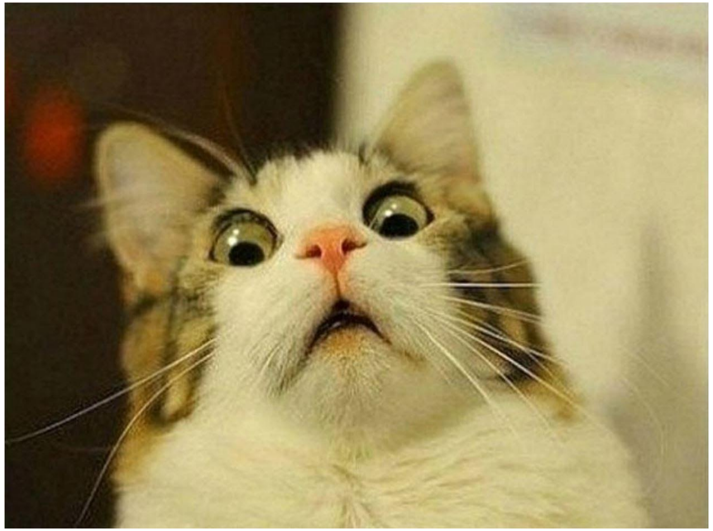
REASON

SYNTAX & TOOLCHAIN FOR OCAML

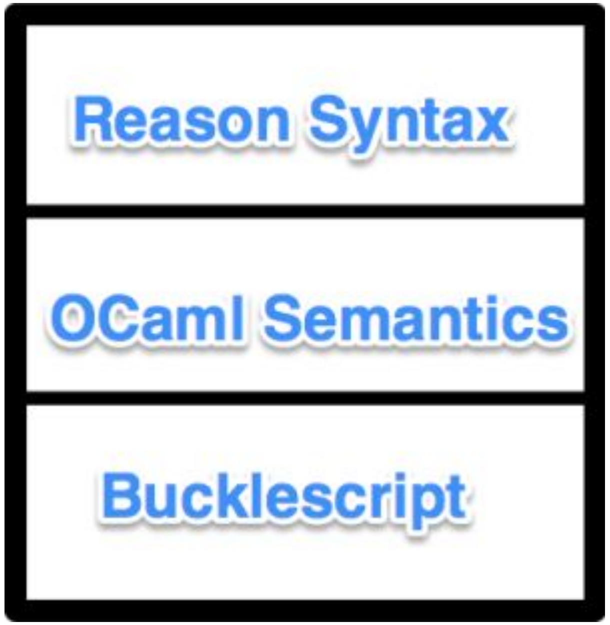
ReasonML

- Started at Facebook by creator of React.js
- First Github commit in Feb 2016
- 50% Messenger (web) in Reason



What is ReasonML?

- JS-like syntax
- OCaml inside
- JS code generator (Bucklescript)
- JS toolchain (npm, webpack)



ReasonML playground

Why ReasonML and OCaml ?

"Learning ReasonML is similar to learning JS + a gradual type system"

https://reasonml.github.io/docs/en/what-and-why.html

Benefits:

- OCaml type system
- pragmatism: opt-in side-effects, mutation and object for familiarity
- A focus on performance & size
- Great ecosystem & tooling (Use your favorite editor, your favorite NPM package, …)

OCaml

- ML family (SML, Haskell, Elm)
- ~20 years old
- ML + Object system
- Industry users
- CS courses

Jane Street

CITRIX®

docker

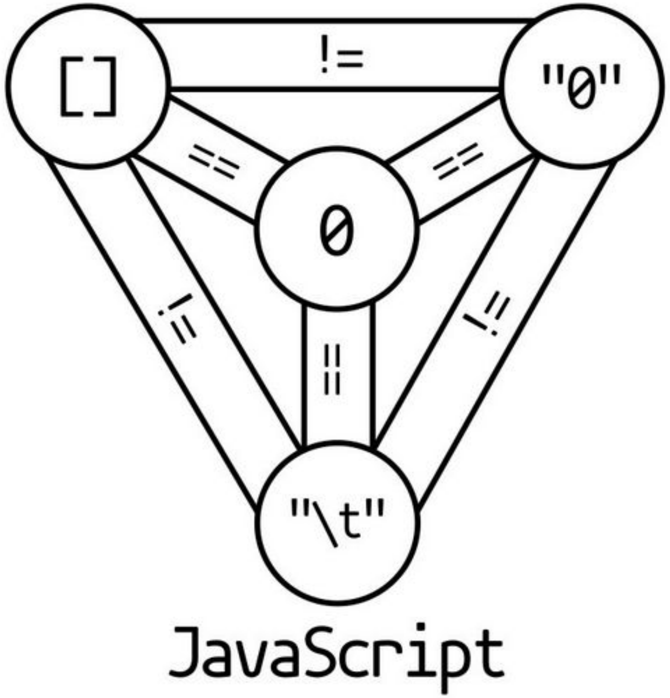
facebook

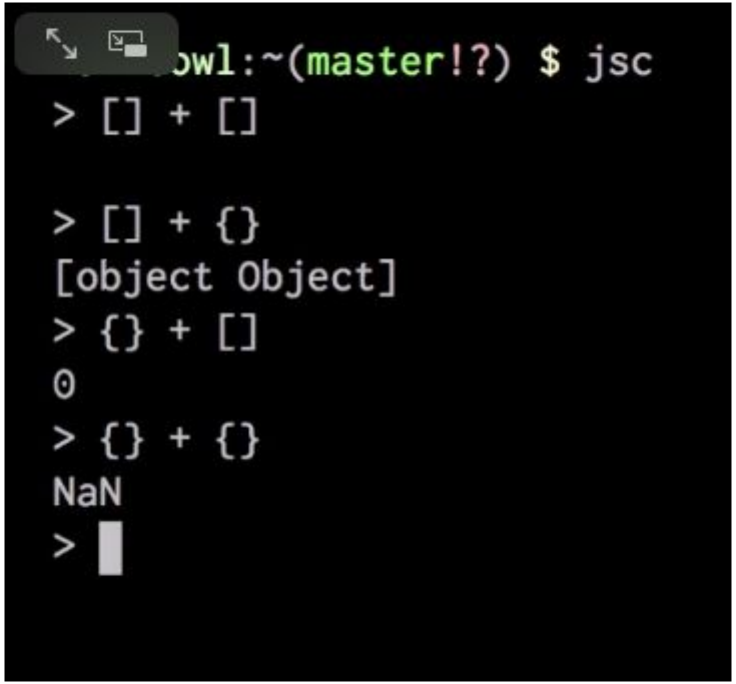
http://ocaml.org/learn/description.html
http://ocaml.org/learn/companies.html

Why not OCaml ?

- Traditional ocaml niche is *compilers* & static analysis, not web apps
  - "OCaml's data types, pattern matching, [...] makes it really nice for doing the tree traversals"
- Reason syntax and tools designed to appeal to JS devs
- More devs: more doc, libraries, resources...

Why not JS?

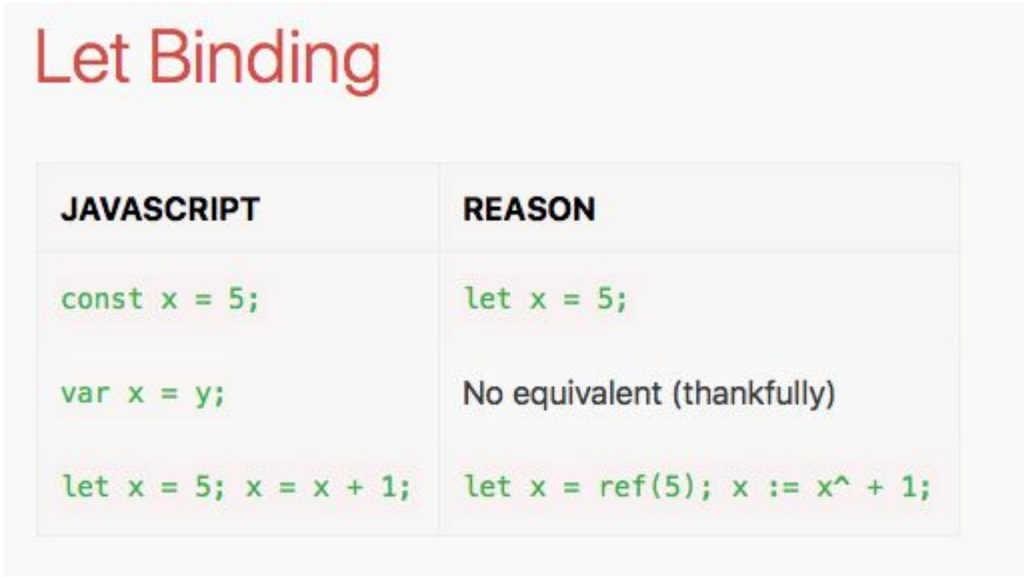




```
owl:~(master!?) $ jsc
> [] + []

> [] + {}
[object Object]
> {} + []
0
> {} + {}
NaN
> 
```

https://www.destroyallsoftware.com/talks/wat

ReasonML Syntax

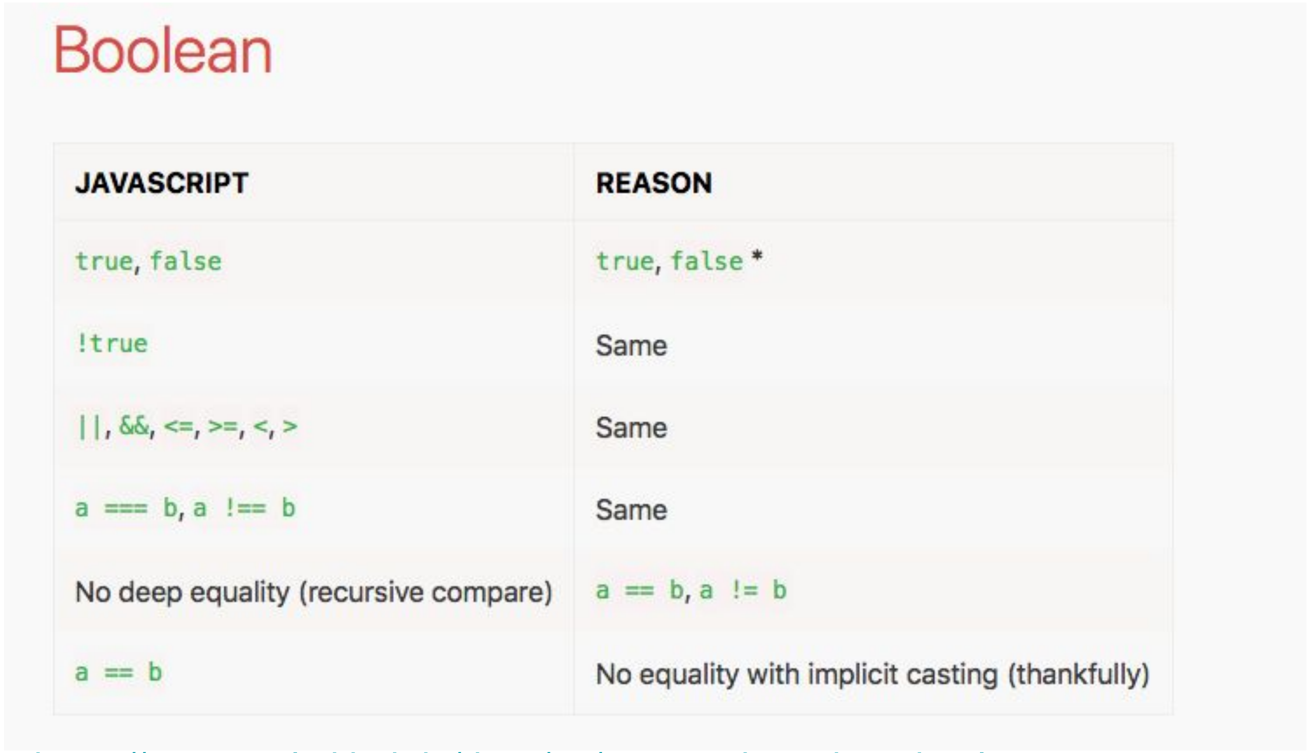


Let Binding

JAVASCRIPT	REASON
`const x = 5;`	`let x = 5;`
`var x = y;`	No equivalent (thankfully)
`let x = 5; x = x + 1;`	`let x = ref(5); x := x^ + 1;`

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax

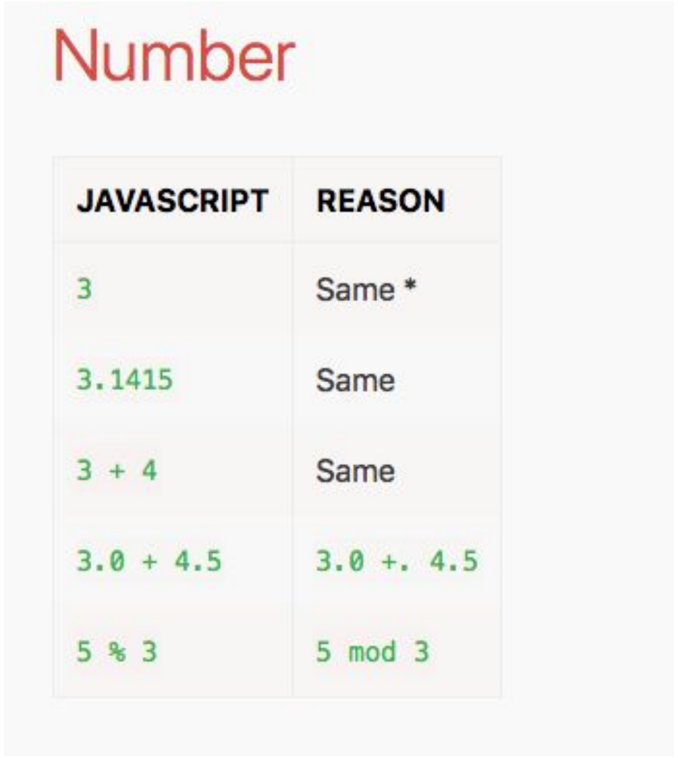


Boolean

JAVASCRIPT	REASON
`true`, `false`	`true`, `false` *
`!true`	Same
`||`, `&&`, `<=`, `>=`, `<`, `>`	Same
`a === b`, `a !== b`	Same
No deep equality (recursive compare)	`a == b`, `a != b`
`a == b`	No equality with implicit casting (thankfully)

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax



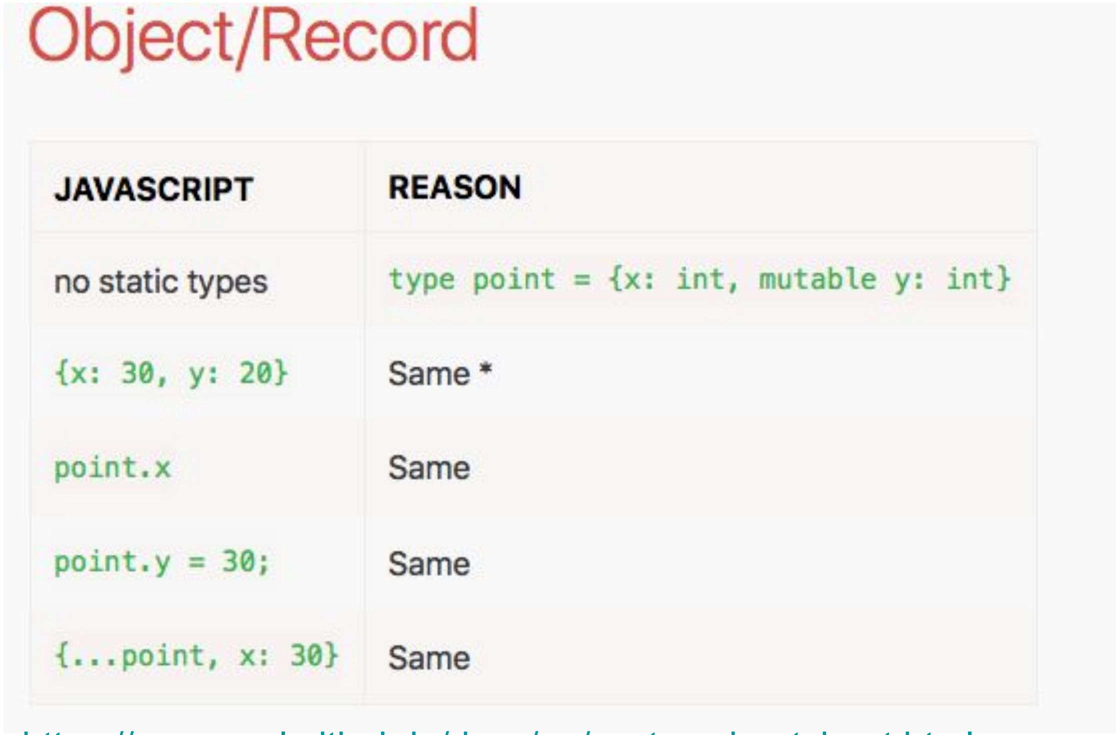
Number

JAVASCRIPT	REASON
3	Same *
3.1415	Same
3 + 4	Same
3.0 + 4.5	3.0 +. 4.5
5 % 3	5 mod 3

```
module FO = {
  let (+) = (+.);
  let (-) = (-.);
  let (*) = (*.);
  let (/) = (/.);
};


… FO.(3.0 + 4.5) …
```

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax

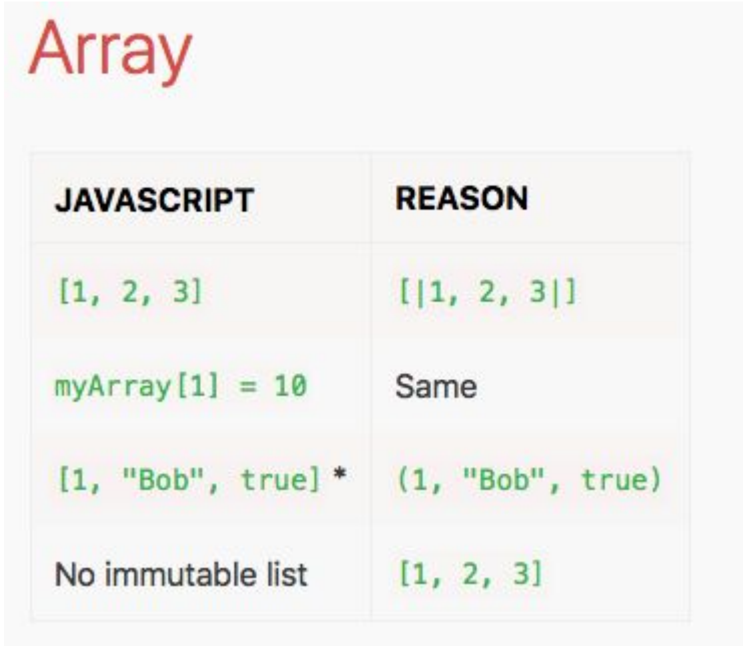


Object/Record

JAVASCRIPT	REASON
no static types	`type point = {x: int, mutable y: int}`
`{x: 30, y: 20}`	Same *
`point.x`	Same
`point.y = 30;`	Same
`{...point, x: 30}`	Same

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax

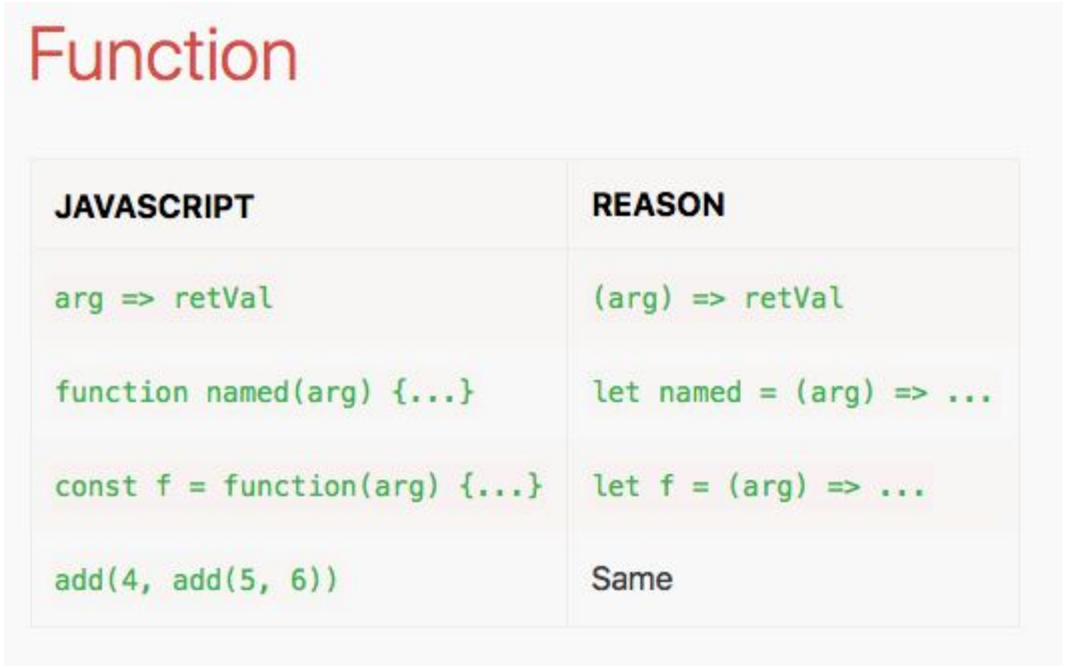


Array

JAVASCRIPT	REASON
`[1, 2, 3]`	`[|1, 2, 3|]`
`myArray[1] = 10`	Same
`[1, "Bob", true]` *	`(1, "Bob", true)`
No immutable list	`[1, 2, 3]`

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax

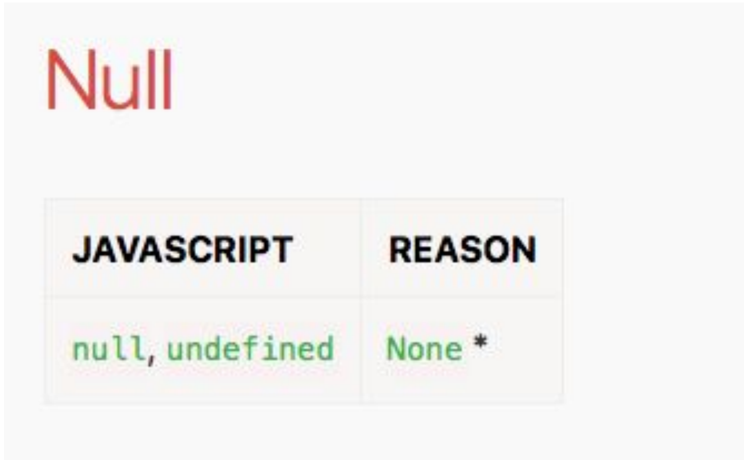


Function

JAVASCRIPT	REASON
`arg => retVal`	`(arg) => retVal`
`function named(arg) {...}`	`let named = (arg) => ...`
`const f = function(arg) {...}`	`let f = (arg) => ...`
`add(4, add(5, 6))`	Same

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax

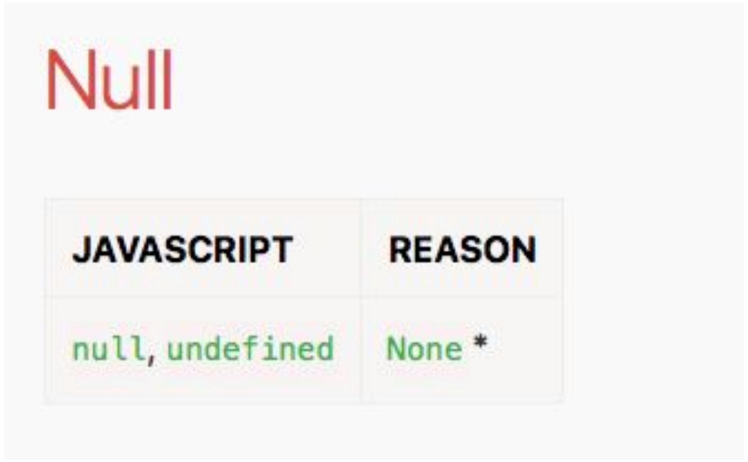


Null

JAVASCRIPT	REASON
`null`, `undefined`	`None` *

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

ReasonML Syntax



Null

JAVASCRIPT	REASON
null, undefined	None *

```
type option('a) = None | Some('a);
let x = Some(5); /* x : option(int) */

switch x {
 | None => -1
 | Some(v) => /* v == 5 */
};
```

https://reasonml.github.io/docs/en/syntax-cheatsheet.html

BuckleScript

- Readable output
- Compact, Fast (build time and run time)
- Comprehensive interop (FFI)
- Stdlib
- Build System

https://bucklescript.github.io/docs/en/what-why.html
https://bucklescript.github.io/docs/en/compiler-architecture-principles.html

BuckleScript FFI

```
[%%bs.raw {|
console.log('here is some js for you');
|}];
```

```
[%%bs.raw {|
console.log('here is some js for you');
|}];
```

```
let x: string = [%bs.raw {| 'well-typed' |}];
```

```
var x = ( 'well-typed' );
```

```
[@bs.val] external alert : string => unit =
"alert";
alert("hello");
```

```
alert('hello');
```

```
[@bs.send] external fillRect : (context, float,
float, float, float) => unit = "";
fillRect(ctx, 0.0, 0.0, 100.0, 100.0);
```

```
ctx.fillRect(0.0, 0.0, 100.0, 100.0);
```

https://reasonml.github.io/docs/en/interop.html

Bucklescript Stdlib

A rewrite of the OCaml stdlib:

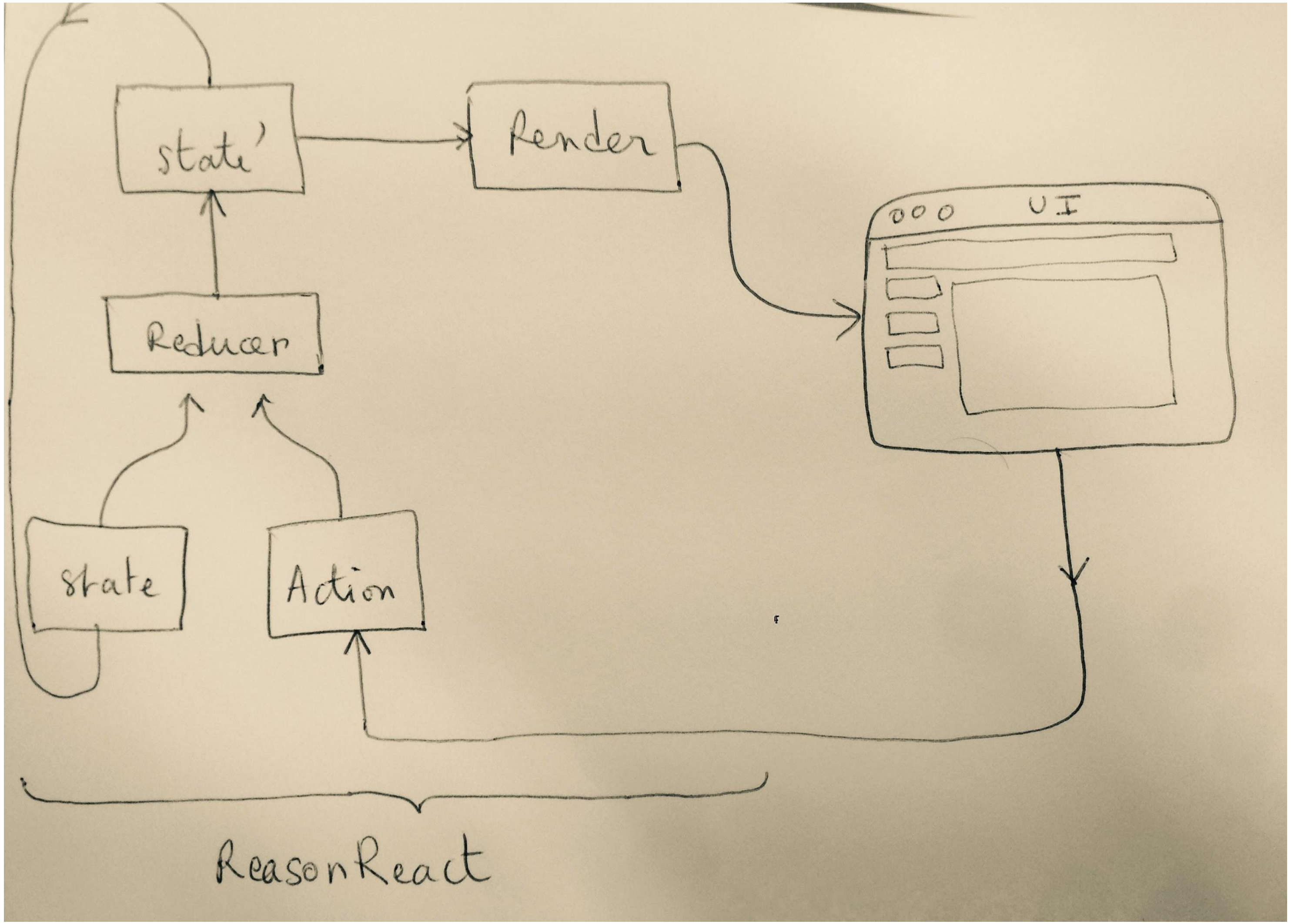
1. Consistent names and arg order
2. Functions suffixed with Exn
3. Better performance and smaller code size
4. More data structures

https://bucklescript.github.io/bucklescript/api/Belt.html

ReasonReact

- Virtual Dom: batch DOM updates
- Component programming: code reuse
- ReasonML bindings to React.js

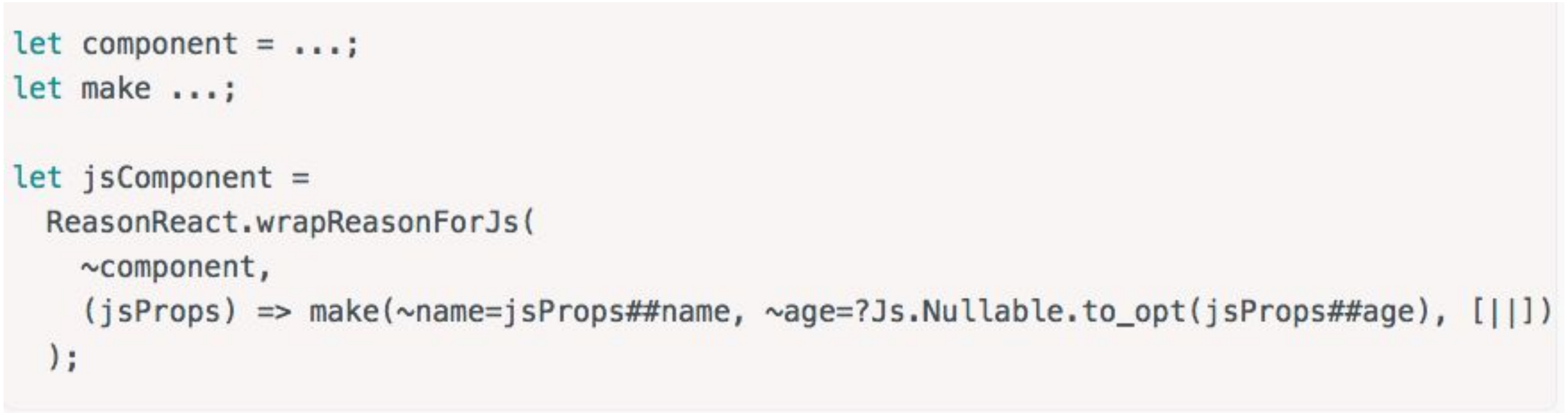
https://reasonml.github.io/reason-react/


state'
Render
UI
Reducer
state
Action
ReasonReact

Elm vs ReasonReact

Elm	**ReasonReact**
model	state
view(), div, css	render(), <div>, ?
update()	reducer()

Using a Reason component with React.js



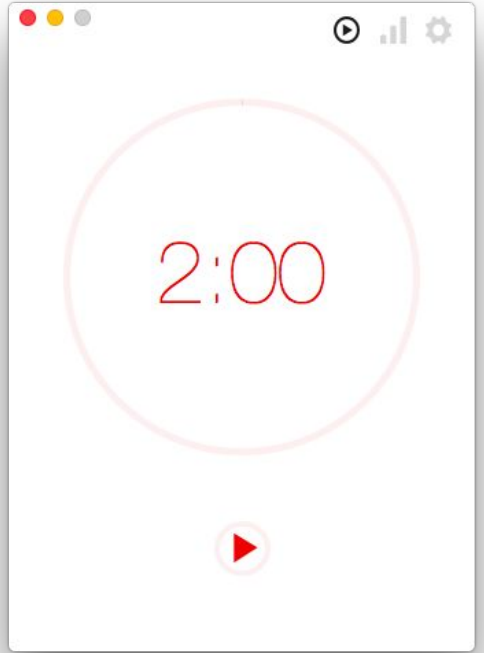
```
let component = ...;
let make ...;

let jsComponent =
  ReasonReact.wrapReasonForJs(
    ~component,
    (jsProps) => make(~name=jsProps##name, ~age=?Js.Nullable.to_opt(jsProps##age), [||])
  );
```

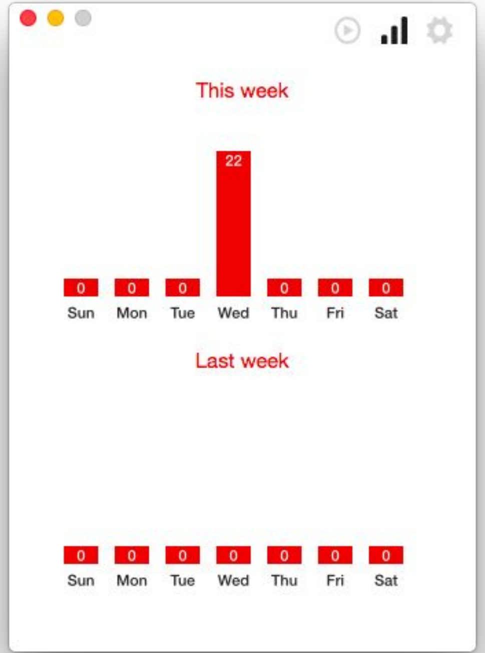
```
var MyReasonComponent = require('./myReasonComponent.bs').jsComponent;

<MyReasonComponent name="John" />
```

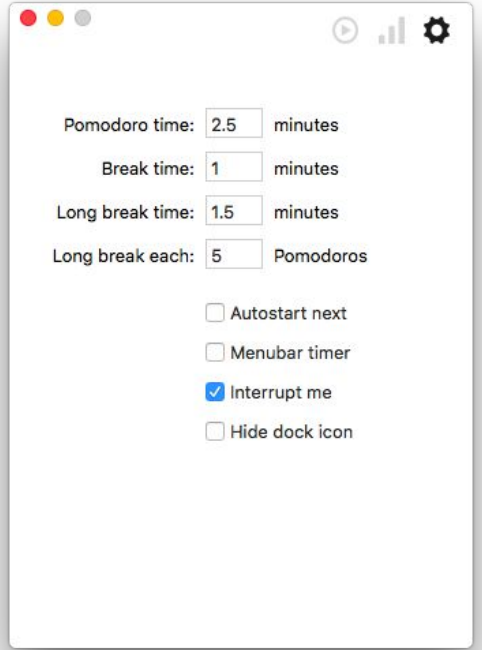
Using a JS component with ReasonReact

```
[@bs.module] external myJSReactClass : ReasonReact.reactClass = "./myJSReactClass";

let make = (~name: string, ~age: option(int)=?, children) =>
  ReasonReact.wrapJsForReason(
    ~reactClass=myJSReactClass,
    ~props={"name": name, "age": Js.Nullable.fromOption(age)},
    children
  );
```

Pomodoro







Pomodoro time:	2.5	minutes
Break time:	1	minutes
Long break time:	1.5	minutes
Long break each:	5	Pomodoros

- [ ] Autostart next
- [ ] Menubar timer
- [x] Interrupt me
- [ ] Hide dock icon

State & Actions

```
/* React Component state */
type state = {
  screen,
  status, /* timer state */
  };
```

```
/* different action values */
type action =
  | Start
  | Stop
  | Tick
  | Screen(screen);

/* different screen values */
type screen =
  | Timer
  | Week
  | Pref;
```

Representing the timer state

```
/* different sort of timers */

type timerType =

  | Work

  | Break;
```

```
/* current status of the timer

 1.   Are we Stopped or Ticking
 2.   If we are Ticking:
       a.   are we Working or having
            a Break?
       b.   How much time is left ?

*/

type status = ?
```

Representing the timer state

```
/* current status of the timer

 1.   Are we Stopped or Ticking
 2.   If we are Ticking:
       a.   are we Working or having a Break?
       b.   How much time is left ?

*/

type status = {

  stopped: bool,

  timerType: timerType,

  timeLeft: int /* seconds */

};
```

Representing the timer state

```
/* current status of the timer

 1.   Are we Stopped or Ticking
 2.   If we are Ticking
      a.   are we Working or having a Break?
      b.   How much time is left ?

*/

type status = {

  stopped: bool,

  timerType: timerType,

  timeLeft: int /* seconds */

};
```

```
/* Does this value make sense ? */

let s = {

  stopped: true,

  timerType: Break,

  timeLeft: 0

};
```

A better representation

```
/* current status of the timer */
type status =
  | Ticking(timerType, int)
  | Stopped;
```

```
let s = Stopped;
let s' = Ticking(Work, 25);
let s'' = Ticking(Break, 25);
```

Getting Started

```
$ bsb -init PROJ -theme react
$ cd PROJ
$ npm i

$ npm run start

$ npm run webpack
$ npm run webpack:production
```

Online Resources

- Awesome ReasonML
- Reason website, Discord, Forum, Podcast, Book
- Reason React tutorial
- Reason-react example
- OCaml forum, Real World OCaml