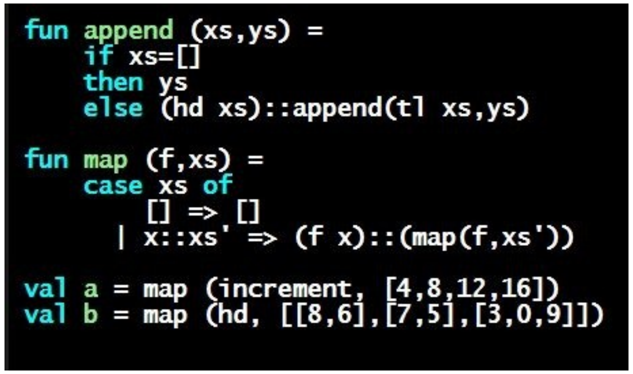

```
fun append (xs,ys) =
    if xs=[]
    then ys
    else (hd xs)::append(tl xs,ys)

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
2013

More Boolean and Comparison Expressions

Some More Expressions

Some “odds and ends” that haven’t come up much yet:

- Combining Boolean expressions (and, or, not)
- Comparison operations

Boolean operations

```
e1 andalso e2
```

- Type-checking: ***e1*** and ***e2*** must have type **`bool`**
- Evaluation: If result of ***e1*** is **`false`** then **`false`** else result of ***e2***

```
e1 orelse e2
```

```
not e1
```

- Syntax in many languages is **`e1 && e2`**, **`e1 || e2`**, **`!e`**
  - **`&&`** and **`||`** don't exist in ML and **`!`** means something different
- "Short-circuiting" evaluation means **`andalso`** and **`orelse`** are not functions, but **`not`** is just a pre-defined function

Style with Booleans

Language does not *need* **`andalso`** , **`orelse`** , **`not`**

```
(* e1 andalso e2 *)
if e1
then e2
else false
```

```
(* e1 orelse e2 *)
if e1
then true
else e2
```

```
(* not e1 *)
if e1
then false
else true
```

Using more concise forms generally much better style

And definitely plea

```
(* just say e (!!!) *)
if e
then true
else false
```

Comparisons

For comparing **`int`** values:

`=   <>   >   <   >=   <=`

You might see weird error messages because comparators can be used with some other types too:

- **`> < >= <=`** can be used with **`real`**, but not 1 **`int`** and 1 **`real`**
- **`=   <>`** can be used with any "equality type" but not with **`real`**
  - Let's not discuss equality types yet