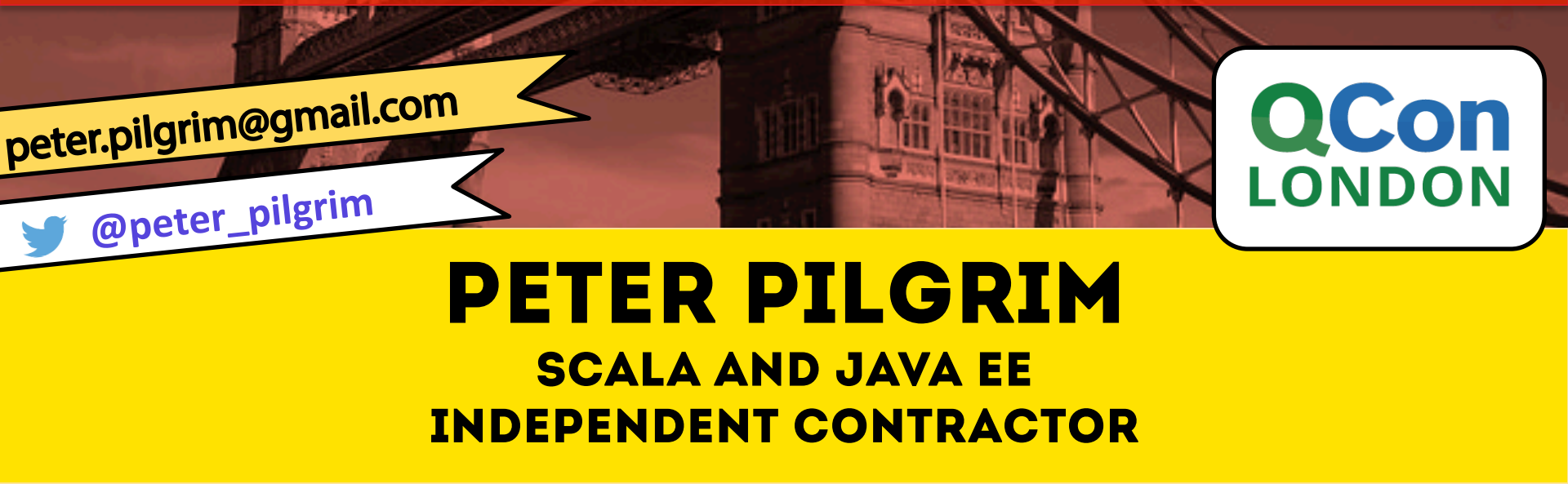
SCALA IN THE ENTERPRISE

Getting FuNkEd Up with the JVM



peter.pilgrim@gmail.com

@peter_pilgrim

QCon LONDON

PETER PILGRIM

SCALA AND JAVA EE

INDEPENDENT CONTRACTOR


Stand on the right

Agenda

- Technical Focus
- Adoption of Scala
- Best Practice
- Magnet Compass

Creative Commons 3.0

Attribution-NonCommercial-NoDerivs 3.0 UK: England & Wales (CC BY-NC-ND 3.0 UK)

https://creativecommons.org/licenses/by-nc-nd/3.0//

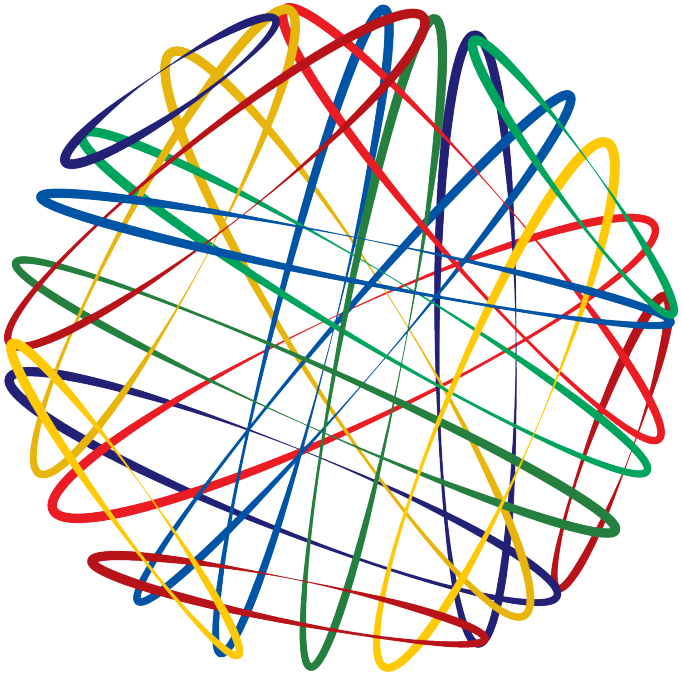
- **Attribute** — you must always attribute the source
- **Share** — copy and redistribute the material in any medium or format
- **No Deriv**— remix, transform, and build upon the material, but you cannot distribute modified

The licensor cannot revoke these freedoms as long as you follow the license terms.

About Me

- **Java Champion**
- Java EE and Scala developer independent contractor
- Digital / Finance / E-commerce
- Working on the JVM since 1998

About Me



Java™ Champions

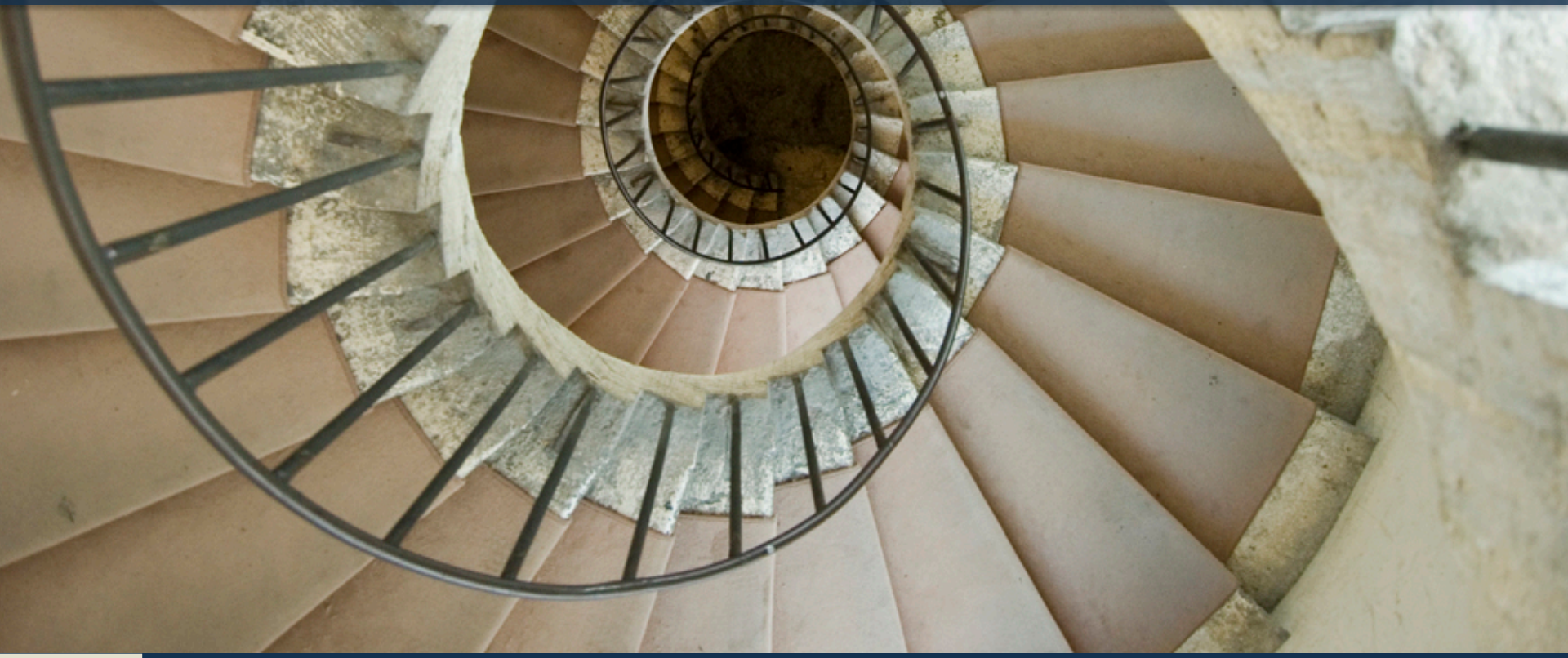
Java

transform
customer-centred change

GOV.UK

Scala

Scala:
object-functional language




Design
Bug Fix
DONE!
PRIORITY
ID

#1 Unlearn

“Learn a new language, people!”

Expand your horizons

Dianne Marsh

Netflix Director, Engineering Tools

http://diannemarsh.com/


ESSO SERVICE
GAS STATION

Scala Test #1

```
import collection.mutable.Stack
import org.scalatest._

class StackSpec extends
    FlatSpec with Matchers {
   /* ... */
}
```

Scala Test #2

```
class StackSpec extends FlatSpec with Matchers {
  "A Stack" should
  "pop values in last-in-first-out order" in
  {
    val stack = new Stack[Int]
    stack.push(1)
    stack.push(2)
    stack.pop() should be (2)
    stack.pop() should be (1)
  }
  /* ... */
}
```

Scala Test #3

```
class StackSpec extends FlatSpec with Matchers {
  /* ... */
  it should
  "throw NoSuchElementException if an empty
stack is popped"
  in {
    val emptyStack = new Stack[Int]
    a [NoSuchElementException]
    should be thrownBy {
      emptyStack.pop()
    }
  }
}
```

How to avoid null pointers in Scala Programming?

```
val greeting1: Option[String] =
    Some(”QCL15")

val greeting2: Option[String] = None
```

Pattern Matching

```
def checker(p : Option[String]): String = {
    p match {
        case Some(v) => v
        case _ => "Unexpected"
    }
}

checker(greeting1)  // "Hello QCL15"
checker(greeting2)  // "Unexpected"
```

Scala's generic immutable list collection type

```
val list: List[Int] = List(1,2,3,4)
list.foreach { x => println(x) }
// 1,2,3,4
```

Scala's option behaves like collection type #2

```
greeting1.foreach { x =>
    println(s"Work: ${x}" ) }
// Work: Hello QCL15

greeting2.foreach { x =>
    println(s"Work: ${x}" ) }
// *** nothing happens ***
```

#2 Unlearn More

"If you only ever learn one thing in Scala, learn the Collection classes"

Dick Wall

Former JavaPosse Podcaster

http://scalawags.com/


BUCHLA & ASSOCIATES BERKELEY CALIFORNIA
trig A
trig C
enable
PULSE OUTPUTS
strobed
continuous
offset
EXTERNAL INPUTS
STAGE ADDRESSING
Pulse
cv in
BUCHLA & ASSOCIATES
BERKELEY CALIFORNIA
MOD TYPE
MODU OSC
fm

Jump To Intermediate Level

```
class DigitalMicroService(
    warehouseClient: warehouseClient,
    applicationClient: ApplicationClient
) extends Logging with PricingFactor {

  def startApplication(productName: String,
    params: Map[String, String]): Future[AppId] = {
     applicationClient.startApplication(
       productName, params)
  }

  // ...
}
```

Async Service Methods

```
class DigitalMicroService(
  // ...
  def submit(applicationId: String): Future[Boolean] = {
    val future: Future[Boolean] = for {
      result <- applicationClient.submit(applicationId)
    } yield {
      result
    }
    future recover {
      case e: DuplicateSubmissionException =>
       logger.info("Application already submitted: " + e);
       true
    }
  }
}
```

Testing With Specifications #1

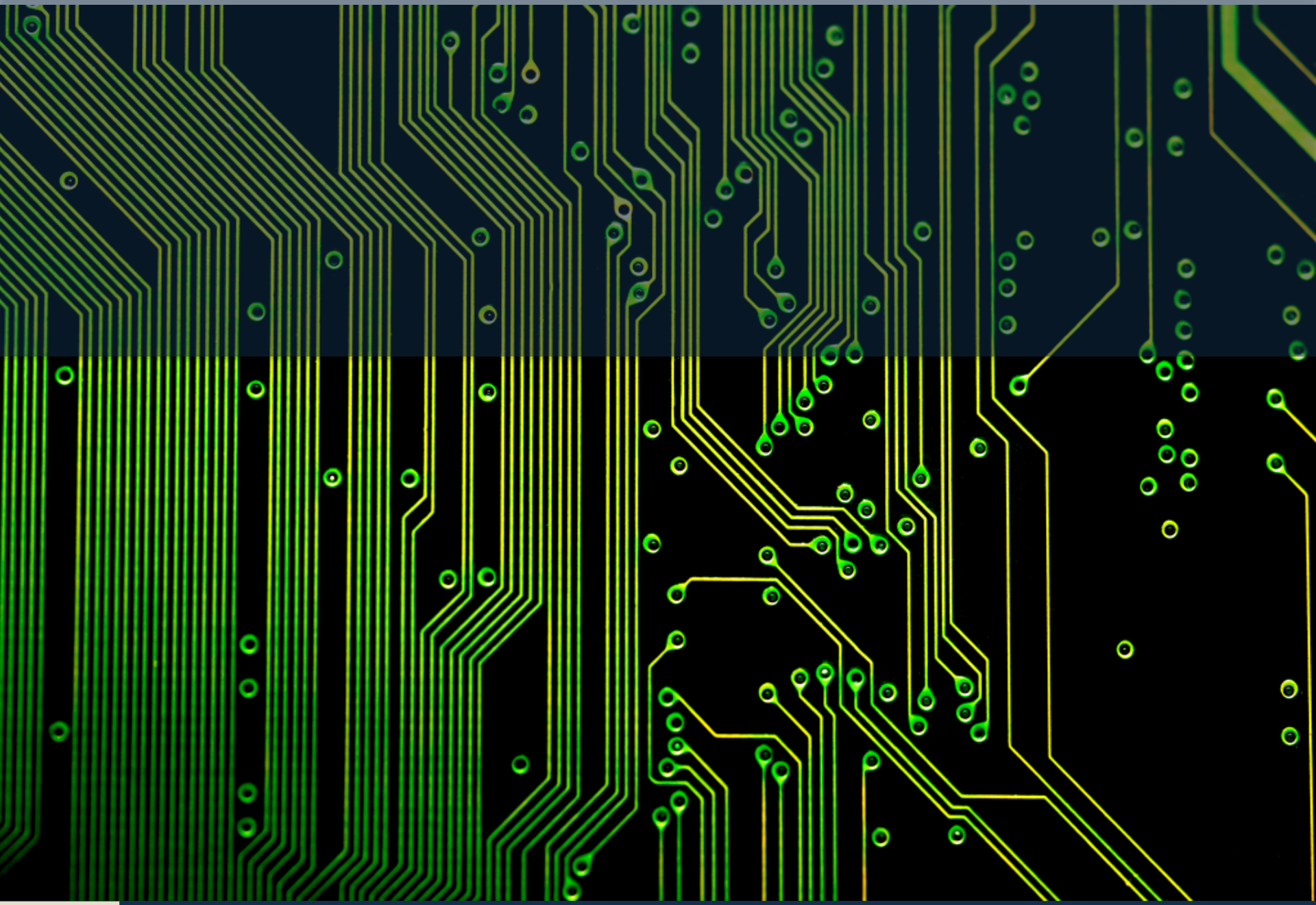
```
class ApplicationClientSpec extends
   Specification with Mocking
   with Futuristic with WsClient {

  val applicationClient =
   new ApplicationClient("/service", ws)

  "start application" should {
    "return the AppId" in {
      /* . . . */
    }
  }
}
```

Testing With Specifications #2

```
"start application" should {
   "return the AppId" in {

     val responseBodyContent =
        """{"id": "777"}"""
     setUpGlobalResponse(
        "/service/application", 200, responseBodyContent)
     response.body returns responseBodyContent

     /* . . . */
   }
 }
```

Testing With Specifications #2

```
 "start application" should { /* ... */
      mockRequestHolder.post(
        Matchers.eq(
          """{"productName":"digital-service","params":
{}}"""
          ))(any[Writeable[String]],
              any[ContentTypeOf[String]])
                  returns Future.successful(response)
      val responseBody =
        await(applicationClient.startApplication(
          ”digital-service", Map()))
      responseBody must_== AppId("777")
    }
  }
```



Digital Scala Adaptation

So a does a modern Digital Engineer learn a new language?

Digital Scala Adaptation

The answer is Yes.

Perhaps learn several languages and DSLs. Puppet, SBT, Grunt etc

#2 DailyWork2014 GOV.UK

- Develop Scala applications
- **Play Framework**, **REST**ful Services, **MongoDB**
- **Cucumber** testing: acceptance & functional
- **Collaborative design**, pairing and reviews
- SCRUM, Sometimes Kanban
- Physical storyboards, retros, planning
- Heavy dose of **UX** / User Centric
- **Government Digital Service**


Technics

#4 Scala Adoption

Highly competent & talented teams

Motivation is key

Developers drives the decision to learn Scala; already self-organising

#4 Scala Adoption

Somewhat confident and enthusiastic development talented teams

Delivery is key

Large responsibility for the end product; mixed Java and Scala

#4 Scala Adoption

Time sensitive teams

Time change is key

"We like everything you say about Scala, however we can't do this now."
Wait for the early adopters to finish.

#4 Scala Adoption

Lesser acquainted teams

Consequences, uncertainty and fear are keys

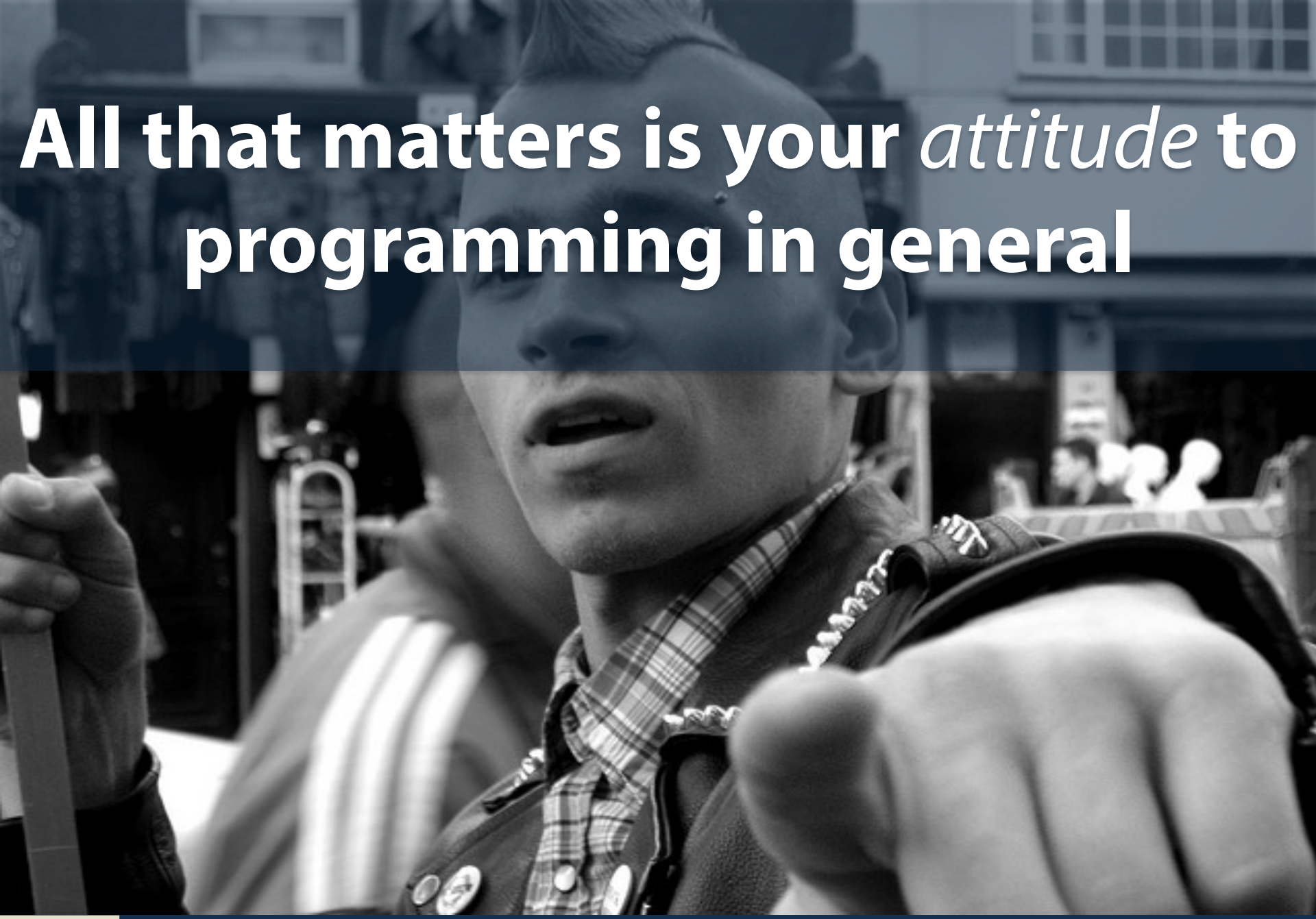
Overly concerned with risk : politics, society, technology and the unknown-unknowns.

“How does X [Scala] help the business going forward?”

Functional Programming is here

Whether Scala or Java or something entirely alternative

All that matters is your *attitude* to programming in general



Let's look at Java SE 8

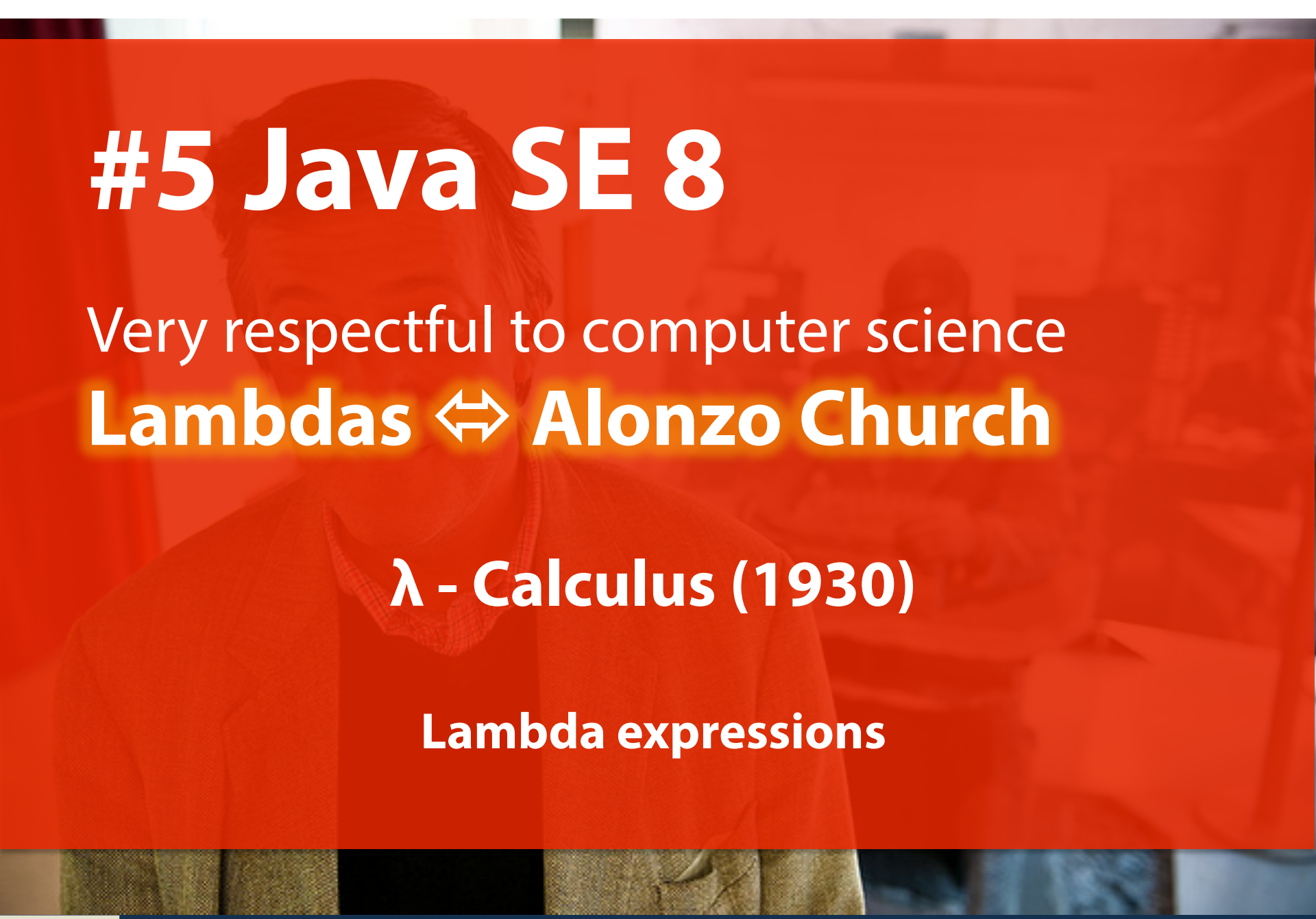
#5 Java SE 8

Very respectful to computer science

Lambdas ⇔ Alonzo Church

λ - Calculus (1930)

Lambda expressions



#5 Java SE 8

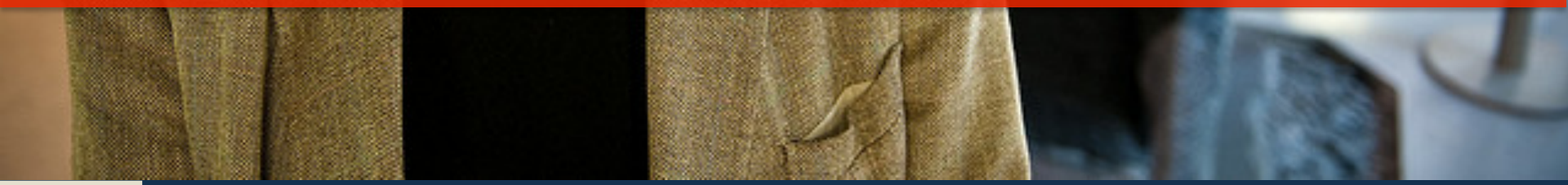
Function Interfaces

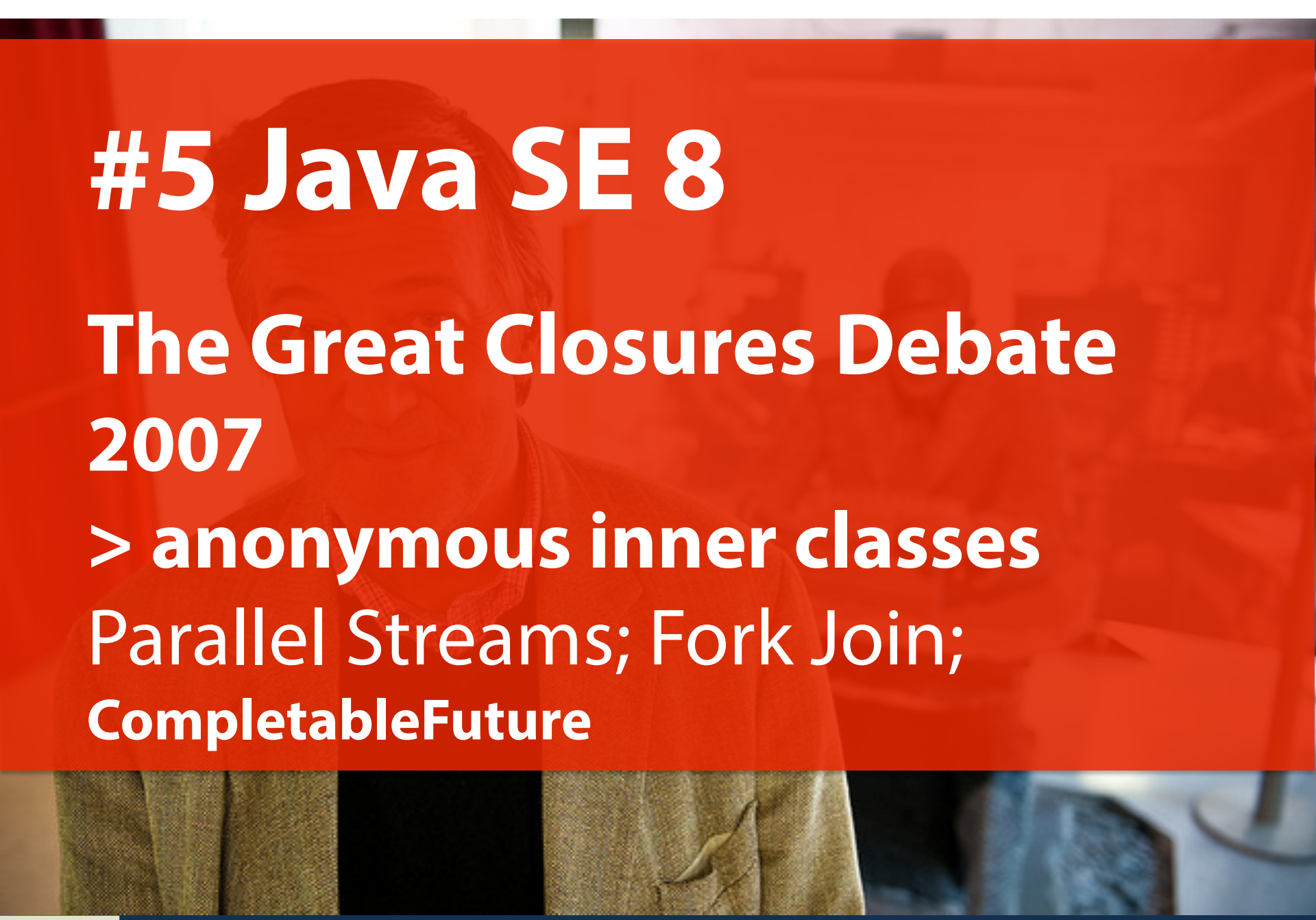
Lambda expressions **Streams**

Default interface methods

Predicates, Filters, Suppliers, **Combiners**,

Collectors, Reducers, Utilities





#5 Java SE 8

The Great Closures Debate 2007

> anonymous inner classes

Parallel Streams; Fork Join;

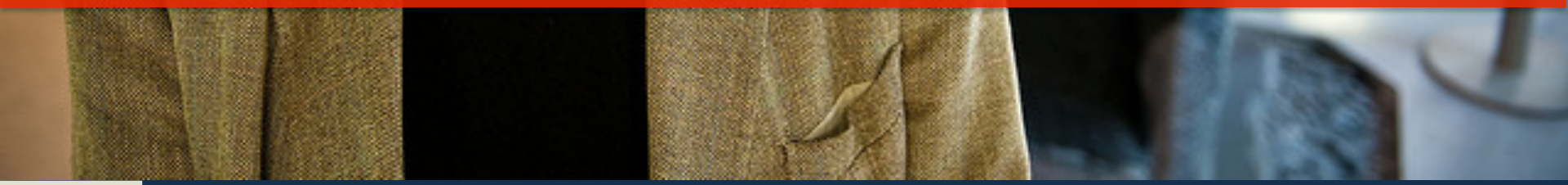
CompletableFuture

#5 Java SE 8

The majority of application programming state of the art (before modularity comes along).

Filter. Map. Reduce. 2015

Processing data over Java Collections and Streams

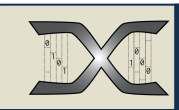


Lambdas Java SE 8

```
Function<Double,Double> double= x -> x + x

Function<Double,Double> square = x -> x * x

Predicate<String> qconFilter =
   (x: String) ->  x.contains("qcon”)
```

Lambdas Java SE 8

```
// However there is no equivalent to Scala
Function<Double,Double, Double> f1
   = (x + 2) * (y -3 )

// Instead you still define Java interface
interface Multiple<T,U,R> {
    R apply(T t, U u );
}

Multiple<Double,Double,Double> f1
  = (x + 2) * (y - 3 )
```



Lambdas Java SE 8

```
Arrays.asList(
  "Adventure","Romance","QCON London","science")
  .stream()
  .map( s -> s.toLowerCase())
  .filter(qconFilter)
  .forEach( e -> System.out.println(e) )

List<String> list = Arrays.asList(
  "Adventure","Romance","QCON London","science")

list.stream() .map(String::toLowerCase)
  .filter(qconFilter)
  .forEach( System::out::println )
```

Design pattern #1 Replace Factory Interface with Lambda

```
class TradeFileRetriever {
   Function<Void,JSch> factory =
      () -> new JSch();

   void setFactory(
       Function<Void,JschSession> factory)
      return this.factory = factory;
   }
   //  jsch = this.factory.apply();
}
```

Lambda Factory #2– Mock Out Hard Hat Frameworks

```
import com.jcraft.jsch.*

JSch mockJsch = mock(Jsch.class)
Session mockSession = mockSession(Session.class)

TradeFileRetriever retriever =
    new TradeFileRetriever();
retriever.setFactory( () -> mockJsch )

when(mockJsch.getSession()).thenReturn(
   mockSession )
```

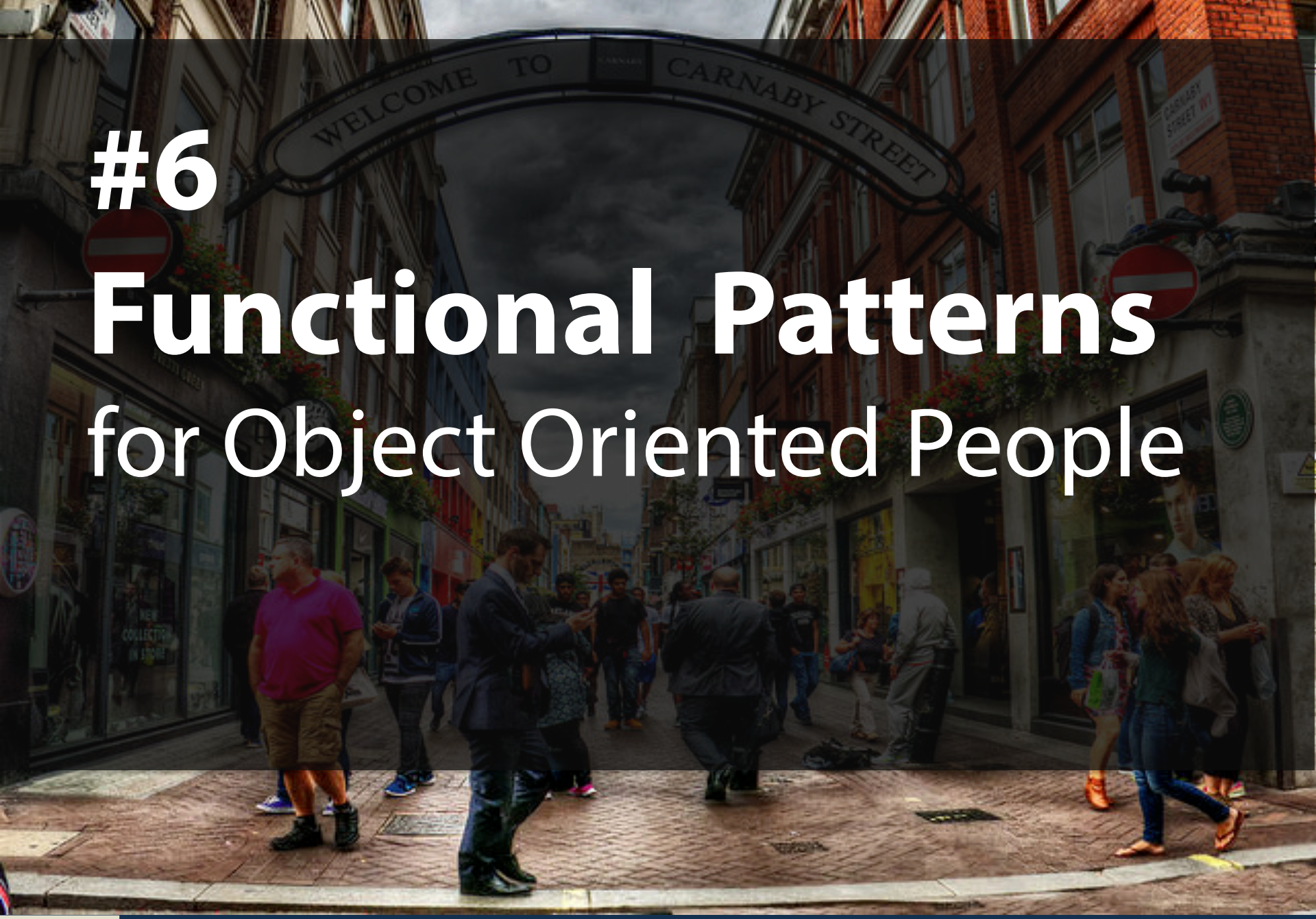
Lambda Flatmap – I just want to process only trade Details

```
class TradeRecord {
    List<String,TradeDetail> tradeDetailMap
        = new TreeMap<>();
    // ...
}

List<TradeRecord> tradeRecords = retrieveSome();

tradeRecords.stream()
     .flatMap( t ->
          t.getTradeDetails().entrySet().stream())
     .filter( s -> s.getKey().startsWith("ASIA") )
     .forEach( s ->
       System.out.printf("%s -> %s",
          s.getKey(), s.getValue() ));
```

Lambda Flatmap – Emergency Request Positions with Hsbc

```
IntSummaryStatistics hsbcFundSummary =
    tradeRecords.stream()
     .flatMap(
       t -> t.getTradeDetails().entrySet().stream())
     .filter(
       s -> s.getValue().getCurrency().equals("GBP"))
     .filter(
       s.getValue().getCounterparty().equals("HSBC"))
     .collect(
          Collectors.summarizingInt(
             s -> s.getValue().getQuantity()));
```


WELCOME TO CARNABY STREET
CARNABY STREET W1



#6
Functional Patterns
for Object Oriented People

Prefer First-Class Functions over Functors

```
// Java Classic
class Comparator<TradeDetail,TradeDetail> {
   public int compare(TradeDetail p1,
       TradeDetail p2) {
      return p1.getCurrency().compareTo(
        p2.getCurrency());
}
```

Prefer First-Class Functions over Functors

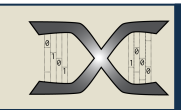
```
// Scala
val currencyComparator =
   (p1:TradeDetail,p2:TradeDetail) =>
      p1.currency.compareTo(p2.currency)

trades.sortWith( currencyComparator )
```

Prefer Internal Iterators over External Interators

```
static int summation( List<Products> products) {
   int sum = 0;
   for (Product e: products) {
      sum += e.getPrice();
   }
   return sum;
}
  // Java
```

Prefer Internal Iterator over External Means

```
// Scala: internal iteration

products.map( x => x.price ).sum

products.foldLeft(0) ( _.price + _.price )

// Soft (closure) white!
var sum = 0
products.foreach {
    e => sum = sum + e.price }
```



Prefer Functions over the Command Pattern

```
// Java
interface Command {
    void execute(Item item)
}

class Ingest implements Command { ... }
class Compress implements Command { ... }
class Enrich implements Command { ... }
```

Prefer Functions over the Command Pattern

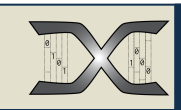
```
// Scala
val ingest = (item: Item) => { ... }
val compress = (item: Item) => { ... }
val enrich = (item: Item) => { ... }

val items: List[Item] = ...
items.map { x =>
   ingest(x); compress(x); enrich(x); x }
```

Prefer Functions over the Command Pattern

```
// Scala : NB the order of composition!
val evenBetter =
  enrich compose compress compose ingest

val items: List[Item] = ...
items.map { x => evenBetter(x); x }
```

Wherever you Can: Filter, Map and Reduce

```
val items: List[Item] = ...

val discountShoppingCartPrice =
   items
      .filter( item => item.price > 25.0)
      .map( item => item.nominal )
      .reduce(
          (total, price )
            => total + price * 0.90 )
```



Function Builders

```
def discounter(percent: Double):
   (Double) => Double = {
   require( percent >= 0.0 &&
            percent <= 100.0 )

   (price: Double) =>
      price - ( price * percent / 100.0)
}
```

Function Builders

```
val tenPercenter = discounter(10)
val twentyPercenter = discounter(20)

val finalCartPrice =
   if (customer.loyaltyPoints >= 3000 )
      gardenItemPrices map tenPercenter sum +
     foodItemsPrices map twentyPercenter sum
   else
     ( gardenItemPrices ::: foodItemPrices ) sum
```

Replace conditional logic with Partially Applied Functions

```
abstract class Item { val nominal: Double }
case class GardenItem(val nominal:Double)
extends Item
case class EnergyItem(val nominal:Double)
extends Item

val gardenTaxCode: PartialFunction[Any, Int] =
    { case i: GardenItem => i.nominal *  0.025 }

val energyTaxCode: PartialFunction[Any, Int] =
    { case i: EnergyItem => i.nominal *  0.015 }
```

Replace conditional logic with Partially Applied Functions

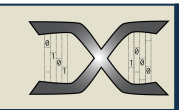
```
gardenTaxCode.isDefinedAt( GardenItem(100))
gardenTaxCode.isDefinedAt( "What is this?"))

val items = List(
  GardenItem(12.34), EnergyItem(73.99))

val gardenTax = items collect gardenTaxCode
```

Lean on to Tail Recursive Functions

```
def summation(xs: List[Int]): Int = {
  xs match {
    case x :: tail => x + summation(tail)
    case Nil => 0
  }
}
```

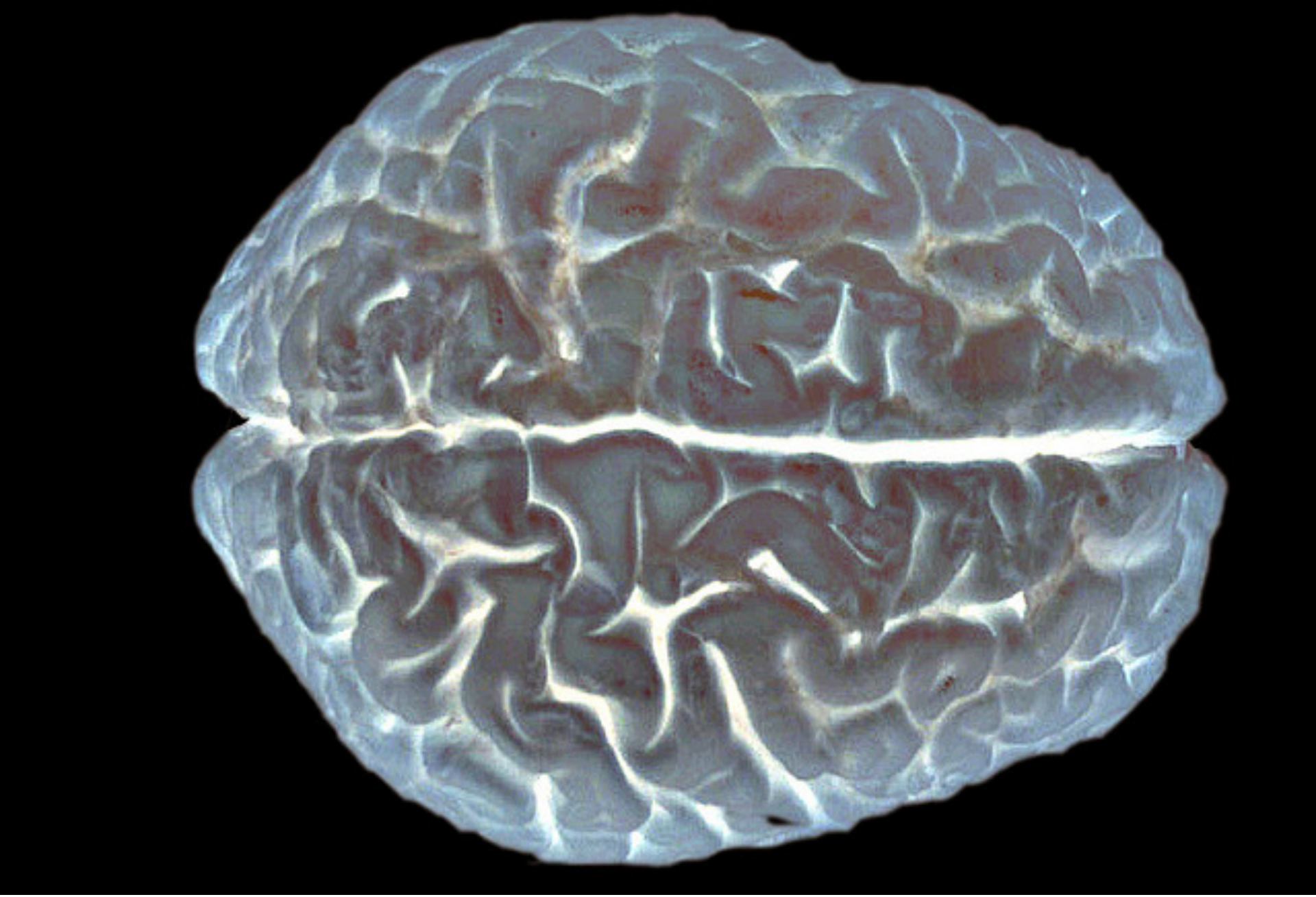
Lean on to Tail Recursive Functions

```
def summation(xs: List[Int]): Int = {
  @tailrec
  def sum0(xs: List[Int], acc: Int): Int = {
    xs match {
      case x :: tail => sum0(tail, acc + x)
      case Nil => acc
    }
  }

  sum0(xs, 0)
}
```




ng of kings

Let's Come Up For Air!



#7 Adventures?

Please extend your inspiration to

Recursive functions (Tail Rec)

Focused mutability

Futures and **Promises**

Control Flow

Domain Specific Languages

#Conclusion(1)

Scala in the Enterprise: *Done*

How far do you want to go (FP)?

Java SE 8 changes everything

Everyone is learning.

#Conclusion(2)

Avoid over reliance on type inference

Break code down

Less is more; Einstein's Simple

Spectrum of abilities, talent

Avoid silos if possible

#Conclusion(3)

Java has lots of frameworks, utilities

Scala has some great tools:
ScalaTest, IDE support

Developers work harder

Conciseness, readability, maintainability


E
SE
S
SW
W
NW
N
NE
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340

#Conclusion(4)

The Future of Scala

Macros, Modularisation

Compiler performance, Push to FP

Integration with Java Lambdas

Who knows really when or next?

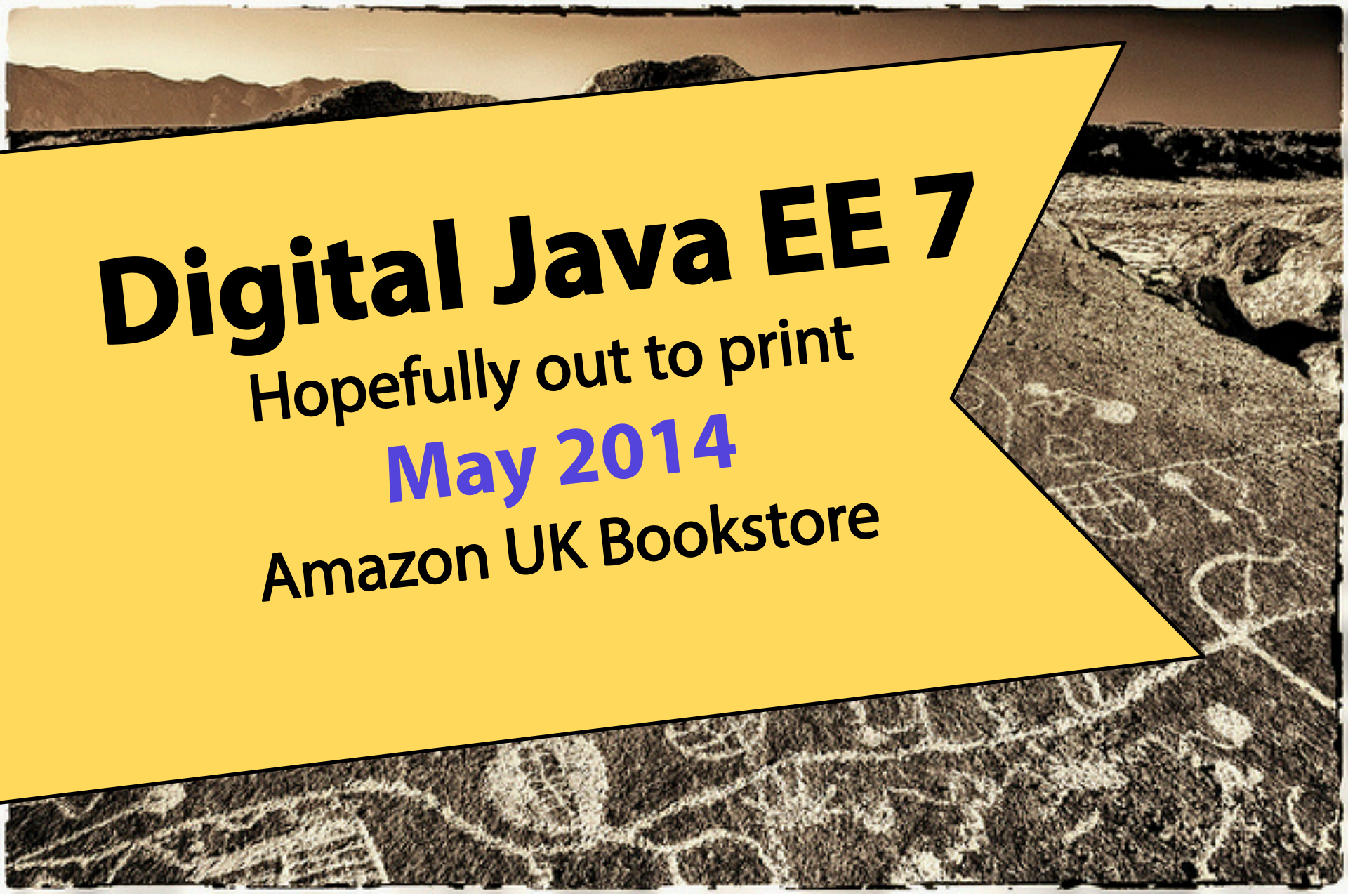
"Always a Left Field contender, there will be" Yodaspeak

#It Yours!

Get yourselves

FuNkEd Up

with the JVM


Digital Java EE 7
Hopefully out to print
May 2014
Amazon UK Bookstore

Thank You!

http://xenonique.co.uk/blog/

@peter_pilgrim

uk.linkedin.com/in/peterpilgrim2000/

peter.pilgrim@gmail.com

QCon LONDON

List of Creative Commons (2.0) photography attributions

https://flic.kr/p/dhf3FQ
https://www.flickr.com/photos/epsos/
epSos.de
Big Beautiful Face Statue in Tenerife

https://flic.kr/p/Pp93n
https://www.flickr.com/photos/spacesuitcatalyst/
William A. Clark Follow
Measurement
Measure twice, cut once.

https://flic.kr/p/opvVsg
https://www.flickr.com/photos/gregbeatty/
Greg Beatty Follow
Three Pillars
Nikon D800 16-35MM F4.0

https://flic.kr/p/81dXuT
https://www.flickr.com/photos/froboy/
Avi Schwab Follow
Equipment used for measuring planes of crystals

List of Creative Commons (2.0) photography attributions

https://flic.kr/p/ayqvZp
https://www.flickr.com/photos/throughpaintedeyes/
Ryan Seyeau Follow
1000 Faces of Canada # 0084 - Zombie Walk - Explore!
Shot during the annual Zombie Walk in Ottawa.

https://flic.kr/p/8XY4tr
https://www.flickr.com/photos/creative_stock/
Creativity 103
electronic circuit board

https://flic.kr/p/8Y2625
https://www.flickr.com/photos/creative_stock/
Creativity 103
computer motherboard tracks

https://flic.kr/p/2QHt7Q
https://www.flickr.com/photos/rosefirerising/
rosefirerising Follow
Pierre Curie, Piezo-Electroscope

List of Creative Commons (2.0) photography attributions

https://www.flickr.com/photos/code_martial/
https://flic.kr/p/7jmU5n
Tahir Hashmi Follow
Developer Death
10 Programmers and a UX Designer died all of a sudden in a war room situation. They were apparently working too hard.

https://www.flickr.com/photos/8268882@N06/
https://flic.kr/p/duwd1M
Peter Pilgrim
IMG_1481
European JUG leaders meeting in Devoxx 2013

https://www.flickr.com/photos/unicefethiopia/
https://flic.kr/p/dv4ooi
UNICEF Ethiopia Follow
Hamer mother and child South Omo Zone, SNNPR
Hamer mother and child South Omo Zone, SNNPR.
©UNICEF Ethiopia/2005/Getachew

List of Creative Commons (2.0) photography attributions

https://www.flickr.com/photos/15216811@N06/
https://flic.kr/p/baULpM
N i c o l a Follow
Tree - IMG_1242

https://flic.kr/p/dwCQ7t
https://www.flickr.com/photos/90585146@N08/
marsmetn tallahassee Follow
IDA .. Integro-Differential Analyzer (Sept., 1952) ...item 2.. Richard Dreyfuss: Technology Has 'Killed Time' -- "In geopolitics, the removal of time is fatal." -- "And you will give up, the protection of Republican Democracy." (July 19 2010) ...

https://flic.kr/p/6ZDbiZ
https://www.flickr.com/photos/ingmar/
Ingmar Zahorsky Follow
Traffic Jam
Accidents are common on the narrow streets going through the mountains of Nepal. When such an accident occurs, traffic is often halted for up to 3-4 hours.

https://flic.kr/p/FAskC
https://www.flickr.com/photos/mari1008/
mari_1008 Follow
traffic jam -B
Date: April,2th
Address: Jiangshu Rd,Shanghai,China.
Around 8:30 AM today, A little accident on YuYuan Rd and JiangSu Rd caused traffic jam.

List of Creative Commons (2.0) photography attributions

https://flic.kr/p/bpss6D
https://www.flickr.com/photos/76029035@N02/
Victor1558 Follow
Creative_Photoshop_HD_Wallpapers_laba.ws

https://flic.kr/p/mLxu3m
https://www.flickr.com/photos/astrangelyisolatedplace/
astrangelyisolatedplace Follow
Office test #1. Setup fully functional. A rare 1200 no longer in production. Test run soundtracked by the only #vinyl in the office; The Sesame Street Soundtrack - a surprise present by @idinesh #technics #turntable #1200 #ocdv1
via Instagram ift.tt/1hpeUEU

https://flic.kr/p/gLPhEk
https://www.flickr.com/photos/stevecorey/
Steve Corey Follow
Treasure Hunt, a short story (5 images)

List of Creative Commons (2.0) photography attributions

https://flic.kr/p/4JcerK
https://www.flickr.com/photos/16949884@N00/
Bömmel Follow
Ice Crystal
Ice is nice

https://flic.kr/p/dSsXiZ
https://www.flickr.com/photos/jeremyhall/
Jeremy Hall Follow
Called to Serve (suit and tie)

https://flic.kr/p/3enERy
https://www.flickr.com/photos/paolocampioni/
Paolo Campioni Follow
scala
Scendo (stair case spiral)

List of Creative Commons (2.0) photography attributions

https://flic.kr/p/m62gmK
https://www.flickr.com/photos/lata/
-p Follow
Abstraction I.
From my platycerium. (Guide: Voronoi diagram)

https://flic.kr/p/9F7Nnn
https://www.flickr.com/photos/followtheseinstructions/
Follow these instructions
The end ; Assignment: This time your assignment is to find or compose a scene that conveys the essence of the end.

https://flic.kr/p/6WSFR4
https://www.flickr.com/photos/62337512@N00/
anthony kelly Follow
big ben
big ben and underground sign

https://flic.kr/p/w7qef
https://www.flickr.com/photos/cobalt/
cobalt123 Follow
All We are Saying... (Thought for a New Year)
Peace Pasta from Annie's, with peas, of course.
Give Peace A Chance