

```
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

ML Expressions and Variable Bindings

Mindset

- “Let go” of all programming languages you already know
- For now, treat ML as a “totally new thing”
 - Time later to compare/contrast to what you know
 - For now, “oh that seems kind of like this thing in [Java]” will confuse you, slow you down, and you will learn less
- Start from a blank file...

A very simple ML program

[The same program we just wrote in Emacs; here for convenience if reviewing the slides]

```
(* My first ML program *)

val x = 34;

val y = 17;

val z = (x + y) + (y + 2);

val q = z + 1;

val abs_of_z = if z < 0 then 0 - z else z;

val abs_of_z_simpler = abs z
```

A variable binding

```
val z = (x + y) + (y + 2) ; (* comment *)
```

More generally:

```
val x = e;
```

- *Syntax:*
 - *Keyword* **val** and *punctuation* = and ;
 - *Variable* **x**
 - *Expression* **e**
 - Many forms of these, most containing *subexpressions*

The semantics

- **Syntax** is just how you write something
- **Semantics** is what that something means
 - **Type-checking** (before program runs)
 - **Evaluation** (as program runs)
- For variable bindings:
 - Type-check expression and extend **static environment**
 - Evaluate expression and extend **dynamic environment**

So what is the precise syntax, type-checking rules, and evaluation rules for various expressions? Good question!