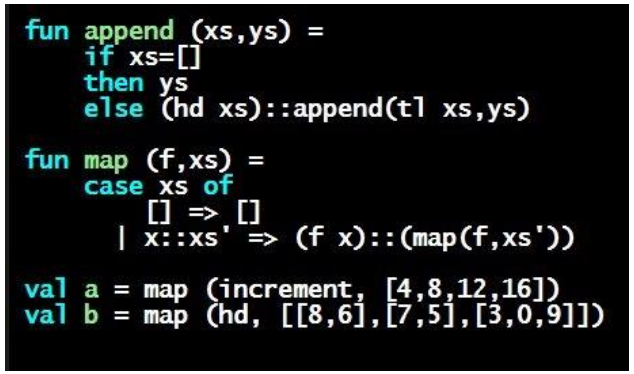

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

fun map (f,xs) =
    case xs of
        [] => []
      | x::xs' => (f x)::(map(f,xs'))

val a = map (increment, [4,8,12,16])
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Programming Languages

Dan Grossman
University of Washington

Part B Overview

Where we've been…

Part A:

1. Basics, functions, recursion, scope, variables, tuples, lists, …
2. Datatypes, pattern-matching, tail recursion
3. First-class functions, closures [and course motivation!]
4. Type inference, modules, equivalence

Overall: A *precisely specified* introduction to statically typed functional programming built up piece-by-piece

- Will highly leverage this foundation in Parts B & C

And now… Part B

"Section 5":

- Quick "re-do" in a dynamically typed language
  - No type system, inference, etc.
  - Different syntax (lots of parentheses)
  - Similar: lists, closures, functions, …
- Delaying evaluation
  - With zero-argument functions
  - Why: Delay/avoid computation, infinite streams, memoization
- Macros
  - Main issues
  - Optional: Some Racket specifics

And now… Part B

"Section 6":

- Datatype-Programming in Racket
- Implementing Programming Languages
  - Compiler vs. Interpreter
  - Abstract syntax trees
  - Implementing Environments and Closures
- Static vs. Dynamic Typing
  - What is static checking
  - Soundness and completeness
  - Advantages and disadvantages of static checking

We decided after recording this video to move Static vs. Dynamic Typing to its own "Section 7"