Introduction to ReasonML

Ryan Watkins

July 31, 2018

Javascript so far

Javascript so far

Features

- ES6 (Babel + Webpack)

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals
- Destructuring (single-branch pattern-matching)

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals
- Destructuring (single-branch pattern-matching)
- Promises + (async/await)

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals
- Destructuring (single-branch pattern-matching)
- Promises + (async/await)
- Spread operator

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals
- Destructuring (single-branch pattern-matching)
- Promises + (async/await)
- Spread operator
- Get/set on class definitions (some OOP)

Problems

- No great pattern matching

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals
- Destructuring (single-branch pattern-matching)
- Promises + (async/await)
- Spread operator
- Get/set on class definitions (some OOP)

Problems

- No great pattern matching
- Non staticly typed, i.e. you get typed errors at runtime

Javascript so far

Features

- ES6 (Babel + Webpack)
- Template literals
- Destructuring (single-branch pattern-matching)
- Promises + (async/await)
- Spread operator
- Get/set on class definitions (some OOP)

Problems

- No great pattern matching
- Non staticly typed, i.e. you get typed errors at runtime
- Module system not great, you have to take care of it yourself

What is ReasonML?

What is ReasonML?

- It's Ocaml with JS-like syntax

What is ReasonML?

- It's Ocaml with JS-like syntax
- Contains pattern-matching

What is ReasonML?

- It's Ocaml with JS-like syntax
- Contains pattern-matching
- Several FP features (currying by default)

What is ReasonML?

- It's Ocaml with JS-like syntax
- Contains pattern-matching
- Several FP features (currying by default)
- Solid type system (30+ years of research)

What is ReasonML?

- It's Ocaml with JS-like syntax
- Contains pattern-matching
- Several FP features (currying by default)
- Solid type system (30+ years of research)
- Extremely fast compilation to JS (10x faster than TypeScript)

What is ReasonML?

- It's Ocaml with JS-like syntax
- Contains pattern-matching
- Several FP features (currying by default)
- Solid type system (30+ years of research)
- Extremely fast compilation to JS (10x faster than TypeScript)

Ocaml to JS?

Ocaml to JS?

- Made possible by bucklescript

Ocaml to JS?

- Made possible by bucklescript
- Human readable output

Ocaml to JS?

- Made possible by bucklescript
- Human readable output
- Ocaml to Native and JS

Reason Primitives

Reason Primitives

Primitive	Example
Strings	‘test‘
Characters	’c’
Integers	23, -23
Floats	23.0, -23.0

Pattern Matching

Pattern Matching

Javascript already has single-branch pattern matching
I.e. const foo, bar, baz = props

Comparing Reason/Bucklescript to Elm

Comparing Reason/Bucklescript to Elm

- From the FAQ (paraphrased)

Comparing Reason/Bucklescript to Elm

- From the FAQ (paraphrased)
- Reason is a general-purpose language that can target node, native code and utilize both the opam and npm ecossytems. Elm is more opininated and focused. Elm has better error messages. BuckleScript has superb js interop, generates highly readable and easily debuggable js code and can utilize the npm ecosystem to its full extent, while Elm's FFI and interop is rather convoluted. Elm is pure, while Reason is pragmatic. OCaml has had 25 or so years to mature, and has an active academic base of contributors that keep it close to the forefront of progamming langauge development. Reason has first-class support for React, while Elm is focused solely on "The Elm Architecture" (TEA).

Comparing Reason/Bucklescript to Elm

- From the FAQ (paraphrased)
- Reason is a general-purpose language that can target node, native code and utilize both the opam and npm ecossytems. Elm is more opininated and focused. Elm has better error messages. BuckleScript has superb js interop, generates highly readable and easily debuggable js code and can utilize the npm ecosystem to its full extent, while Elm's FFI and interop is rather convoluted. Elm is pure, while Reason is pragmatic. OCaml has had 25 or so years to mature, and has an active academic base of contributors that keep it close to the forefront of progamming langauge development. Reason has first-class support for React, while Elm is focused solely on "The Elm Architecture" (TEA).

Potential and PL Theory

Last statement regarding the potential of PL theory