

Objective Caml version 3.08.3

```

(* ----- *)
(* Blockschachtelung: Wird ein Name mehrmals deklariert, so ist die äußere De- *)
(* klaration durch die innere "verdeckt", aber sobald der Gültigkeitsbereich *)
(* der inneren Deklaration beendet ist, wird die äußere wieder sichtbar. Das *)
(* gilt natürlich auch für lambda-Abstraktionen anstelle von Deklarationen. *)
(* ----- *)

# let x = 1
in (let x = 2 in x) + x;;

- : int = 3

(* ----- *)
(* Statische Bindung: Der Name x, der in der Deklaration von add_x frei vor- *)
(* kommt, wird "zum Deklarationszeitpunkt gebunden", d.h. die Funktion add_x *)
(* addiert stets 1 zu ihrem Argument, auch wenn x später neu deklariert wird. *)
(* ----- *)

# let x = 1;;

val x : int = 1

# let add_x y = y + x;;

val add_x : int -> int = <fun>

# add_x 3;;

- : int = 4

# let x = 2;;

val x : int = 2

# add_x 3;;

- : int = 4

# add_x x;;

- : int = 3

(* ----- *)
(* Das gleiche Prinzip gilt natürlich auch für Funktionsnamen. Deshalb liefert *)
(* succ_f x stets das Resultat x^2 + 1, auch wenn f neu deklariert wird. Man *)
(* beachte den Unterschied zum "late binding" von Methoden bei der Vererbung. *)
(* ----- *)

# let f x = x * x;;

val f : int -> int = <fun>

# let succ_f x = f x + 1;;

val succ_f : int -> int = <fun>

# succ_f 5;;

- : int = 26

# let f x = x + 2;;

val f : int -> int = <fun>

# succ_f 5;;

- : int = 26

# succ_f (f 5);;

- : int = 50

#

```