(Implementing) Recursion

```
fun inf(x): inf(x) end
```

```
fd("x", e-app("inf", e-id("x")))
```

```
inf(0)
↳ inf(0)
```

```
e-app("inf", e-num(0))
e-app("inf", e-num(0))
```

```
→ (let (inf-helper (lam (f) (f f)))
    (let (inf (lam () (inf-helper inf-helper)))
      (inf)))
```

inf-helper unbound when body was (f inf-helper)

```
(let (inf (lam () ((lam (f) (f f)) (lam (f) (f f))))) ←
  (inf))
```

```
(let (x (f x 1))
  x)
```

```
↳ ((lam (f) (f f)) (lam (f) (f f)))
```

Recursion HoF

```
(let (inf (lam (x) (inf x)))
     (inf 0))
```

```
inf = lam (x): inf(x) end
fun inf(x): inf(x) end
```

To be continued...

State, Variables, Mutation

"pass by value" "pass by reference"

FORGET THEM.

```
x = 4

this.x = 5

x = 6
```

Desugaring →

```
class D {
  int x;
  void setX1(){
    this.x = 4;
  }
  void setX2(){
    this.x = 5;
  }
  void setX3(int x) {
    x = 6;
  }
}
```

Field Mutation

Shadowing

Variable Mutation

```
D d = new D();        (d.x = 0)
int x = 10;
d.setX1();            d.x = 4
d.setX2();            d.x = 5
d.setX3(x);           d.x = 5

x = 10
```

<exp> = <exp>

Field Mutation

Structure
Value

```
<exp> := ...
       | (box <exp>)
       | (set-box <exp> <exp>)
       | (get-box <exp>)
       | (block <exp> <exp> ...)
```

```
(block (+ 12) 3 (- 2 1))
 (block 3 3 (- 2 1))
  (block 3 3 1)
   1
```

```
(let (b (box 5))
  (block
    (set-box b 6)
    (get-box b)
    (let (b2 b)
      (set-box b2 4)
      (get-box b2)
      (get-box b)))
```