PEP Scala (1)

Email:	christian.urban at kcl.ac.uk
Office:	N7.07 (North Wing, Bush House)
Slides & Code:	KEATS
Scala Office Hours:	Thursdays 11 – 13

Why Scala?


twitter
LinkedIn
theguardian
Morgan Stanley
CREDIT SUISSE

...


EDF
ENERGY
Novell
foursquare
HSBC

...

Why Scala?

- compiles to the JVM
  (also JavaScript, native X86 in the works)
- integrates seamlessly with Java
- combines **functional** and **object-oriented** programming
- it is a bit on the "mathematical" side
  (no pointers, no `null`)
- often one can write very concise and elegant code

alternatives:
Elm, Haskell, Ocaml, F♯, Erlang, ML, Lisp (Racket), ...

Java vs Scala

Java

```
public class Point {
  private final int x, y;

  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public int x() { return x; }

  public int y() { return y; }
}
```

Scala

```
class Point(val x: Int, val y: Int)
```

First Steps: Scala Tools

- there is a plugin for Eclipse (called Scala IDE)
- there is also a plugin for IntelliJ
- there is a worksheet mode in Eclipse and IntelliJ
- I use Sublime or venerable Emacs ;o)

Why Scala?

Scala, Elm, Haskell, Ocaml, F♯, Erlang, ML, Lisp (Racket), ...

Why Functional Programming?

Scala, Elm, Haskell, Ocaml, F♯, Erlang, ML, Lisp (Racket), ...

Why Functional Programming?

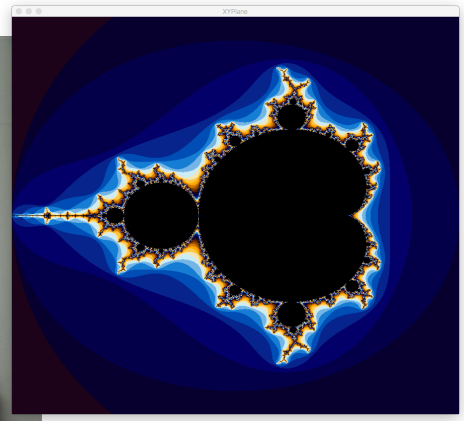
"If you want to see which features will be in mainstream programming languages tomorrow, then take a look at functional programming languages today."

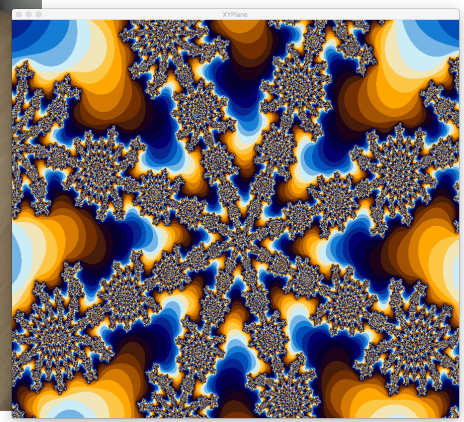
—Simon Peyton Jones (works at Microsoft)
main developer of the Glasgow Haskell Compiler

Scala, Elm, Haskell, Ocaml, F♯, Erlang, ML, Lisp (Racket), ...

1986

3 days





64K RAM, no HD, no monitor, lots of cables

1986

1988, C

1986

1988, C

1992, Linux

1986

1988, C

1992, Linux

1996

1986

1988, C

1992, Linux

1996

2000

1986

1988, C

1992, Linux

1996

2000

2012?

1986

1988, C

1992, Linux

1996

2000

2012?

2017

1986

1988, C

1992, Linux

1996

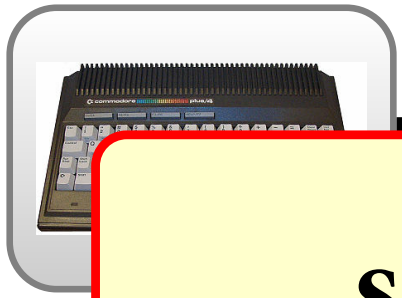
2000

2012?

2017

1986: no Internet
no Amazon
no FB, no mobiles,...

1986



1988, C

1992, Linux

2012?

2017

Speedup by Moore's Law

1986:	3 days	**1996:**	135 mins
1988:	1.5 days	**1998:**	67 mins
1990:	18 hs	**2000:**	33 mins
1992:	9 hs	**2002:**	16 mins
1994:	4.5 hs		???

Every two years, computers got twice as powerful.

no Amazon

no FB, no mobiles,...

1986

1988, C

1992, Linux

1996

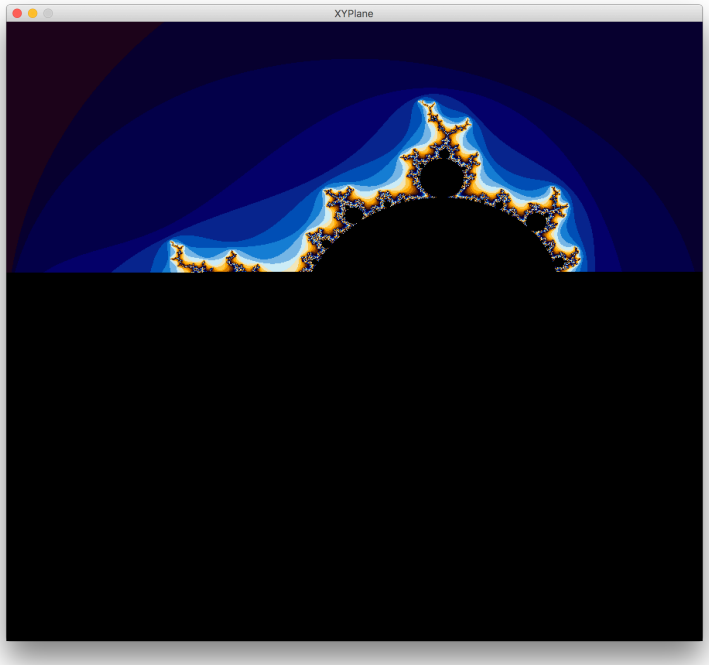
2000

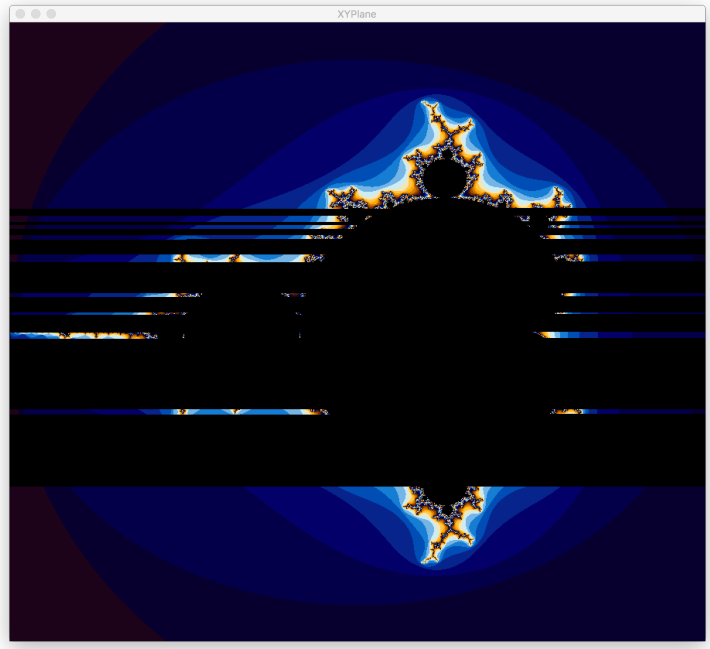
2012?

2017

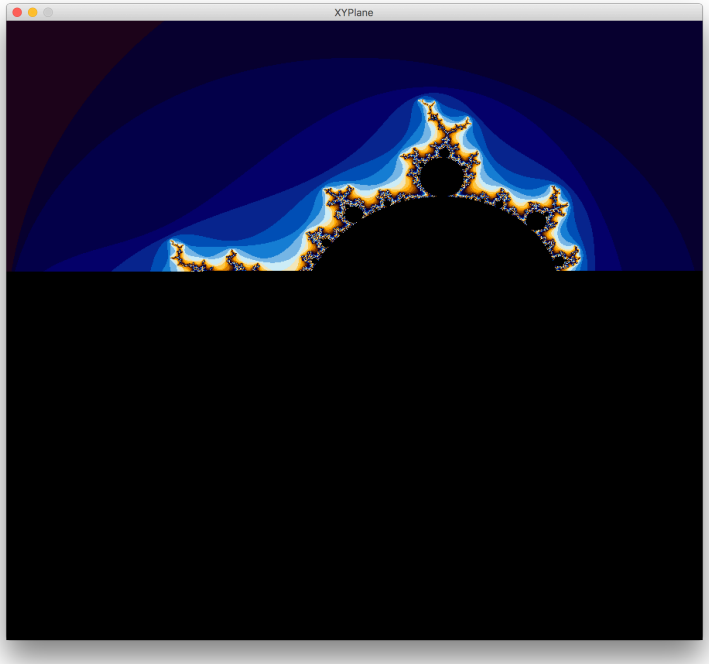
1986: no Internet
no Amazon
no FB, no mobiles,...

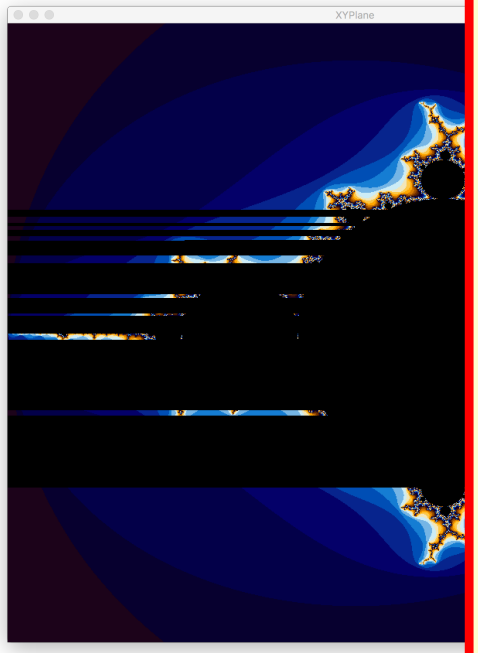
Seq vs Par



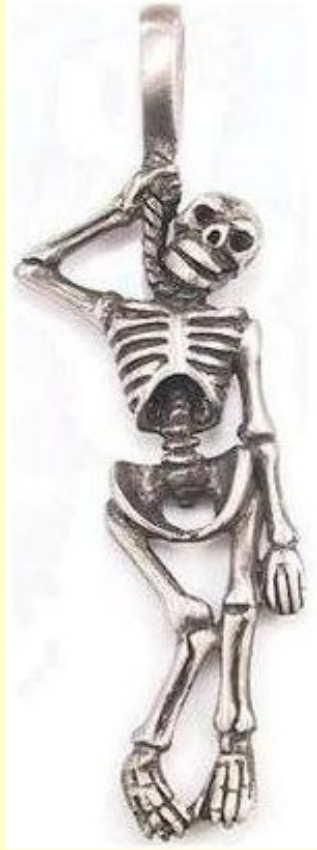


Seq vs Par

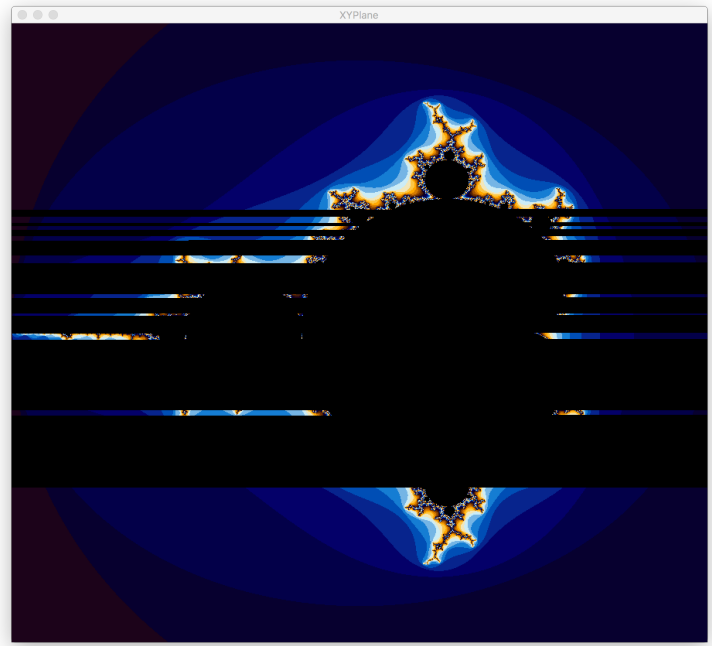




in Java or C++



Seq vs Par



In FP: Once a variable is created, it is assigned a value and then never changed again $\Rightarrow$ no synchronisation

(Andrew's second favourite feature of C++)

Types

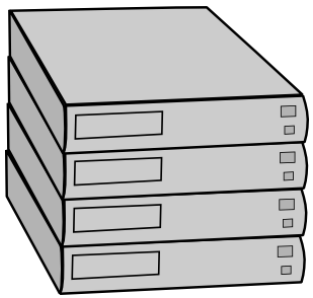
- Base types

  `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

An Http Request



Server

GET request

webpage

POST data

Browser

```
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Scanner;

public class URLReader {

    public static String readURL(String sUrl) {
        StringBuilder buf = new StringBuilder();
        Scanner in = null;

        try {
            URL url = new URL(sUrl);
            in = new Scanner(url.openStream());

            while (in.hasNextLine()) {
                buf.append(in.nextLine() + "\n");
            }
            return buf.toString();

        } catch (MalformedURLException e) {
            System.err.println(e);
        } catch (IOException e) {
            System.err.println(e);
        } finally {
            if (in != null) {
                in.close();
            }
        }
        return null;
    }
}
```

Coursework

- sorry, I might have been a bit wordy:
  CW description is 7 pages, but I only needed
  < 100 loc for all the CW

- there is email feedback when pushing code to github

- we want you to learn FP: no vars, no mutable datastructures, e.g. `ListBuffer`

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...

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

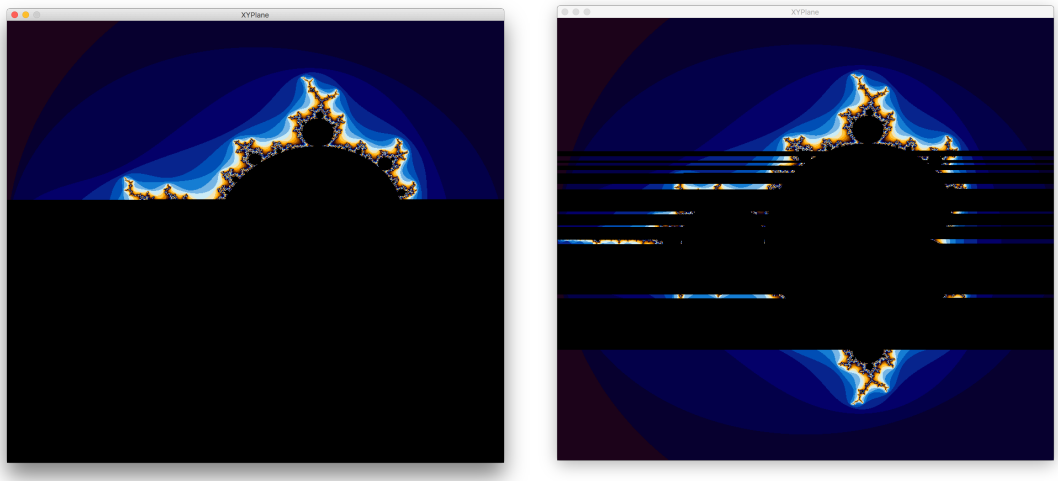
THANK YOU! You made me change my coding perspective. Because of you, I figured out the next one...

Me: 

Conclusion

- Scala is still under heavy development (the compiler is terribly slow)
- `http://www.scala-lang.org/`
- it is a rather **deep** language...i.e. gives you a lot of rope to shoot yourself
- learning functional programming is not easy...when you have spent all of your career thinking in a procedural way it is hard to change
- hope you have fun with the coursework

Questions?



My Scala Office Hours: Thursdays 11 – 13