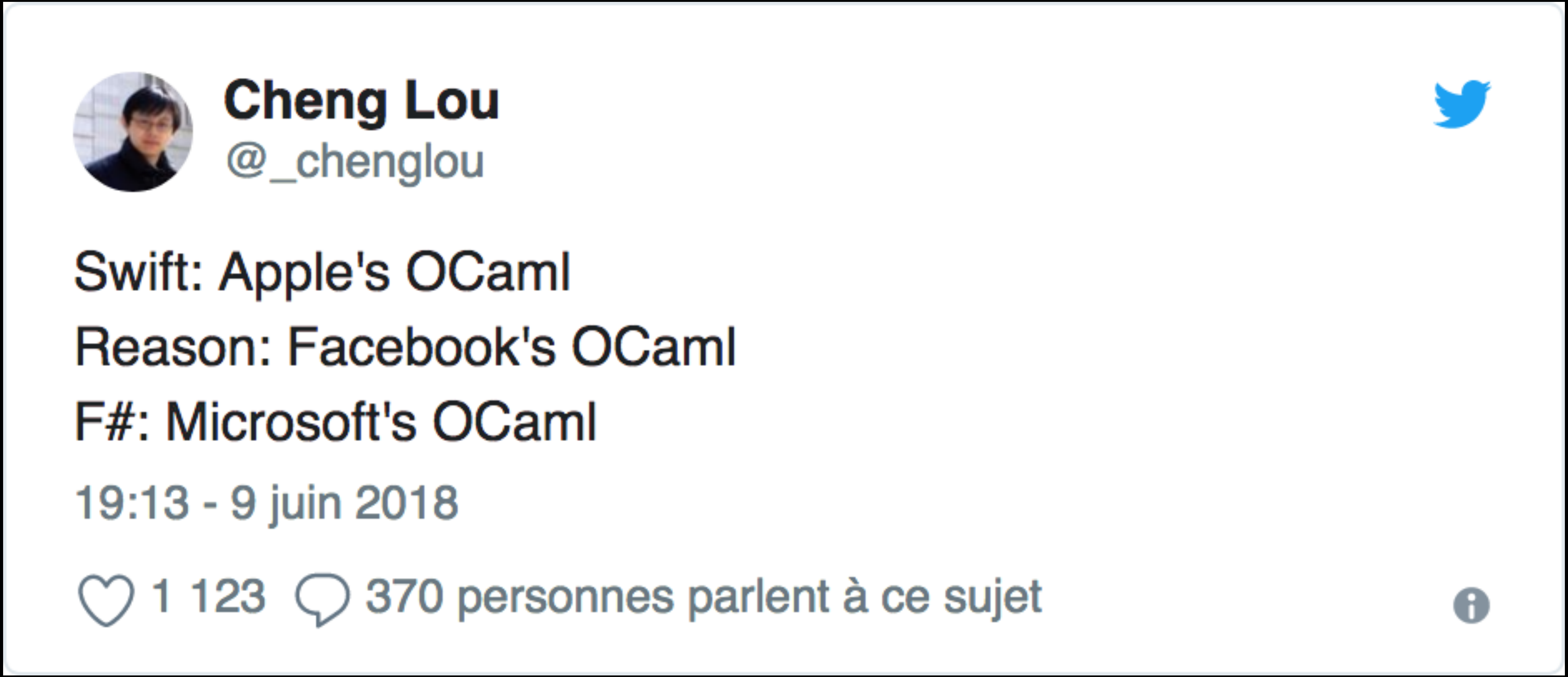
REASON



Cheng Lou
@_chenglou

Swift: Apple's OCaml
Reason: Facebook's OCaml
F#: Microsoft's OCaml

19:13 - 9 juin 2018

♡ 1 123 370 personnes parlent à ce sujet

ReasonML

- OCaml new syntax for JS programmers
- Powerful and safe type inference
- Compile to Javascript
- https://reasonml.github.io/en/try
- https://sketch.sh/

Tuple

- Immutable
- Ordered
- Fix-sized
- Heterogenous

```
let ageAndName: (int, string) = (24, "Lil' Reason");

type coord3d = (float, float, float);
let my3dCoordinates: coord3d = (20.0, 30.5, 100.0);

let (_, y, _) = my3dCoordinates;
```

Record

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

let me = {
  age: 5,
  name: "Big Reason"
};

let name = me.name;

let meNextYear = {...me, age: me.age + 1};
```

- Immutable
- Fixed in field names and types

Module

```
module Cat = {
    type t = string;

    let sleep = (cat: t) =>
      print_endline(cat ++ ": ZZZzzzzZZZ !");
  };

let felix: Cat.t = "Felix";
felix |> Cat.sleep;
```

Function

```
let add = (x, y) => {
    x + y;
};

let five = add(2, 3);
```

Currying

```
let add = (x, y) => x + y;
let add = x => y => x + y;

let add2 = add(2);
let five = add2(3);
```

Pipe & Fast pipe

```
let contains: (string, char) => bool =
    String.contains;
let concat: (string, list(string)) => string =
    String.concat;

let true_ =
  ["a", "b", "c"]
  |> concat("")
  |. contains('b');
```

If-else

```
let message = if (isMorning) {
  "Good morning!"
} else {
  "Hello!"
};
```

Pattern matching

```
switch (isMorning) {
| true => "Good morning!"
| false => "Hello!"
}
```

Pattern matching

```
switch (isMorning) {
| true => "Good morning!"
}
```

```
Warning number 8
OCaml preview

You forgot to handle a possible value here, for example:
false
```

Pattern matching

```
type animal = Dog | Cat | Bird;
let result =
  switch (isBig, myAnimal) {
  | (true, Dog) => 1
  | (true, Cat) => 2
  | (true, Bird) => 3
  | (false, Dog | Cat) => 4
  | (false, Bird) => 5
  }
```

Variant!

```
type maybe =
  | Some(int)
  | Just(string)
  | None;
```

Variant! + Pattern matching

```
type maybe =
  | Some(int)
  | Just(string)
  | None;

let string_of_maybe = maybe =>
  switch (maybe) {
  | None => "None"
  | Some(1) => "Some(1)"
  | Just(x) => "Just(" ++ x ++ ")"
  | Some(x) when x mod 2 == 0 => "Some(%2)"
  | _ => "Some(_)"
  };
```

Null & Undefined

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

let div = (x, y) =>
  switch (y) {
    | 0 => None
    | _ => Some(x / y)
  };

let print = message => {
  print_endline(message);
  ();
};
```

Exceptions

🤬

Operators

```
let eight = (+)(7, 1);

let (+++) = (s, t) => s ++ " " ++ t;

"hello" +++ "world";
```

Next

- BuckleScript
- ReasonReact