, 3]`
Immutable Prepend	`[item1, item2, ...theRest]`
String Concatenation	`"Hello " ++ "World"`
Comparison	`> , < , >= , =<`
Boolean operations	`! , && , ||`
Reference, Physical (deep) Equality	`=== , ==`

RECORDS

```
/* Records */
/* Type: mandatory */
type person = {age: int, name: string};
/* values */
let me: School.person = {age: 20, name: "Big Reason"};
/* or */
let me = School.{age: 20, name: "Big Reason"};
/* or */
let me = {School.age: 20, name: "Big Reason"};
```

DESTRUCTURING

```
/* Destructuring */
type person = {name: string, age: int};
let somePerson = {name: "Guy", age: 30};
let {name, age} = somePerson;
```

TUPLES

```
/* tuples */
type coord3d = (float, float, float);
let my3dCoordinates: coord3d = (20.0, 30.5, 100.0);
let (_, y, _) = my3dCoordinates; /* now you've retrieved y */
```

VARIANTS

```
/* Variants */
type myResponseVariant =
  | Yes
  | No
  | PrettyMuch;

let areYouCrushingIt = Yes;
```

PATTERN MATCHING

```
/* Pattern Matching */
type payload =
  | BadResult(int)
  | GoodResult(string)
  | NoResult;

let data = GoodResult("Product shipped!");

let message =
  switch data {
  | GoodResult(theMessage) => "Success! " ++ theMessage
  | BadResult(errorCode) => "Something's wrong. The error
  code is: " ++ string_of_int(errorCode)
  };
/*
Warning 8: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
NoResult
*/
```

FUNCTIONS

```
/* Functions */
let add = (x, y) => x + y;
Js.log(add(3, 2)); /* 5 */
/* Functions are curried by default */
let add1 = add(1);
Js.log(add1(2)); /* 3 */

/* Use rec to expose recursive function definition */
let rec sum = xs => switch xs {
  | [] => 0
  | [x, ...xs] => x + sum(xs)
};
let oneToFive = [1, 2, 3, 4, 5];
Js.log(sum(oneToFive)); /* 15 */
```

EXAMPLE: QUICKSORT

```
/* Quicksort */
let rec quicksort = (gt) =>
fun
| [] => []
| [x, ...xs] => {
    let (ys, zs) = List.partition(gt(x), xs);
    quicksort(gt, ys) @ [x, ...quicksort(gt, zs)]
  };
[4, 65, 2, (-31), 0, 99, 83, 782, 1] |> quicksort((>)) |>
Array.of_list |> Js.log;
/* [-31,0,1,2,4,65,83,99,782] */
```

REASON VS . . .

QUESTIONS