

Introduction to Functional Programming in *OCaml*

Roberto Di Cosmo, Yann Régis-Gianas, Ralf Treinen

Week 1 - Sequence 3: Definitions



Global Definitions

- ▶ give names to values
- ▶ global: effective for the rest of the toplevel session
- ▶ syntax: `let name = expression`
- ▶ there is no separate declaration of identifiers
- ▶ once set, the value of an identifier never changes
- ▶ once defined, an identifier can be used in expressions

Global Definition Examples I

```
let x = 2+3;;  
# val x : int = 5
```

```
let y = 2*x;;  
# val y : int = 10
```

```
let x = 42;;  
# val x : int = 42
```

```
y;;  
# - : int = 10
```

```
x;;  
# - : int = 42
```

Local Definitions

- ▶ Naming with a delimited scope
- ▶ Syntax: `let name = exp1 in exp2`
- ▶ Here, the *scope* of *name* is *exp2*
- ▶ A local definition may temporarily hide a more global one.

Local Definition Examples I

```
let x = 4+5 in 2*x;;
```

```
# - : int = 18
```

```
x;;
```

```
# Characters 0-1:
```

```
  x;;
```

```
  ^
```

```
Error: Unbound value x
```

```
let x = 17;;
```

```
# val x : int = 17
```

```
x;;
```

```
# - : int = 17
```

```
let y = x+1 in y/3;;
```

```
# - : int = 6
```

Local Definition Examples II

```
let x = 4 in  
let y = x+1 in  
let x = 2*y in x;;  
# - : int = 10
```

```
let x = 4 in  
(let x = 17 in x+1) + x;;  
# - : int = 22
```

Visibility of Definitions

```
let x = 1;;  
:  
}x = 1  
let x = 2 in  
:  
}x = 2  
  let x = 3 in  
  :  
  }x = 3  
  :  
  }x = 2  
  :  
  }x = 1
```

Local definitions hide more global definitions

Simultaneous Definitions

- ▶ `let x = e :`
e is evaluated w.r.t. the value bindings before the `let`
- ▶ `let x1 = e1 and x2 = e2 :`
both expressions are evaluated w.r.t. the value bindings before the `let`
- ▶ Same effect as `let x2 = e2 and x1 = e1`
- ▶ Works both with global and local definitions

Simultaneous Definitions Examples I

```
let x = 1;;  
# val x : int = 1
```

(* sequential definitions *)

```
let x = 2 in  
  let y = x + 1 in (* y = 2+1 *)  
    x*y;;          (* 2*3 *)  
# - : int = 6
```

(* simultaneous definition *)

```
let x = 2  
  and y = x+1 in (* y = 1+1 *)  
    x*y;;        (* 2*2 *)  
# - : int = 4
```