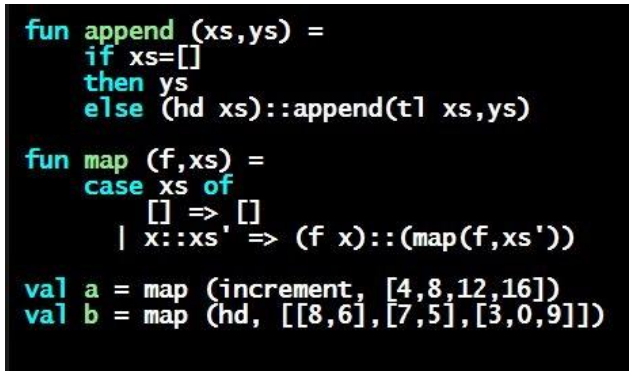

Programming Languages

Dan Grossman

Closures and Recomputation

When Things Evaluate

Things we know:

- A function body is not evaluated until the function is called
- A function body is evaluated every time the function is called
- A variable binding evaluates its expression when the binding is evaluated, not every time the variable is used

With closures, this means we can avoid repeating computations that do not depend on function arguments

- Not so worried about performance, but good example to emphasize the semantics of functions

Recomputation

These both work and rely on using variables in the environment

```
fun allShorterThan1 (xs,s) =
    filter(fn x => String.size x < String.size s,
           xs)

fun allShorterThan2 (xs,s) =
    let val i = String.size s
    in filter(fn x => String.size x < i, xs) end
```

The first one computes **`String.size`** once per element of **`xs`**

The second one computes **`String.size s`** once per list

- Nothing new here: let-bindings are evaluated when encountered and function bodies evaluated when *called*