Introduction to OCaml

Programming Languages

William Killian

Millersville University

Term Definitions

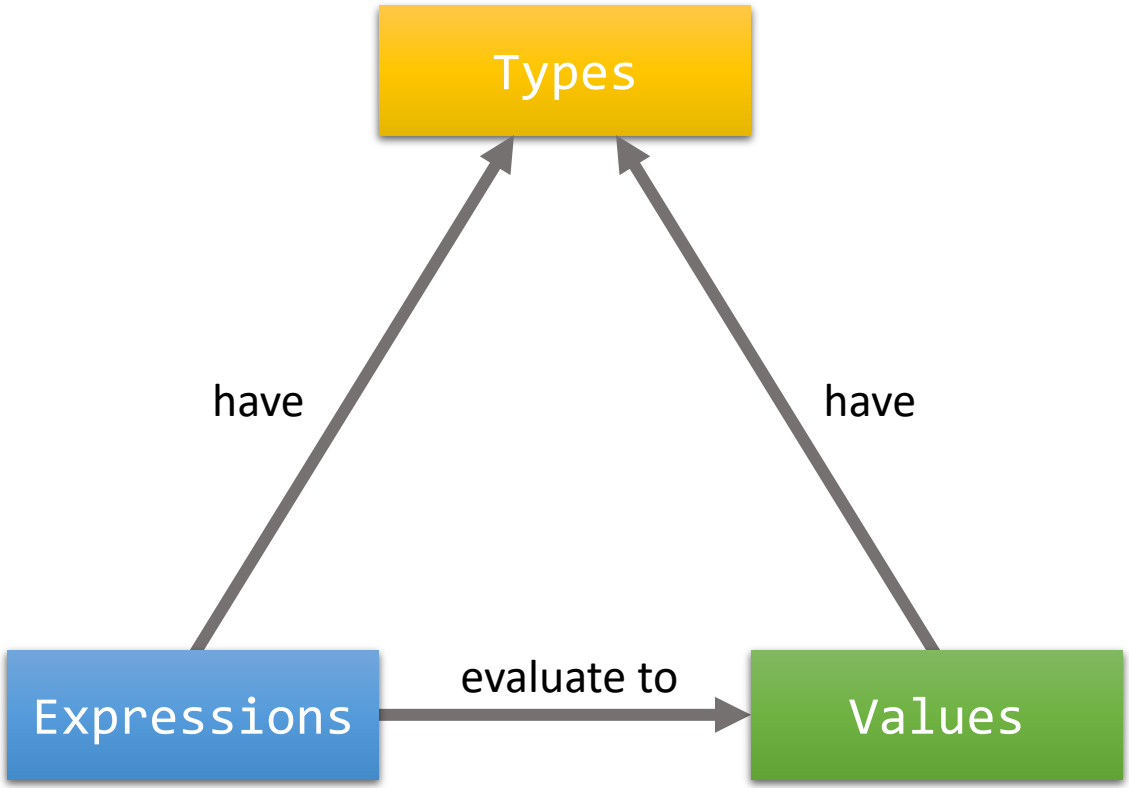
- Type
  - is a Concept
  - manually annotated or inferred
- Value
  - is a Concrete Entity
  - evaluated by the runtime / compiler
- Expression
  - is an Entity
  - what we write
  - what we call “code”

Example Code (Not Ocaml)

```
int x = 3 + 4;
```

- What is the type of:
  - x? 3? 4?
- What is the value of:
  - x? 3? 4?
- Where is the expression?

The OCaml Trinity



OCaml Types

- We will **never** write any types explicitly
- Everything will be *inferred*

```
int        4     (-2)    0
float      4.0
string     "Hello"     "tacocat"
a' list    [ 1; 2; 3 ]   ["a"; "b"; "c" ]
            int list       string list
```

OCaml Expressions (int)

Standard Arithmetic Expressions

```
1 + 2
7 mod 3
2 * (4 + 2)
9 / 4
(-10) - 4
```

OCaml Expressions (float)

Standard Arithmetic Expressions

```
1.0 +. 2.3
2.0 *. (4.66 +. 2.33)
9.0 /. 2.5
(-10.0) -. 4.0
(float_of_int 4)
```

OCaml Expressions (string)

Standard Expressions

```
"h" ^ "ello"
"hello" ^ (string_of_int 4)
"hello" ^ (string_of_float 4.0)
```

OCaml Variables

- CANNOT be reassigned

```
let x = 4;;
let y = x + 1;;

let x = 5;; (* future x's will be 5 *)
```