PEP Scala (3)

Email:	christian.urban at kcl.ac.uk
Office:	S1.27 (1st floor Strand Building)
Slides & Code:	KEATS

The Joy of Immutability

- If you need to manipulate some data in a list say, then you make a new list with the updated values, rather than revise the original list. Easy!

```
val old_list = List(1, 2, 3, 5)
val new_list = 0 :: old_list
```

- You do not have to be defensive about who can access the data (concurrency, lazyness).

Email: Hate 'val'

Subject: **Hate 'val'** 01:00 AM

Hello Mr Urban,

I just wanted to ask, how are we suppose to work with the completely useless **`val`**, that can't be changed ever? Why is this rule active at all? I've spent 4 hours not thinking on the coursework, but how to bypass this annoying rule. What's the whole point of all these coursework, when we can't use everything Scala gives us?!?

Regards.
«deleted»

Subject: **Re: Hate '`val`'** 01:02 AM

«my usual rant about fp...
concurrency bla bla... better programs yada»

PS: What are you trying to do where you desperately want to use `var`?

Subject: **Re: Re: Hate 'val'** 01:04 AM

Right now my is_legal function works fine:

```
def is_legal(dim: Int, path: Path)(x: Pos): Boolean = {
  var boolReturn = false
  if(x._1 > dim || x._2 > dim || x._1 < 0 || x._2 < 0) {
  else { var breakLoop = false
         if(path == Nil) { boolReturn = true }
         else { for(i <- 0 until path.length) {
                    if(breakLoop == false) {
                      if(path(i) == x) {
                        boolReturn = true
                        breakLoop = true
                      }
                      else { boolReturn = false }
                    } else brea
            }
         }
         boolReturn
  }
```

...but I can't make it work with boolReturn being val. What approach would you recommend in this case, and is using var in this case justified?

Subject: **Re: Re: Hate 'val'** 01:04 AM

Right now my is_legal function works fine:

```
def is_legal(dim: Int, path: Path)(x: Pos): Boolean = {
  var boolReturn = false
  if(x._1 > dim || x._2 > dim || x._1 < 0 || x._2 < 0) {
  else { var breakLoop = false
         if(path == Nil) { boolReturn = true }
         else { for(i <- 0 until path.length) {
                    if(breakLoop == false) {
                      if(path(i) == x) {
                        boolReturn = true
                        breakLoop = true
                      }
                      else { boolReturn = false }
                    } else brea
```

```
                   turn
```

Me:

...but I can't make it work with boolReturn being val. What approach would you recommend in this case, and is using var in this case justified?

Subject: **Re: Re: Re: Hate 'val'** 01:06 AM

OK. So you want to make sure that the x-position is not outside the board....and furthermore you want to make sure that the x-position is not yet in the path list. How about something like

```
def is_legal(dim: Int, path: Path)(x: Pos): Boolean =
...<<some board conditions>>... && !path.contains(x)
```

Does not even contain a val.

(This is all on one line)

Subject: **Re: Re: Re: Re: Hate '`val`'** 11:02 AM

THANK YOU! You made me change my coding perspective. Because of you, I figured out the next one...

Types

- Base types

  `Int`, `Long`, `BigInt`, `Float`, `Double`
  `String`, `Char`
  `Boolean`

- Compound types

`List[Int]`	lists of Int's
`Set[Double]`	sets of Double's
`(Int, String)`	Int-String pair
`List[(BigInt, String)]`	lists of BigInt-String pairs
`List[List[Int]]`	list of lists of Int's

Where to go on from here?

- Martin Odersky (EPFL)...he is currently throwing out everything and starts again with the dotty compiler for Scala
- Elm (`http://elm-lang.org`)...web applications with style
- Haskell, Ocaml, Standard ML, Scheme

Questions?

Thanks: "*By the way - Scala is really getting quite fun when you start to get the hang of it...*"