How we evaluate Scheme expressions
[we'll get to quote in particular]
Lambda/closures, maybe

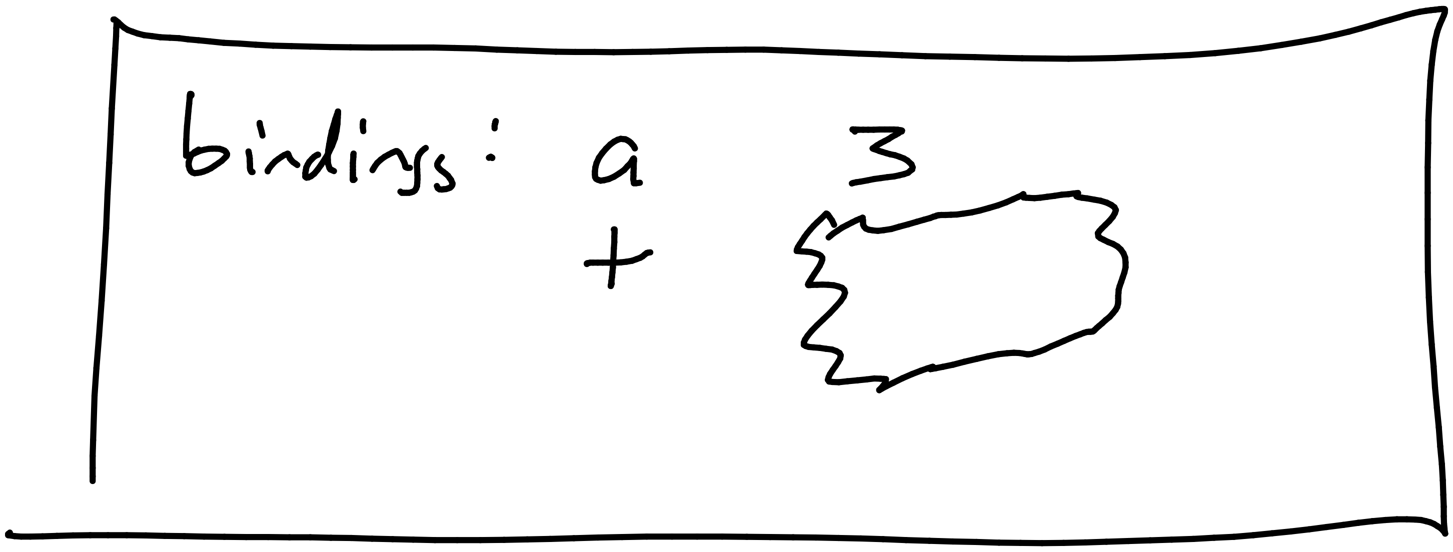
When we talk about <u>evaluating</u> a Scheme expression we mean executing it, or interpreting it, but effectively "doing what Scheme does with it."

evaluate 3 → 3

evaluating a <u>constant</u> just returns itself, just one call to eval

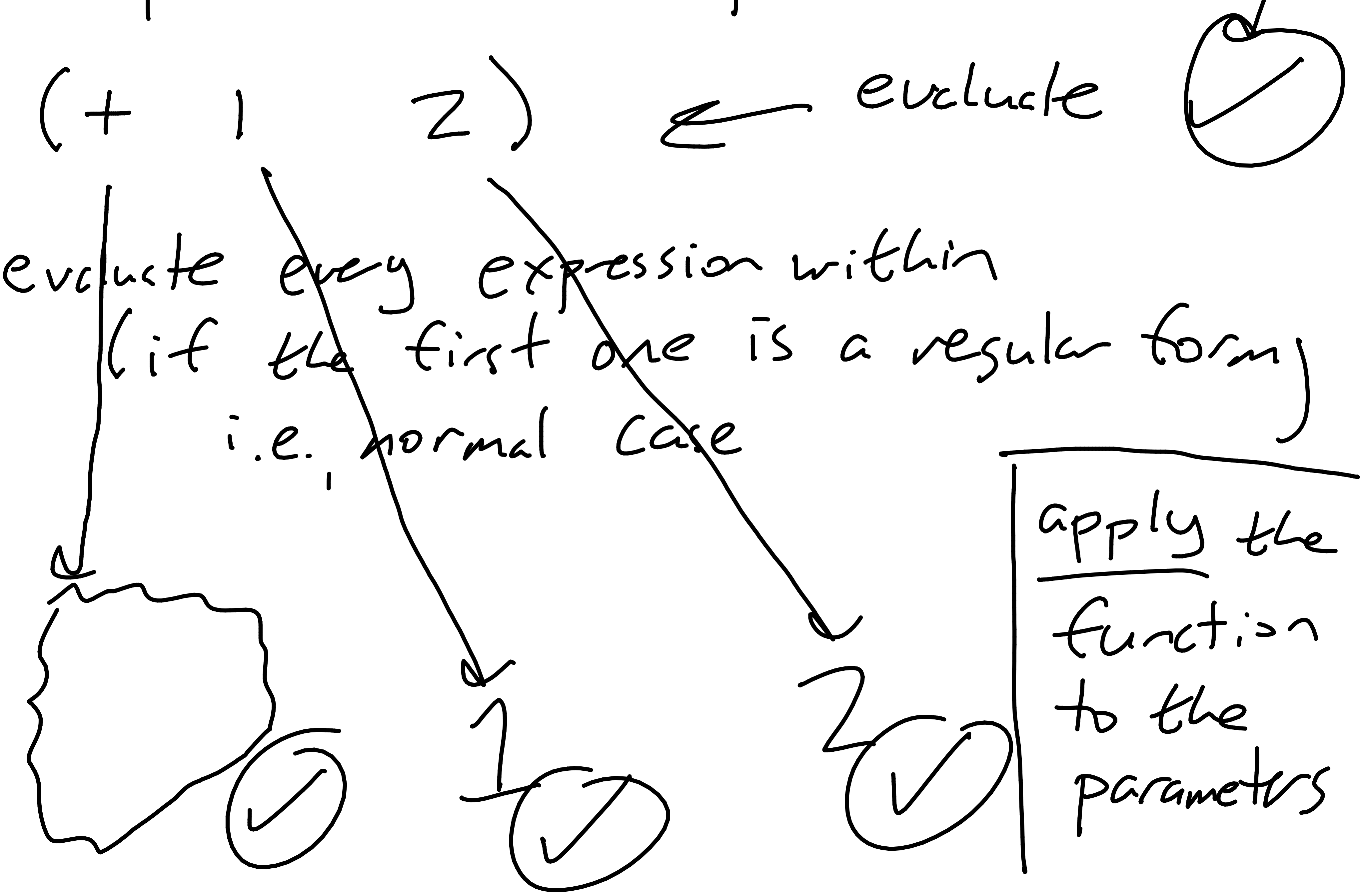
evaluating a symbol looks up the value and returns it - one call to eval

In the case of + it's still a symbol look up



What happens when you evaluate a parenthesized expression?

counting



evaluate every expression within (if the first one is a regular form)

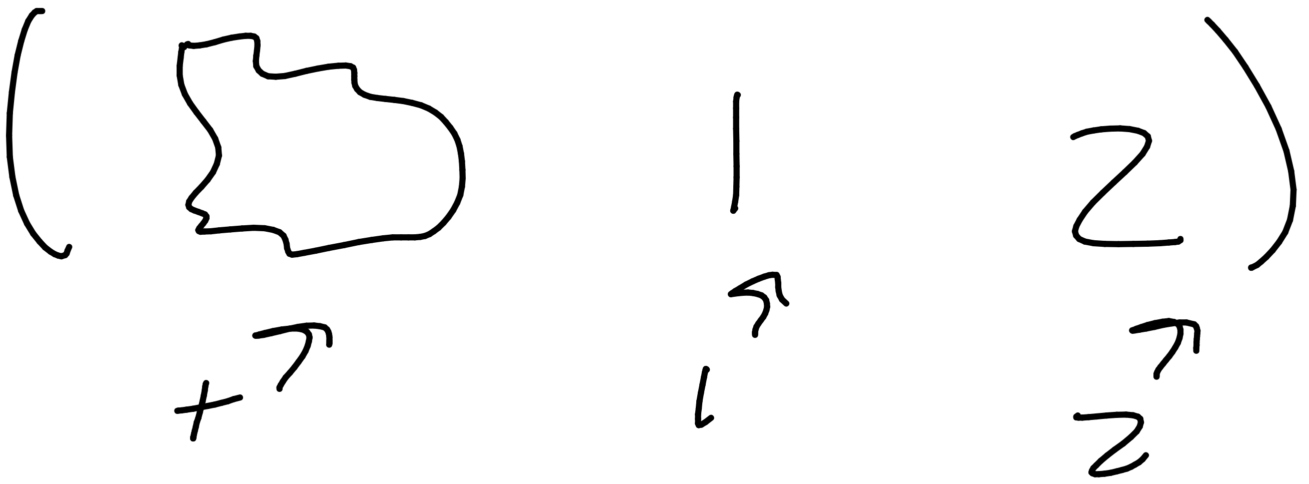
i.e., normal case

apply the function to the parameters

evaluate 3. How many?

3

evaluate (+ 1 2)



(if #t 3 4)

if is not a normal Scheme function. if is a special form, and it breaks how Scheme normally works.

Why did the Scheme designers make *if* weird?

if does "short circuit evaluation)

```
(if #t (+ 3 5) (+ 9 7))
```

when the condition is true, skip evaluating the alternative

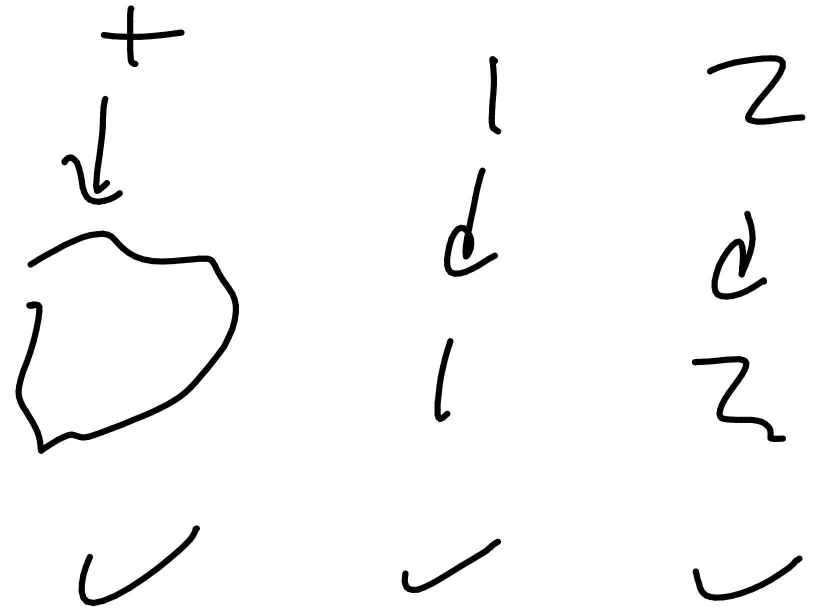
when the condition is false, skip evaluating the consequent

That means that *if* breaks the whole normal Scheme evaluation process.

(Repeat) How does Scheme evaluate a parenthesized expression?

(+ 1 2) evaluate ✓

if the first symbol is <u>not</u> <u>a</u> <u>special</u> <u>form</u>, proceed as usual



4 evaluations

if the first symbol <u>is</u> a special form, it is entirely in control of what evals happen.

(in our code, we never eval the special form itself.)

(if #t 3 5) ← eval

✓

↑

AACK! special form.
never gets evaluated, and instead,
we pass the remaining
parameters unevaluated to
a helper function that does
the special form

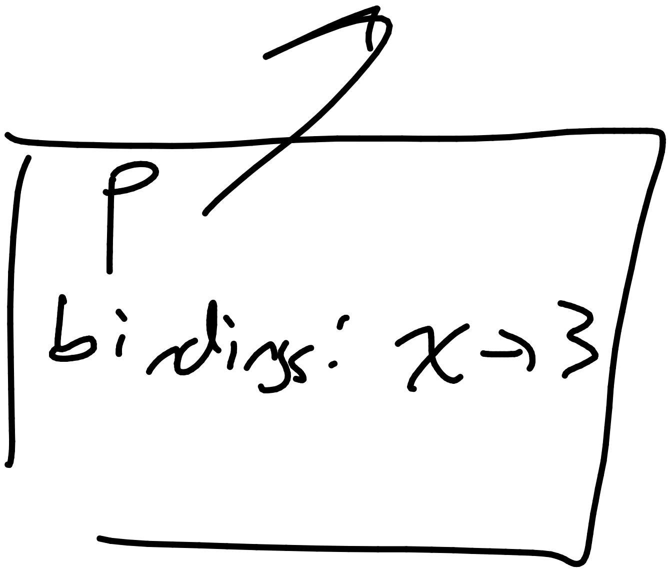
Call ↙ note! not eval unevaled

evalIf(- - - - - . #t 3 5)

– eval condition #t ⇒ #t ✓

Since true, eval 3 ⇒ 3 ✓

```
(let ((x 3))          ← eval ✓
  (+ x x))
```



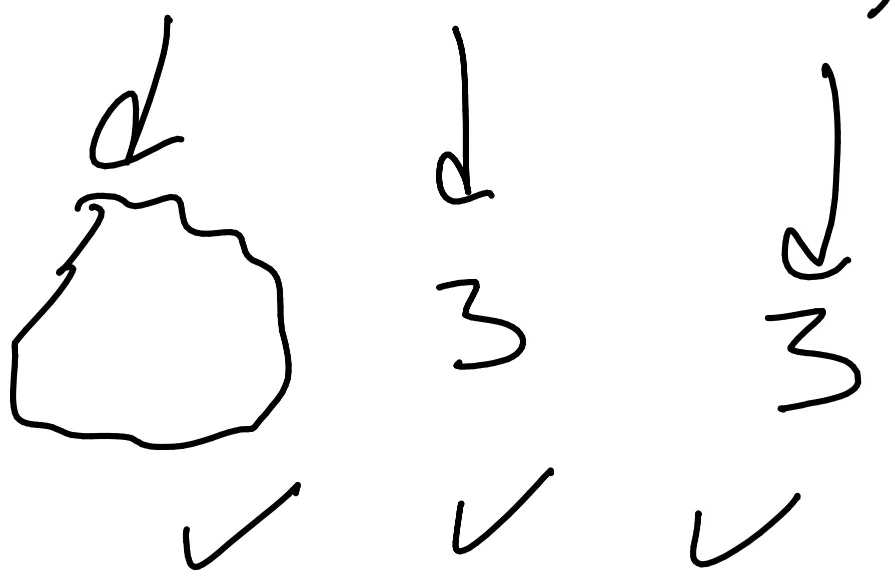
ARG~~H~~H special form

does eval 3 when binding x

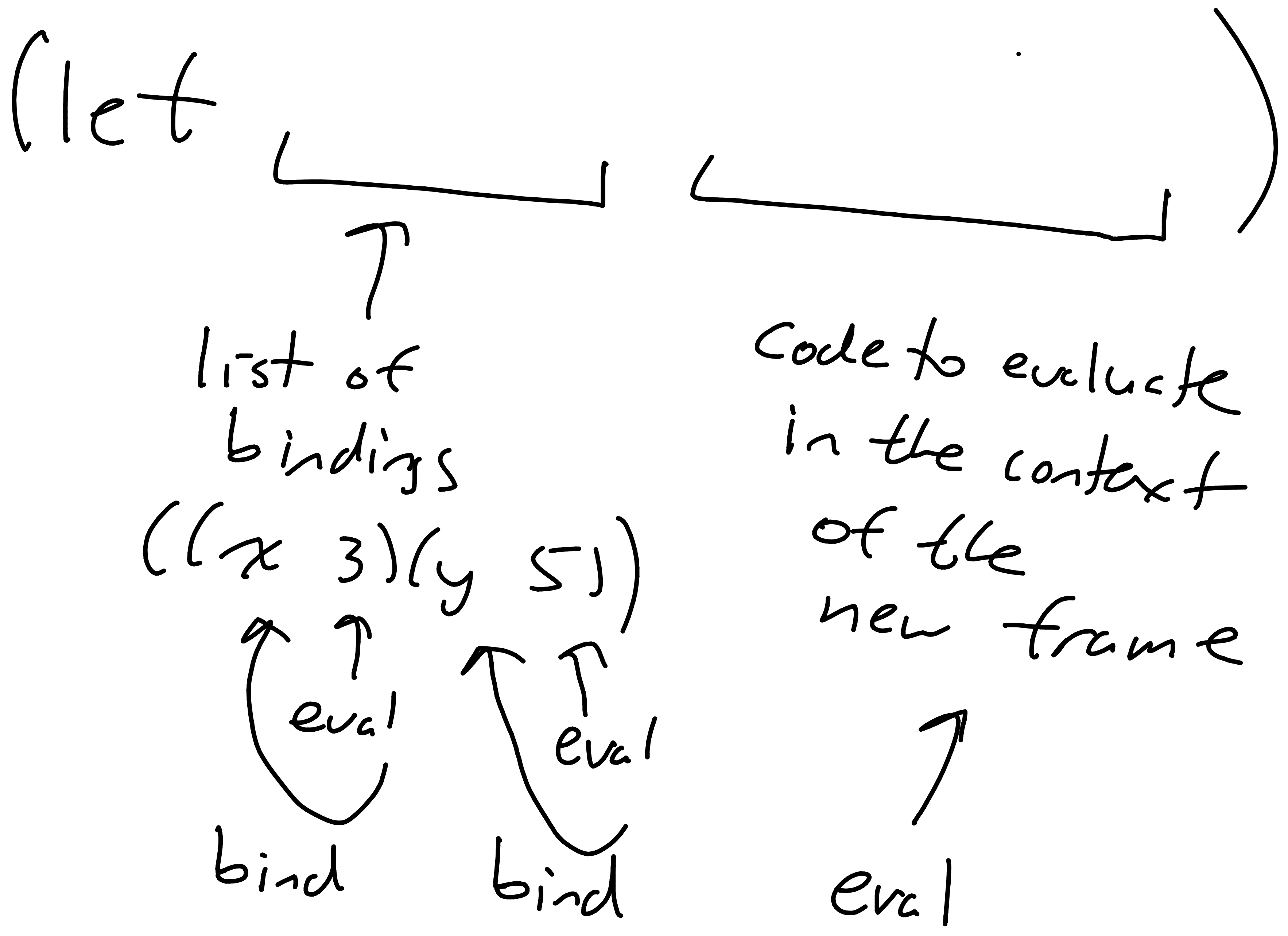
Afte frame is set up

let says

"eval (+ x x)" ✓



6 evals



(let ______ ______)

list of bindings

((x 3)(y 5))

eval

bind

eval

bind

Code to evaluate in the context of the new frame

eval

Quote is also a special form

(quote ________) ← eval this ✓

returns ________

<u>evals</u> <u>nothing</u>

evaluate (quote 3) ✓

special form noo~~ooooo~~o!

yechhhhh!

do no evals just return the parameter, uneval'd