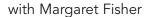
## **Scala Essential Training**





## Scala Reference Guide

Scala uses arrows in many of its expressions, including ->, <-, =>.

Examples: val m = Map(1->'a', 2->'b', 3->'c')

Type = SimpleType | FunctionType

FunctionType = SimpleType '=>' Type | '(' [Types]') '=>' Type

SimpleType = Identifier

Types = Type {','Type}

A type can be a numeric type, Boolean type, String type, or function type (ie: Int=>Int or (Int,Int)=>Int).

Expr = InfixExpr | FunctionExpr | if(Expr) Expr else Expr

InfixExpr = PrefixExpr | InfixExpr Operator InfixExpr

Operator = identifier

 $PrefixExpr = [+ | - | ! | \sim ] SimpleExpr$ 

SimpleExpr = identifier | literal | SimpleExpr.identifier | Block

Function Expr = Bindings => Expr

Bindings = identifier [':' SimpleType] | ([Binding {,Binding}])

Binding = identifier [:Type]

Block = {{Def;}Expr}

## An expression can be...

an identifier like x

a literal like 0, 1.9, "abc"

a function application like sqrt(x)

operator application like -x, y+x

selection like math.abs

conditional expression like if(x<0) -x else x

block like { val  $x = math.abs(y); x*2 }$ 

anonymous function like x=>x+1

## **Definitions**

```
Def = FunDef | ValDef
```

FunDef = def ident  $\{([Parameters])\}\$  A function definition: def square(x:Int)=x\*x

$$ValDef = val ident [:Type] = Expr$$
 A value definition:  $val y = square(2)$ 

Parameter = ident:[=>] Type Call by value, (x:Int)

Call by name parameter, (y:=>Double)

Parameters = Parameter { , Parameter}