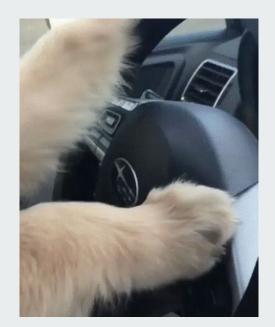
CSE 341 AA: Section 7

Porter Jones pbjones@cs.washington.edu
Office Hours: Thursdays 5:30 - 7:30pm



Implementing a language in Racket

- Hardest distinction is to know what is handled by Racket and what needs to be handled by your language
- You get to choose (sort of) what expressions your language supports and the semantics for evaluating them
 - We get to decide and implement all of those semantics rules we have been learning all quarter!!!

Implementing a language in Racket

Racket prevents "wrong" expressions like the one below from being created

```
(const #t)
```

 We have to define an interpreter that prevents "wrong" expressions like this one:

```
(add (bool #t) (const 3))
```

Implementing a language in Racket

```
; prevents first example from previous slide
(struct const (int) #:transparent)
; inside interpreter: prevents second example
[(add? e)
(let ([v1 (eval-exp (add-e1 e))]
      [v2 (eval-exp (add-e2 e))])
  (if (and (const? v1) (const? v2))
      (const (+ (const-int v1) (const-int v2)))
      (error "add applied to non-number")))]
```

quote

- quote is a Racket function for converting code to a list of tokens
- quote: parses its argument as data
 - o can also use 'for alternate syntax
 - (e) is equivalent to (quote e)
- eval: takes data and evaluates it
 - (eval (quote e)) = e

quote examples

```
(quote (+ 3 4 (+ 5 6)))

; produces the list '(+ 3 4 (+ 5 6))

(eval (quote (+ 3 4 (+ 5 6)))) ; produces 18
```

quasiquote

- Similar to quote, but with the option to unquote tokens inside of quasiquote!
 - o can also use `for alternate syntax
 - (e) is equivalent to (quasiquote e)
- unquote unquotes the next token inside a quasiquote
 - o can also use `for alternate syntax
 - , (e) is equivalent to (unquote e)

quasiquote examples

```
(quasiquote (+ 3 4 (unquote (+ 5 6))))
; produces the list '(+ 3 4 11)
(eval (quasiquote (+ 3 4 (unquote (+ 5 6)))))
; still produces 18 when called with eval
```