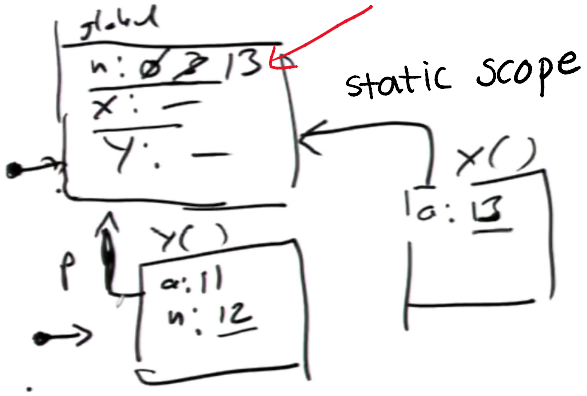
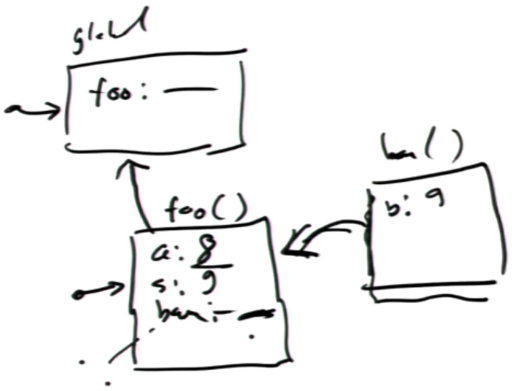
What about parameters?

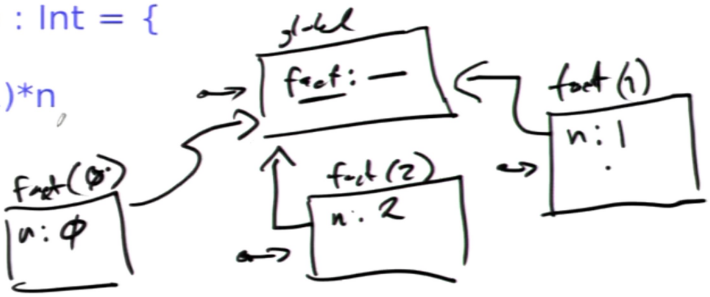
```
var n = 0
def X(a:Int) = {
  n = a
}
def Y(a:Int) = {
  var n = a + 1
  X(n+1)
}
n = 3
Y(n+8)
```




```
def foo(a:Int) : Int = {
  val s = a + 1
  def bar(b:Int) : Int = {
    a+b
  }
  bar(s)
}
foo(8)
```




```
def fact(n:Int) : Int = {
  if (n==0) 1
  else fact(n-1)*n
}
fact(2)
```



execute code
→ code itself, env → p → gp → ggp →
→ pass this around

To execute a function:
<Parameters: — env it was >

To execute a function:

< parameters: —— , env it was
body: —— defined >

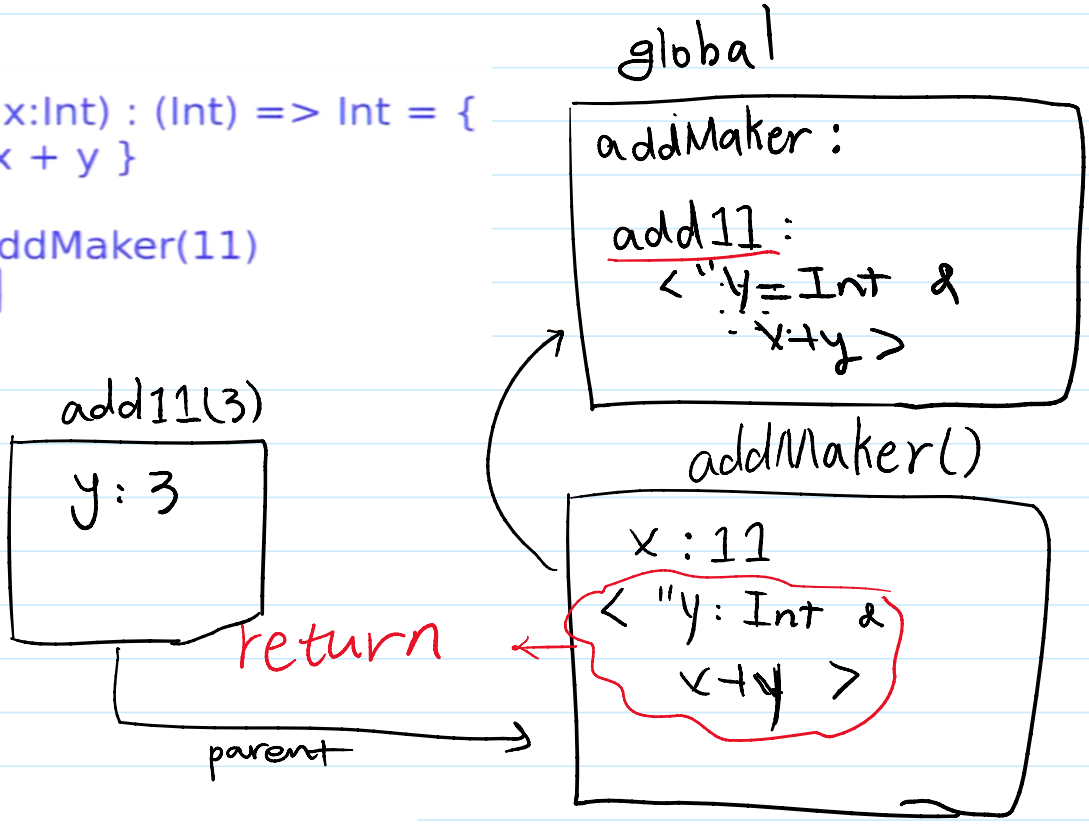
closure → name + env.
(para, body)

→ Continuation after the func.

```
foo("hello")
val y = 2
print(y+x)
```
+ env.

Why?

```
def addMaker(x:Int) : (Int) => Int = {
  (y:Int) => { x + y }
}
val add11 = addMaker(11)
add11(3) // 14
```



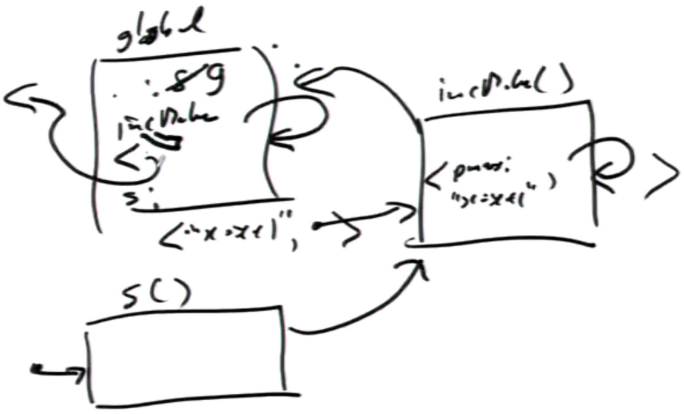

```
scala> var x = 8
x: Int = 8

scala> def incMaker() = {
     |   () => { x = x + 1 }
     | }
incMaker: ()() => Unit

scala> val s = incMaker()
s: () => Unit = $$Lambda$1057/572370257@2cec704c

scala> x
res0: Int = 8

scala> s()
```

```
scala> x
res2: Int = 9

scala> s()

scala> x
res4: Int = 10
```

```
scala> var x = 8
x: Int = 8

scala> def incMaker() = {
     |   () => { x = x + 1 }
     | }
incMaker: ()() => Unit

scala> val s = incMaker()
s: () => Unit = $$Lambda$1057/572370257@2cec704c

scala> x
res0: Int = 8

scala> s()

scala> x
res2: Int = 9
```

```
scala> x
res2: Int = 9

scala> s()

scala> x
res4: Int = 10
```

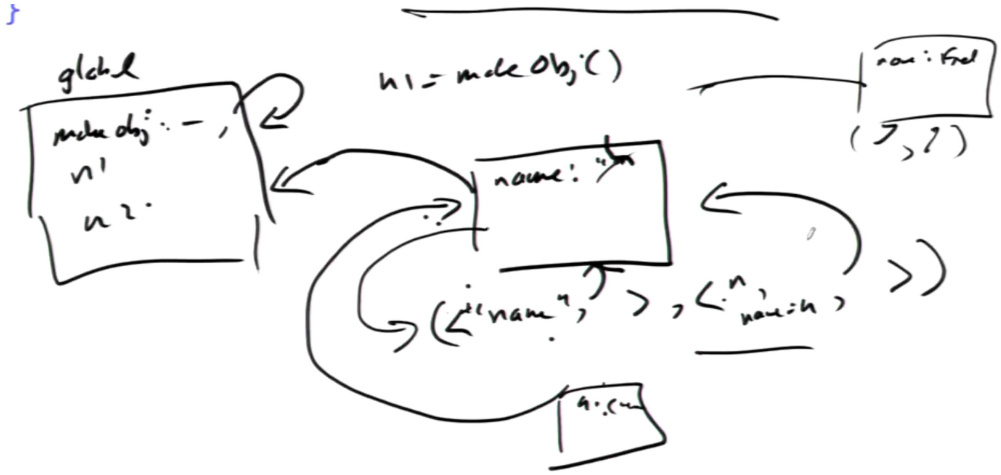
```
var x = 8
def incMaker() = {
  () => { x = x + 1 }
}
val s = incMaker()
x
```



→ Use the encapsulation of local data inherient to closure → to build objects

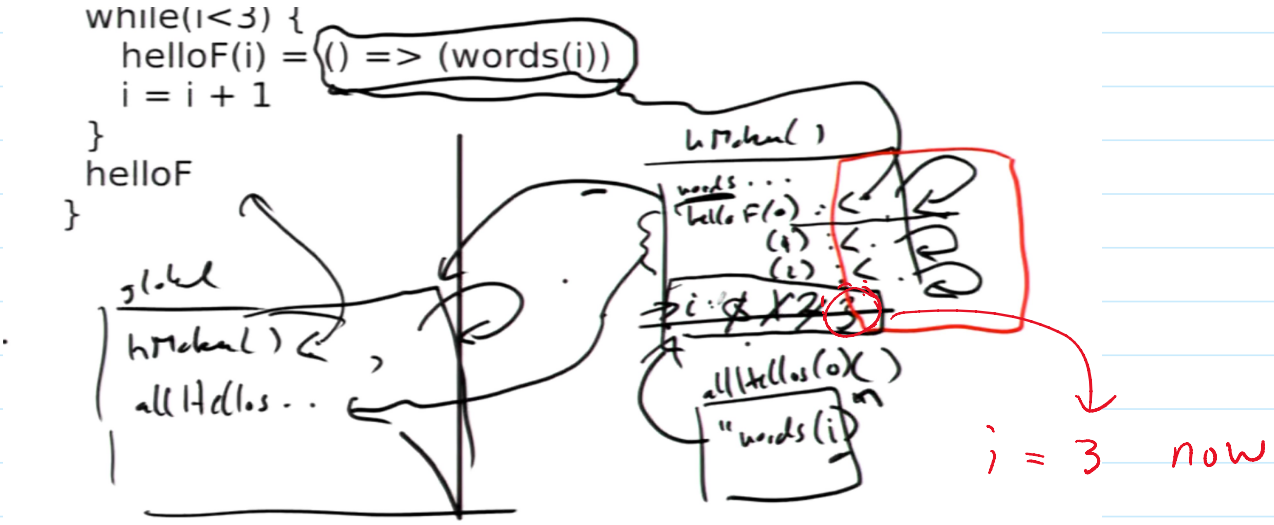
```
scala> def makeObj() = {
     |   var name = ""
     |   ( () => {name}, (n:String) => { name = n } )
     | }
makeObj: ()(() => String, String => Unit)

scala> val n1 = makeObj()
n1: (() => String, String => Unit) = ($$Lambda$1086/948315053@32118208,$$Lambda$
1087/153048669@414f87a9)

scala> val n2 = makeObj()
n2: (() => String, String => Unit) = ($$Lambda$1086/948315053@79b18230,$$Lambda$
1087/153048669@1d4f5506)
```

```
scala> n1._2("Clark")

scala> n1._1()
res6: String = Clark

scala> n2._1()
res7: String = ""

scala> n2._2("Fred")

scala> n2._1()
res9: String = Fred
```



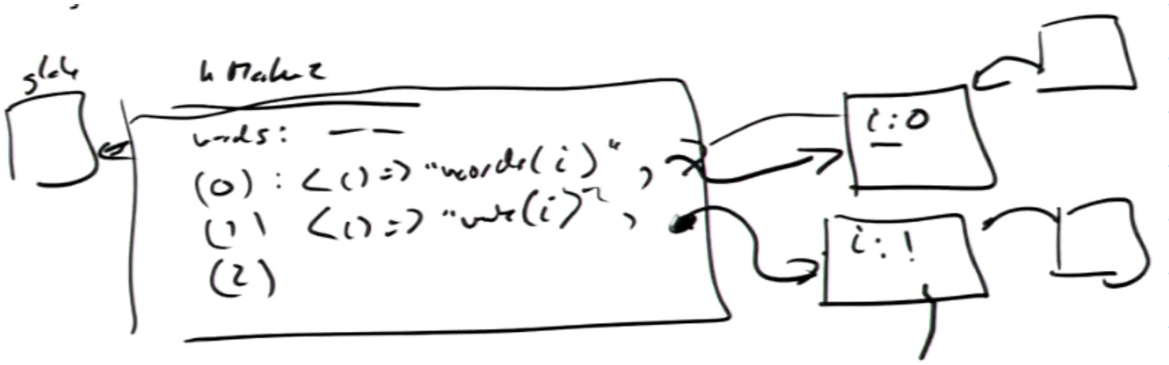

```
scala> def hMaker() = {
     |   var words = Array ("hello", "hi", "yo")
     |   var helloF = new Array[()=>String](3)
     |   var i = 0
     |   while(i<3) {
     |     helloF(i) = () => (words(i))
     |     i = i + 1
     |   }
     |   helloF
     | }
hMaker: ()Array[() => String]

scala> val allHellos = hMaker()
allHellos: Array[() => String] = Array($$Lambda$1163/1155566202@17ae13d5, $$Lamb
da$1163/1155566202@69f3e556, $$Lambda$1163/1155566202@1eb9d69a)
```

```
scala> allHellos(0)()
java.lang.ArrayIndexOutOfBoundsException: 3
  at .$anonfun$hMaker$1(<console>:16)
  ... 29 elided
```




```
def hMaker2() = {
  val words = Array("hello","hi","yo")
  (for (i <- 0 until 3) yield { () => (words(i)) }).toArray
}
```

```
scala> val allHellos2 = hMaker2()
allHellos2: Array[() => String] = Array($$Lambda$1211/2143645025@1382a7d8, $$Lam
bda$1211/2143645025@6074d638, $$Lambda$1211/2143645025@402feb85)

scala> allHellos2(0)()
res12: String = hello

scala> allHellos2(0)()
res13: String = hello

scala> allHellos2(0)()
res14: String = hello

scala> allHellos2(1)()
res15: String = hi
```



Language Construction

→ formal specification
→ leads to implementations

* → Syntax → its representation

(later) → Semantics → what it means to execute it.
→ type systems.

→ programs → mass of symbols
→ testool → stream of chars

executable execution → compiler execute