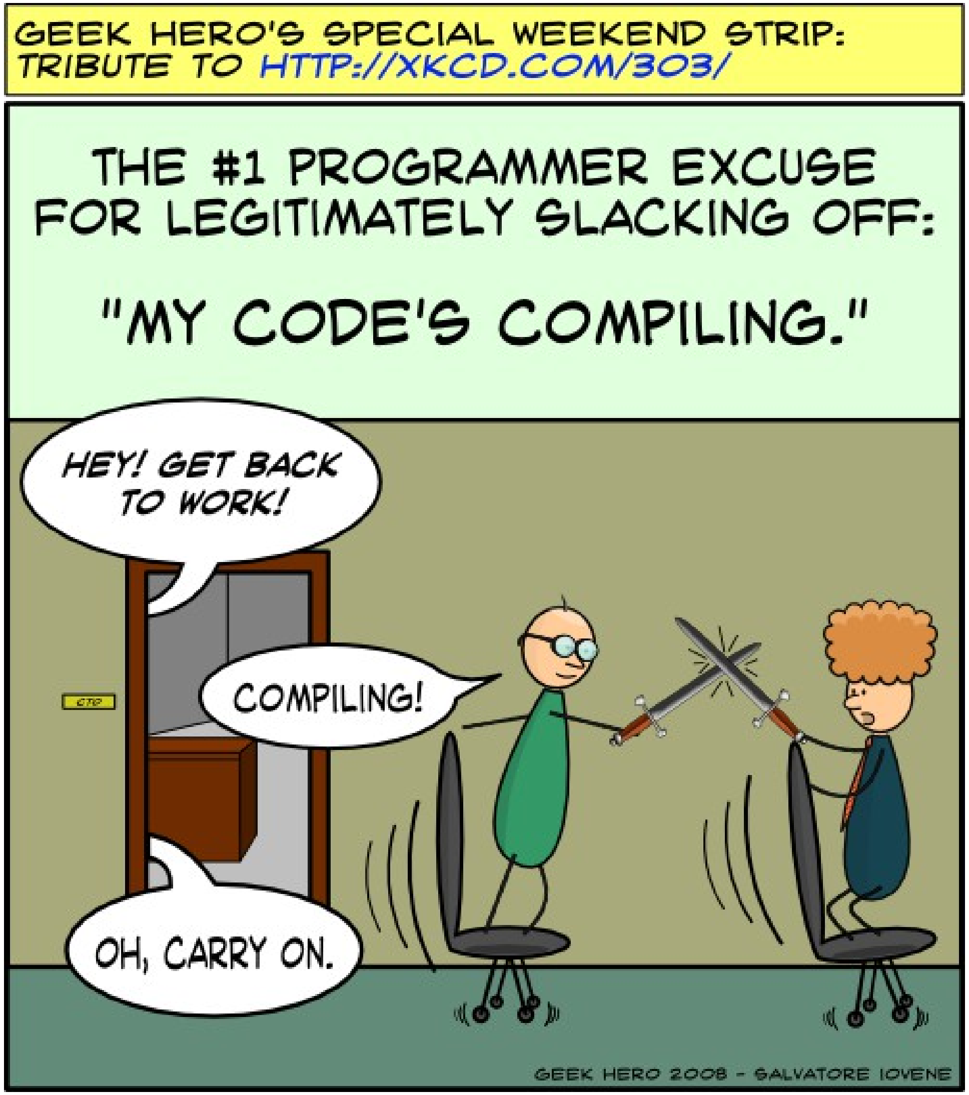
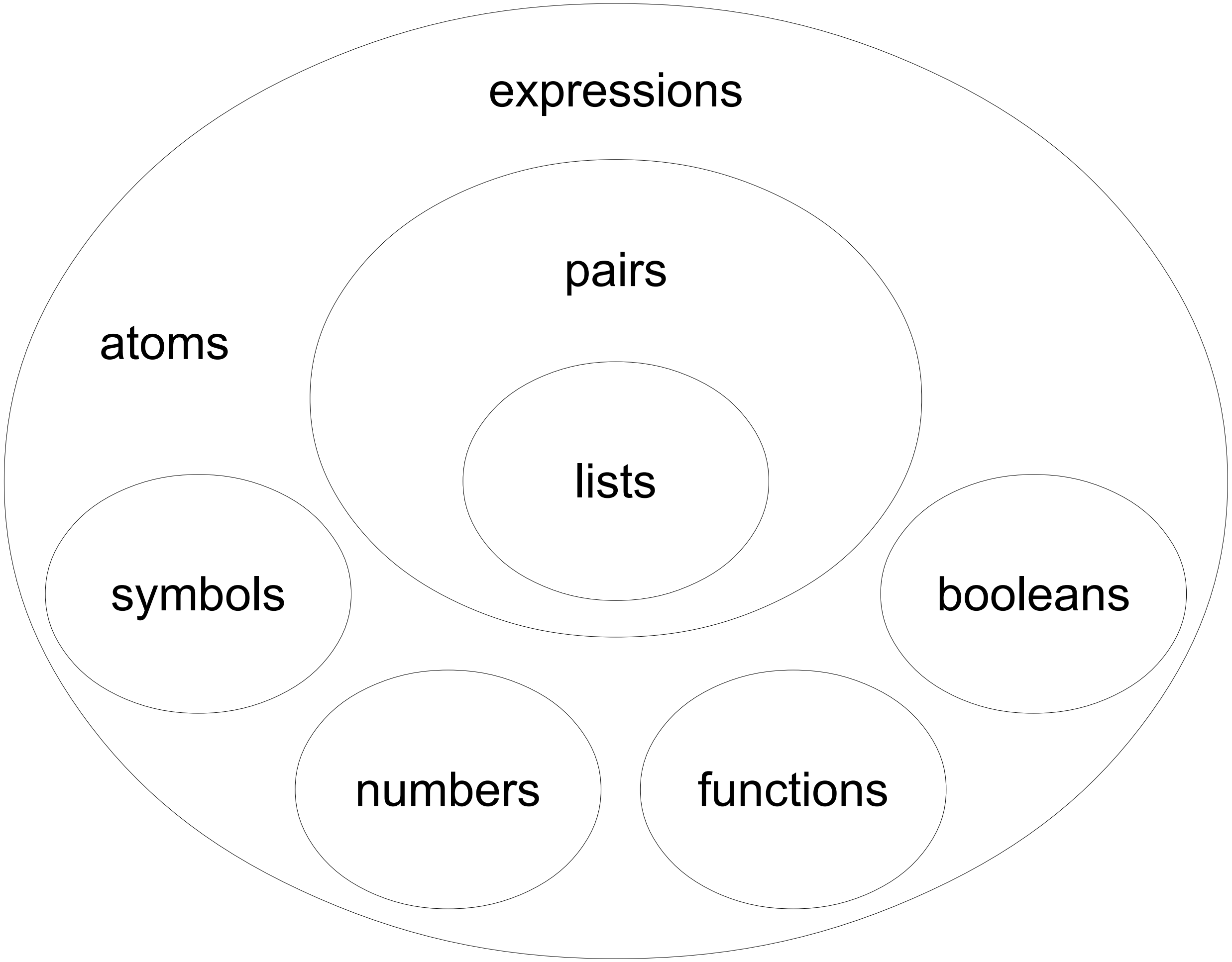

GEEK HERO'S SPECIAL WEEKEND STRIP:
TRIBUTE TO HTTP://XKCD.COM/303/
THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
CTO
COMPILING!
OH, CARRY ON.
GEEK HERO 2008 - SALVATORE IOVENE

Homework 1

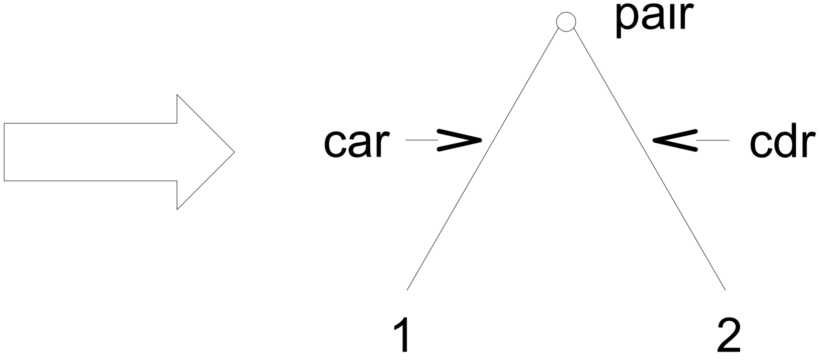
- Springer and Friedman
  - 1.2, 1.3, 1.4, 1.5, 1.6
  - 1.10, 1.14
  - 2.1, 2.3, 2.4, 2.6, 2.7, 2.10
  - 2.12, 2.13, 2.14, 2.15, 2.16
  - 2.18
- Any answers which are not Scheme definitions should be commented out using ;;

Scheme Datatypes



cons Makes Pairs

(cons 1 2)



pair

car

cdr

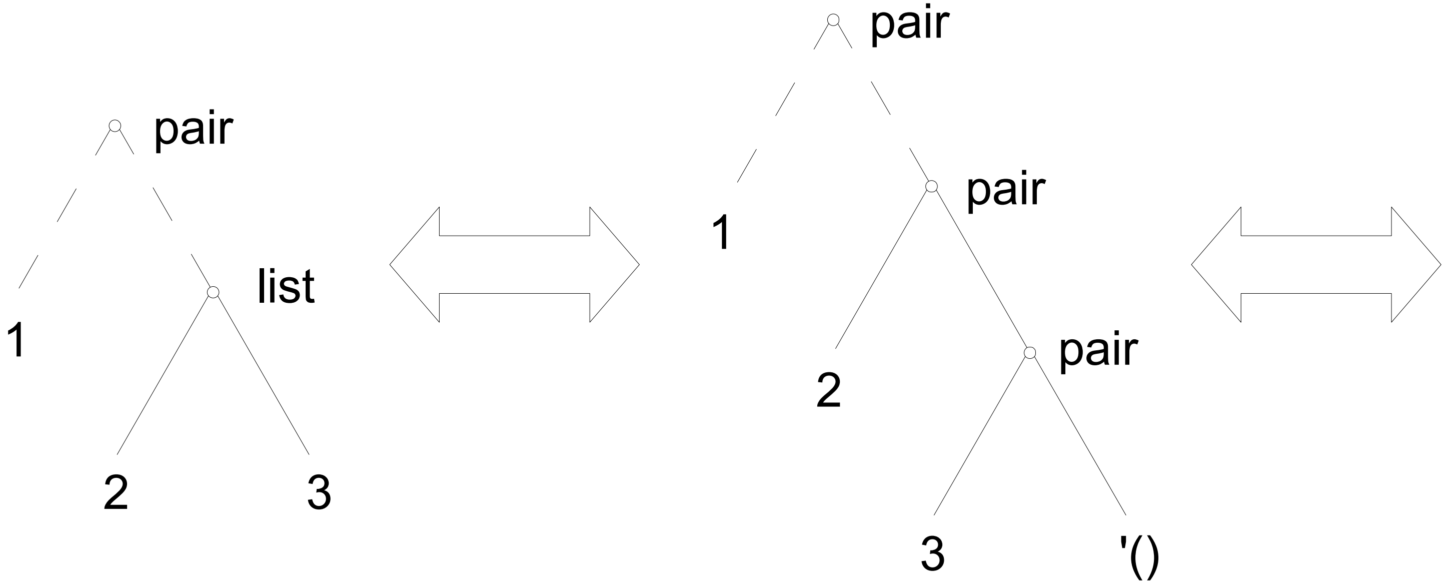
1

2

Lists

- A list is either
  - an empty list '()
  - a pair whose *cdr* is a list.

Adding Something to the Front of a List



(cons 1 (list 2 3))

(cons 1 (cons 2 (cons 3 '())))

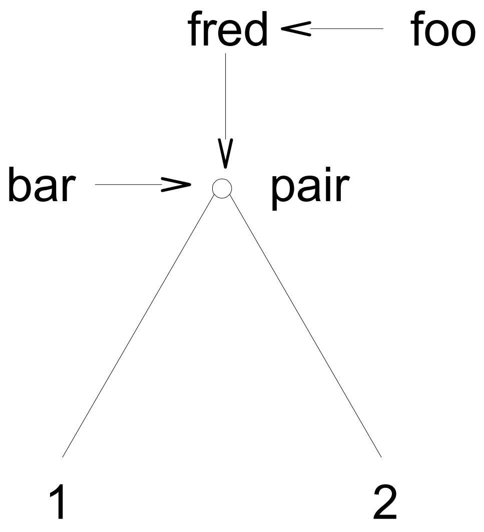
(list 1 2 3)

Symbols and Quotation

```
(define fred (cons 1 2))

(define foo 'fred)

(define bar fred)
```



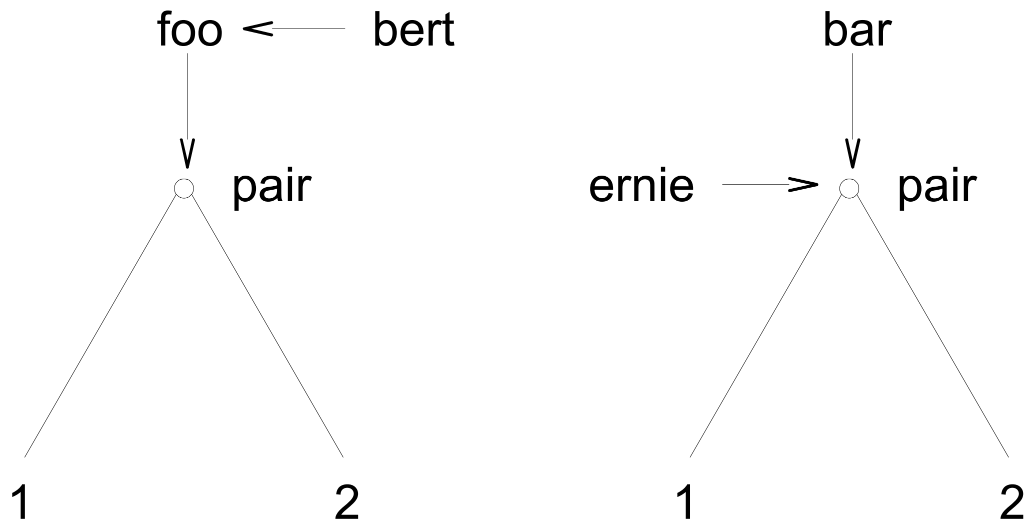
Symbols and Quotation (contd.)

```
(define foo (cons 1 2))

(define bar (cons 1 2))

(define bert 'foo)

(define ernie bar)
```



eq? versus *equal?*

(eq? foo bar) → #f

(equal? foo bar) → #t

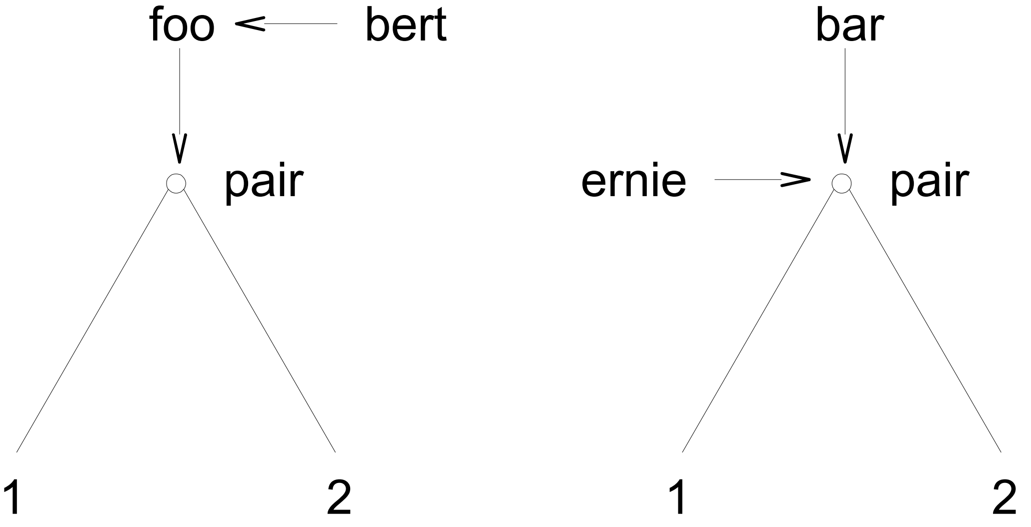
(eq? bert 'foo) → #t

(eq? bert foo) → #f

(equal? bert 'foo) → #t

(equal? bert ernie) → #f

(eq? bar ernie) → #t



(eq? x y) → (equal? x y)

cond special-form

(**cond** (*pred*$_1$ *val*$_1$) … (*pred*$_{N-1}$ *val*$_{N-1}$) (else *val*$_N$))

- The **cond** special-form evaluates *pred*$_1$.
- If *pred*$_1$ is not #f it evaluates and returns *val*$_1$.
- Otherwise **cond** evaluates *pred*$_2$.
- If *pred*$_2$ is not #f it evaluates and returns *val*$_2$.
- If none of *pred*$_1$… *pred*$_{N-1}$ evaluates to not #f **cond** evaluates and returns *val*$_N$.

or special-form

(**or** pred_1 pred_2 … pred_{N-1} pred_N)

- The **or** special-form evaluates pred_1.
- If pred_1 is not #f **or** returns it.
- Otherwise **or** evaluates pred_2.
- If pred_2 is not #f **or** returns it.
- If none of pred_1… pred_{N-1} evaluates to not #f **or** returns pred_N.

and special-form

(**and** *pred*$_1$ *pred*$_2$ … *pred*$_{N-1}$ *pred*$_N$)

- The **and** special-form evaluates *pred*$_1$.
- If *pred*$_1$ is #f **and** returns #f.
- Otherwise **and** evaluates *pred*$_2$.
- If *pred*$_2$ is #f **and** returns #f.
- If none of *pred*$_1$… *pred*$_{N-1}$ evaluates to #f **and** returns *pred*$_N$.

Imperative Programs

- A program in an imperative language is a *sequence of statements.*
- Each statement transforms the state of the machine, *i.e.*, the contents of registers and memory.
- The goal is to find a sequence of statements that will transform the input state into the desired output state.
- The sequence of statements is a *description of a process*.

Functional Programs

- A program in a functional language is an *expression.*
- Expressions are evaluated by recursively evaluating subexpressions.
- The expression is the *definition of the answer to a problem*.

A Program that Recognizes Lists

- Recall that a list is either
  - an empty list '()
  - a pair whose *cdr* is a list.
- In Scheme, the program that recognizes lists is literally the definition of a list

```
(define list?
  (lambda (sexpr)
    (or (null? sexpr)
          (and (pair? sexpr)
                 (list? (cdr sexpr))))))
```