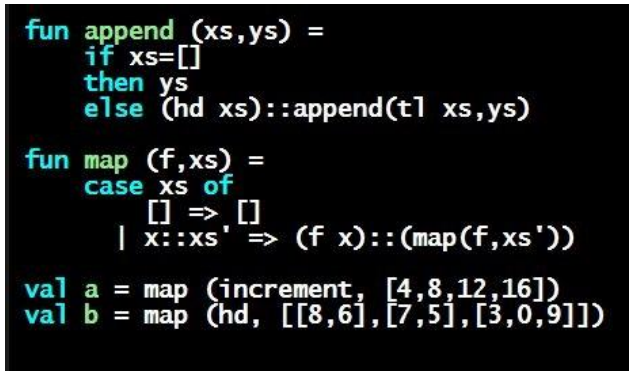

Programming Languages

Dan Grossman
2013

Cond

Better style

Avoid nested if-expressions when you can use cond-expressions instead

- Can think of one as sugar for the other

General syntax:
```
(cond [e1a e1b]
      [e2a e2b]
       ...
      [eNa eNb])
```

- Good style: **`eNa`** should be **`#t`**

Example

```
(define (sum xs)
  (cond [(null? xs) 0]
        [(number? (car xs))
         (+ (car xs) (sum (cdr xs)))]
        [#t (+ (sum (car xs)) (sum (cdr xs)))]))
```

A variation

As before, we could change our spec to say instead of errors on non-numbers, we should just ignore them

So this version can work for any list (or just a number)

- Compare carefully, we did *not* just add a branch

```
(define (sum xs)
  (cond [(null? xs) 0]
        [(number? xs) xs]
        [(list? (car xs))
         (+ (sum (car xs)) (sum (cdr xs)))]
        [#t (sum (cdr xs))]))
```

What is true?

For both **`if`** and **`cond`**, test expression can evaluate to anything

- It is not an error if the result is not **`#t`** or **`#f`**
- (Apologies for the double-negative ☺)

Semantics of **`if`** and **`cond`**:

- "Treat anything other than **`#f`** as true"
- (In some languages, other things are false, not in Racket)

This feature makes no sense in a statically typed language

Some consider using this feature poor style, but it can be convenient