

Objective Caml version 3.06

```
# type counter = {get: unit -> int; inc: unit -> unit};;
type counter = { get : unit -> int; inc : unit -> unit; }

# let c =
  let x = ref 0
  in
    {get = (fun () -> !x);
     inc = (fun () -> x := !x + 1)};;
  val c : counter = {get = <fun>; inc = <fun>}

# c.get;;
- : unit -> int = <fun>

# c.get ();;
- : int = 0

# c.inc;;
- : unit -> unit = <fun>

# c.inc (); c.inc ();;
- : unit = ()

# c.get ();;
- : int = 2

# let new_counter () =
  let x = ref 0
  in
    {get = (fun () -> !x);
     inc = (fun () -> x := !x + 1)};;
  val new_counter : unit -> counter = <fun>

# let c1 = new_counter ();;
val c1 : counter = {get = <fun>; inc = <fun>}

# let c2 = new_counter ();;
val c2 : counter = {get = <fun>; inc = <fun>}

# c1.inc (); c1.inc ();;
- : unit = ()

# c1.get ();;
- : int = 2

# c2.get ();;
- : int = 0

# type reset_counter =
{get: unit -> int; inc: unit -> unit; reset: unit -> unit};;
type reset_counter = {
  get : unit -> int;
  inc : unit -> unit;
  reset : unit -> unit;
}

# c1 = c2;;
Exception: Invalid_argument "equal: functional value".

#
```