

Software Craftsmanship Barcelona

2016



@bcnswcraft



#scbcn16

Sponsors



YOU CAN'T BUY

HAPPINESS

• BUT YOU CAN BUY A •

Kayak

AND THAT'S PRETTY CLOSE



**DON'T MESS WITH A KAYAKER.
WE KNOW PLACES WHERE
NO ONE WILL FIND YOU.**

Railways Programming

SCBCN'16

Content

- you
- me
- railways programming
- final remarks

thanks!

about you



Content

- you
- me
- railways programming
- final remarks

SPOILER

<input checked="" type="checkbox"/> you	2 slides
<input type="checkbox"/> me	71 slides
<input type="checkbox"/> railways programming	15 slides
<input type="checkbox"/> final remarks	21 slides
<hr/>	
	+ 112 slides

Avstrija
Italija

A

I

7

Ljubljana 85





Not Jim Carrey

@ItIsJimCarrey



 Follow

When you drink alcohol you are just borrowing happiness from tomorrow

RETWEETS

110

LIKES

95



3:08 PM - 10 Apr 2014



 110

 95

...

```
public static void happy(  
    java.util.List<String> stringList) {  
  
    for (String s : stringList) {  
        int i = (50 / Integer.valueOf(s)) - 1;  
        System.out.println(" -> " + i);  
    }  
  
}
```



```
public static void happy(  
    java.util.List<String> stringList) {  
  
    for (String s : stringList) {  
        int i = (50 / Integer.valueOf(s)) - 1;  
        System.out.println(" -> " + i);  
    }  
  
}
```

```
public static void happy(  
    java.util.List<String> stringList) {  
  
    for (String s : stringList) {  
        int i = (50 / Integer.valueOf(s)) - 1;  
        System.out.println(" -> " + i);  
    }  
}
```

A medium shot of a man with dark brown hair, wearing thin-framed glasses and a mustache, smiling slightly. He is wearing a dark, zip-up jacket. The background shows a sandy beach and the ocean under a clear sky.

IDIOOOOOT

```
public static void sad(java.util.List<String> stringList) {  
    for (String s : stringList) {  
        try {  
            int parsed = Integer.valueOf(s);  
            try {  
                int div = 50 / parsed;  
                int x = div - 1;  
                System.out.println(" --> " + x);  
            } catch (ArithmetcException ae) {  
                // What do we do ?!  
            }  
        } catch (NumberFormatException nfe) {  
            // What do we do ?!  
        }  
    }  
}
```

```
public static void sad(java.util.List<String> stringList) {  
    try {  
        try {  
            for (String s : stringList) {  
                int parsed = Integer.valueOf(s);  
                int div = 50 / parsed;  
                int x = div - 1;  
                System.out.println(" --> " + x);  
            }  
        } catch (ArithmetcException ae) {  
            // What do we do ?!  
        }  
    } catch (NumberFormatException nfe) {  
        // What do we do ?!  
    }  
}
```



Mario Fusco
@mariofusco



Follow

That's what Java devs do in checked exceptions catch blocks ... and why I believe they are just useless verbosity

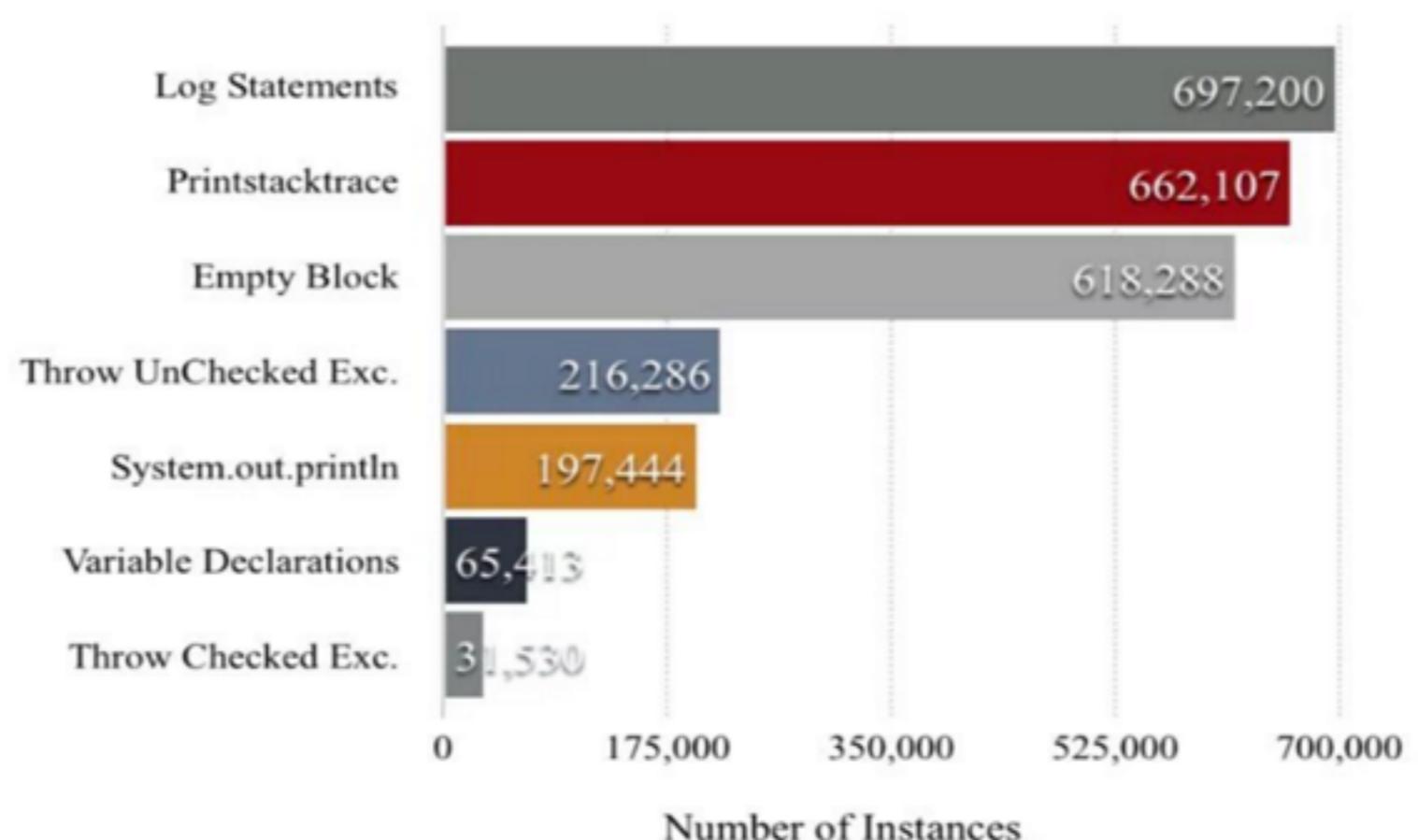


Figure 2: Top Operations performed in Checked Exception catch blocks (GH).



Mario Fusco @mariofusco · Jun 6

That's what Java devs do in checked exceptions catch blocks ... and why I believe they are just useless verbosity

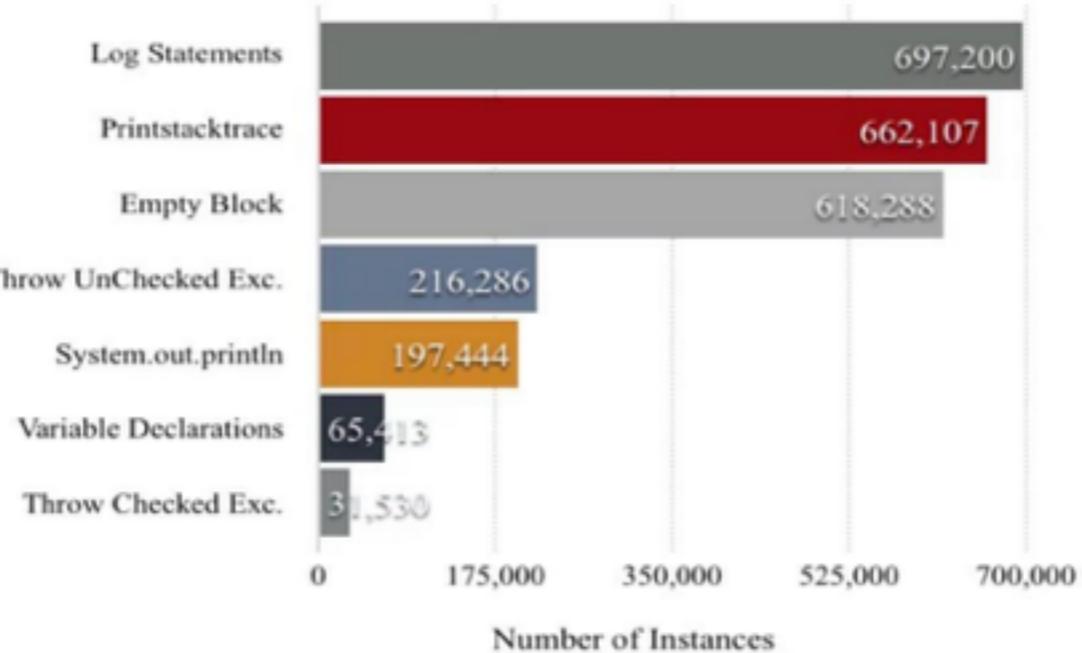


Figure 2: Top Operations performed in Checked Exception catch blocks (GH).

430 294



Peter Lawrey

@PeterLawrey



Following

@mariofusco it tells me developers have a hard enough time getting the happy path to work let alone figure out what to do if something fails

RETWEET

1

LIKES

5



11:14 AM - 6 Jun 2016

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf("") + java.lang.Integer.MAX_VALUE)
res9: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf("") + java.lang.Integer.MAX_VALUE)
res9: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483647")
res10: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf("") + java.lang.Integer.MAX_VALUE)
res9: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483647")
res10: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483648")
```

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf("") + java.lang.Integer.MAX_VALUE)
res9: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483647")
res10: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483648")
java.lang.NumberFormatException: For input string: "2147483648"
  at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
  at java.lang.Integer.parseInt(Integer.java:587)
  at java.lang.Integer.valueOf(Integer.java:766)
  ... 33 elided
```

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf("") + java.lang.Integer.MAX_VALUE)
res9: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483647")
res10: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483648")
java.lang.NumberFormatException: For input string: "2147483648"
  at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
  at java.lang.Integer.parseInt(Integer.java:587)
  at java.lang.Integer.valueOf(Integer.java:766)
  ... 33 elided
```

```
scala> java.lang.Integer.valueOf("asdf")
```

```
scala> java.lang.Integer.valueOf(23)
res8: Integer = 23
```

```
scala> java.lang.Integer.valueOf("") + java.lang.Integer.MAX_VALUE)
res9: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483647")
res10: Integer = 2147483647
```

```
scala> java.lang.Integer.valueOf("2147483648")
java.lang.NumberFormatException: For input string: "2147483648"
  at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
  at java.lang.Integer.parseInt(Integer.java:587)
  at java.lang.Integer.valueOf(Integer.java:766)
  ... 33 elided
```

```
scala> java.lang.Integer.valueOf("asdf")
java.lang.NumberFormatException: For input string: "asdf"
  at java.lang.NumberFormatException.forInputString(NumberFormatException.java:65)
  at java.lang.Integer.parseInt(Integer.java:580)
  at java.lang.Integer.valueOf(Integer.java:766)
  ... 33 elided
```

```
scala> █
```



Münchner
Oktoberfest
feiert holl

BOXE
3

MUUUU

```
override def update(areaId: AreaId, entityId: EntityId,  
// validate Command <-- application error  
// update DB           <-- app errors and/or sys errors  
// report Event        <-- sys errors  
}
```



Münchner
Oktoberfest
feiert holl

BOXE
3

MUUUU







Types

```
class PersonBuilder {  
    Person build(String name, String surname) {  
        ...  
    }  
}
```





Types

```
class PersonBuilder {  
  
    Person build(Name name, Surname surname) {  
        ...  
    }  
  
}
```



hotline

STONE
FREE

1 Firs



Func Prog

Pure Functions

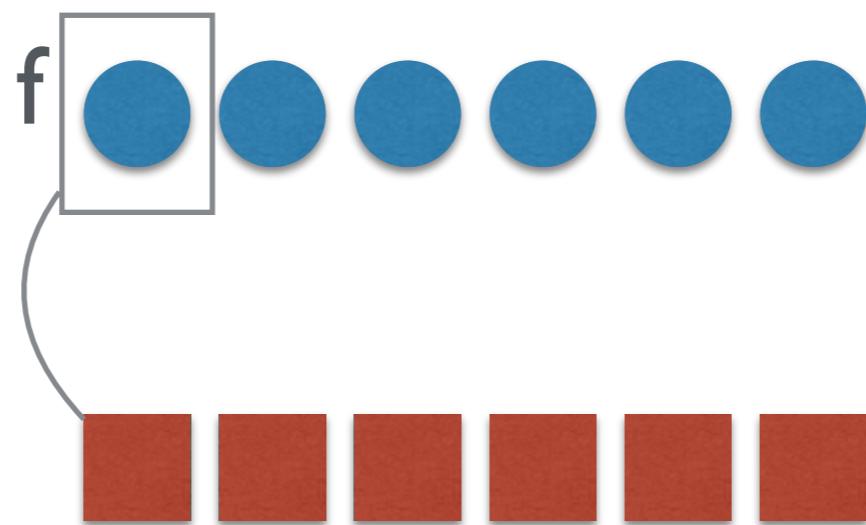
no side effects

if not used, remove it

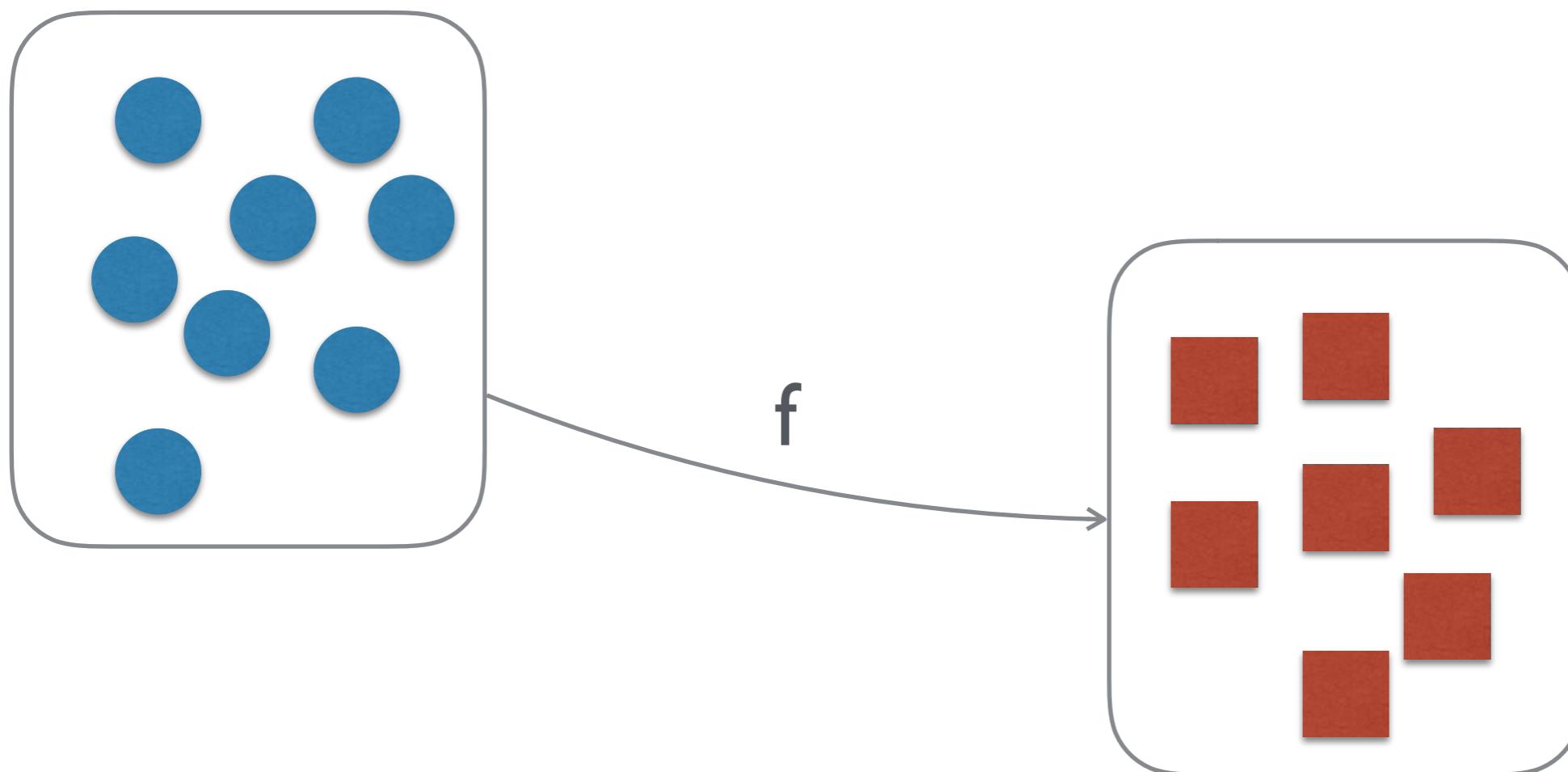
fixed in — fixed out

Don't throw exceptions!

map

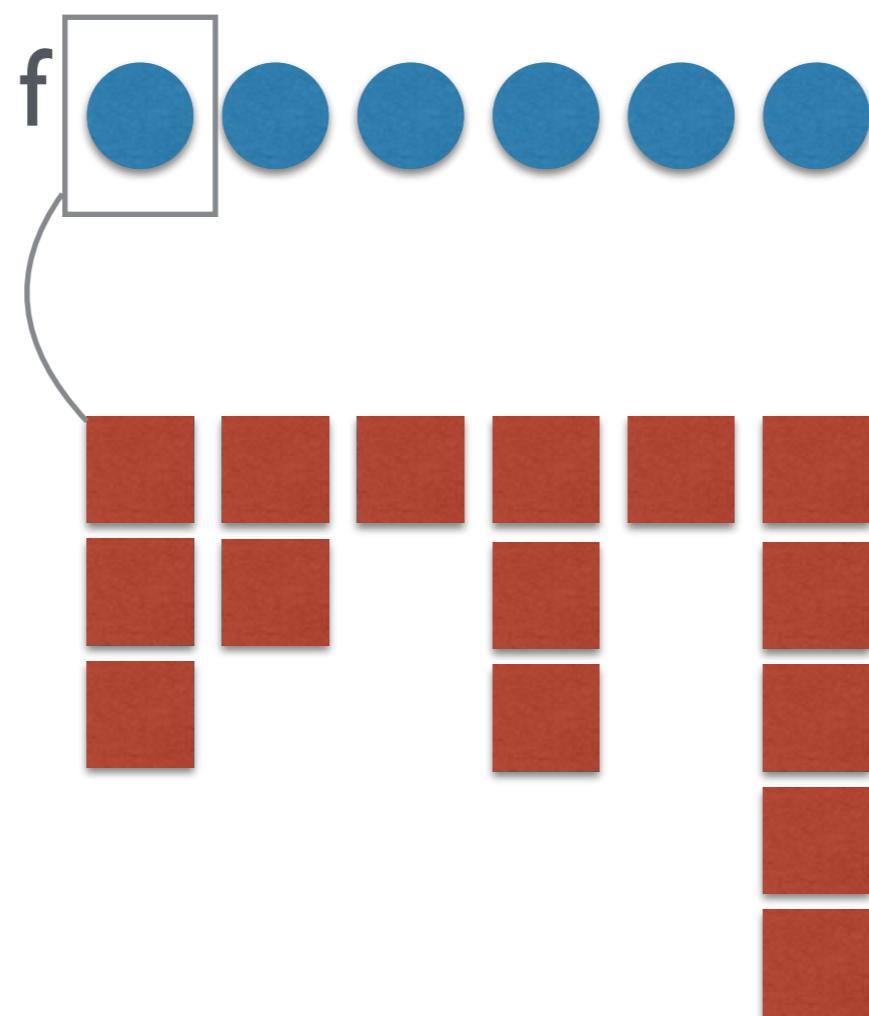


map

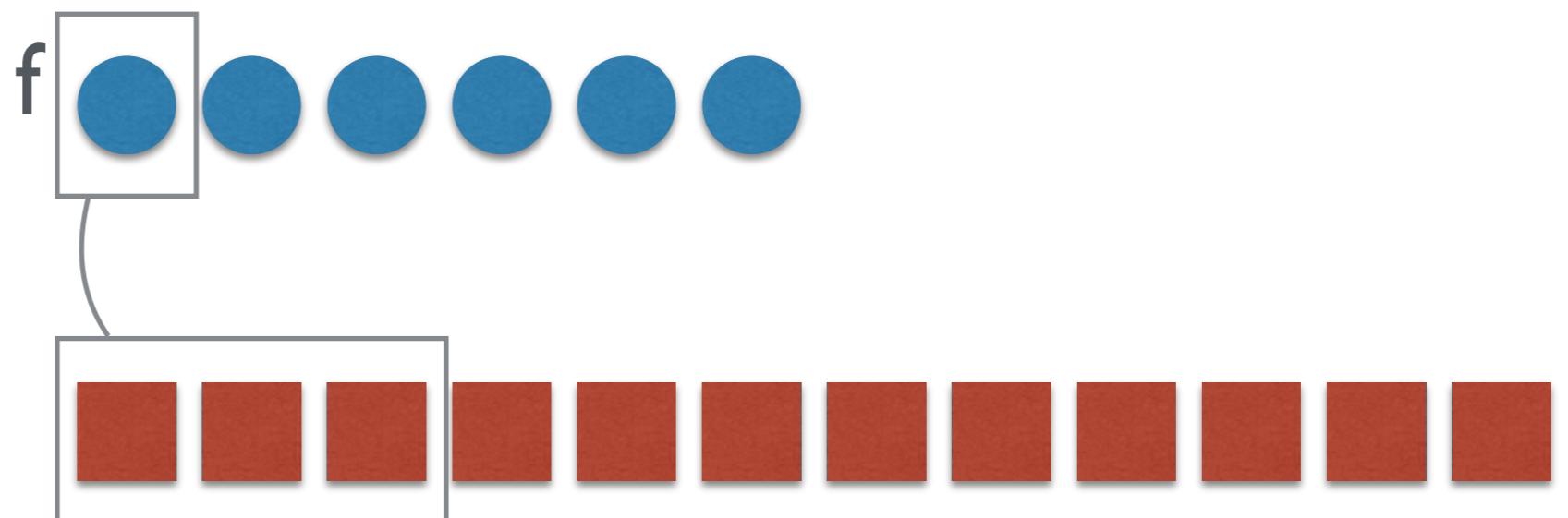


map revisited

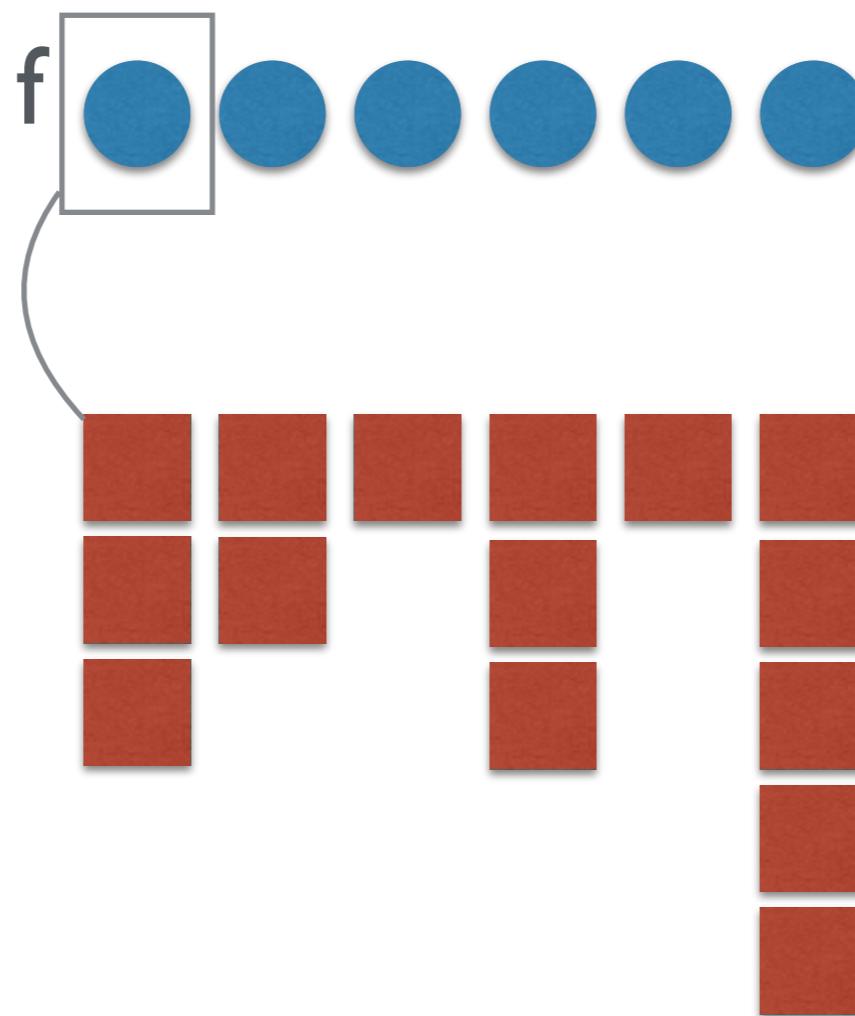
“Hi, this is Ignasi”.tokenize



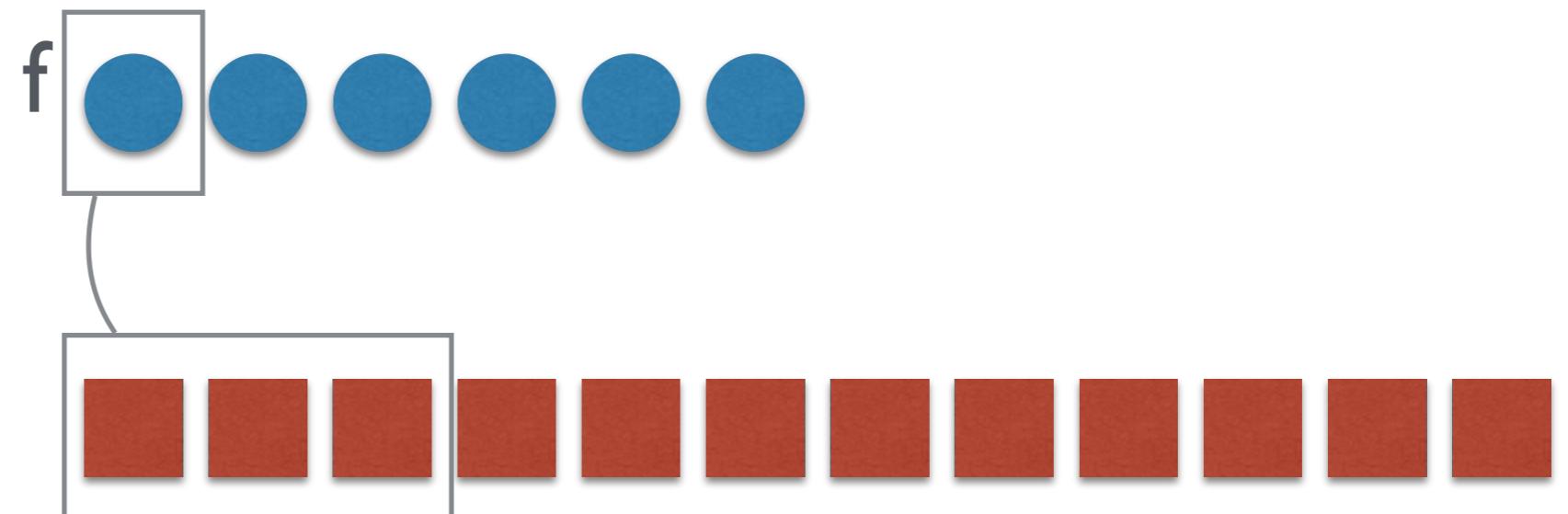
map revisited



map



flatMap





hotline

STONE
FREE

1 Firs



A MONAD IS JUST A
MONOID IN THE
CATEGORY OF
ENDOFUNCTORS.

WHAT'S THE PROBLEM ?



Did you just tell me to go
fuck myself?

I believe I did, Bob.

hotline

STONE
FREE

1 Firs



lino
to First Class



#5
RENTON



Trainspotting





A MONAD IS JUST A
MONOID IN THE
CATEGORY OF
ENDOFUNCTORS.

WHAT'S THE PROBLEM ?



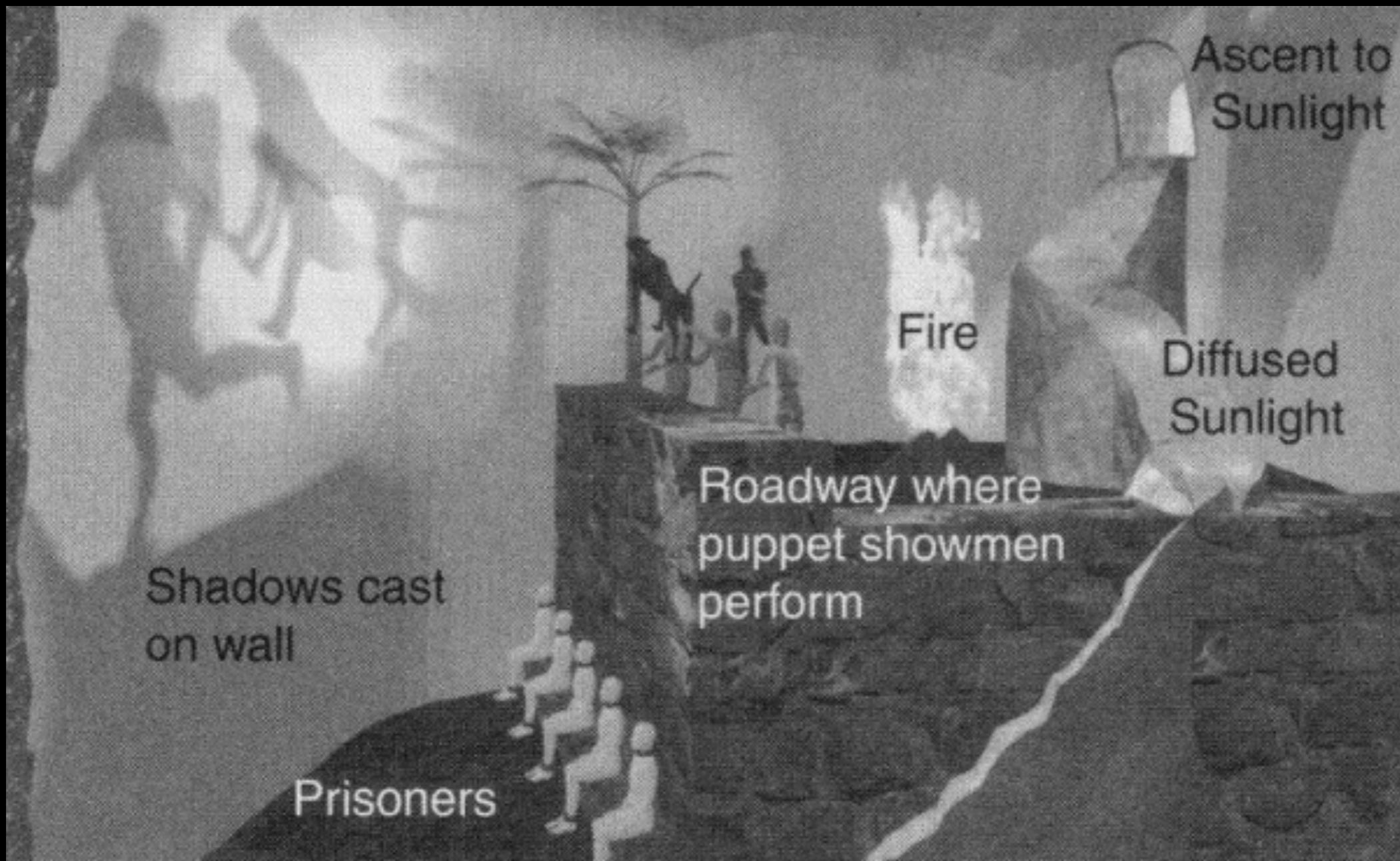
DOOOORK











```
public interface Try<T> {  
  
    <R> Try<R> map(Function<T, R> f);  
  
    <R> Try<R> flatMap(Function<T, Try<R>> f);  
  
    T get();  
  
    boolean isSuccess();  
  
    default boolean isFailure() { return !isSuccess(); }  
}
```

```
public static <T> Try<T> to(Supplier<T> f) {  
    try {  
        return new Success<T>(f.get());  
    } catch (Throwable t) {  
        return new Failure<T>(t);  
    }  
}
```

```
def railwayOperation(i: String) = {  
    for {  
        valid <- parse(i)  
        parsed <- authorize(valid)  
        persisted <- persist(parsed)  
        response <- forAUser(persisted)  
    } yield response  
}
```

```
static final Function<String, Try<Integer>> toInt =  
    x -> Tries.to(() -> Integer.valueOf(x));
```

```
static final Function<Integer, Try<Integer>> divide50 =  
(x) -> Tries.to(() -> 50 / x);
```

```
static final Function<Integer, Integer> minusOne =  
    x -> x - 1;
```





Park



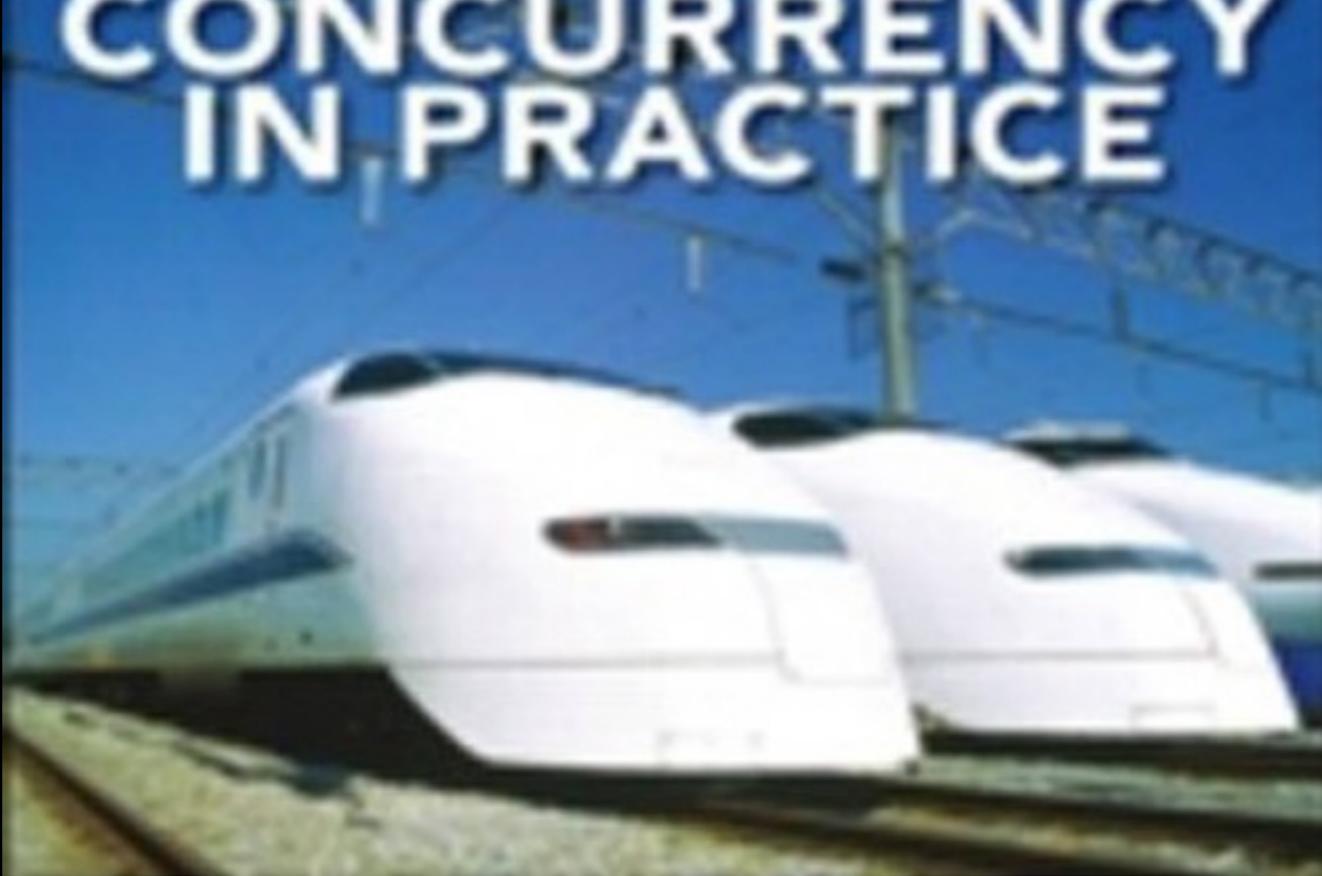


BRIAN GOETZ



WITH TIM PEIERLS, JOSHUA BLOCK,
JOSEPH BOWIEER, DAVID HOLMES,
AND DOUG LEA.

JAVA CONCURRENCY IN PRACTICE





Learning Concurrent Programming in Scala

Learn the art of building intricate, modern, scalable concurrent applications using Scala

Foreword by Martin Odersky, professor at EPFL, the creator of Scala

Aleksandar Prokopec

[PACKT] open source[®]
PUBLISHING

Try vs Promises

- Op that succeeds or fails
- Op that *eventually* succeeds or fails.

Park





日本新聞

企業春闘速報
人とおもしろスタート!

10レースは、アシの面で決まり



A close-up photograph of a man with dark brown hair and a mustache, wearing a black zip-up jacket. He has a wide-eyed, open-mouthed expression of surprise or shock. The background is a blurred beach scene with the ocean and sky.

WTF! ? ! ?





Content

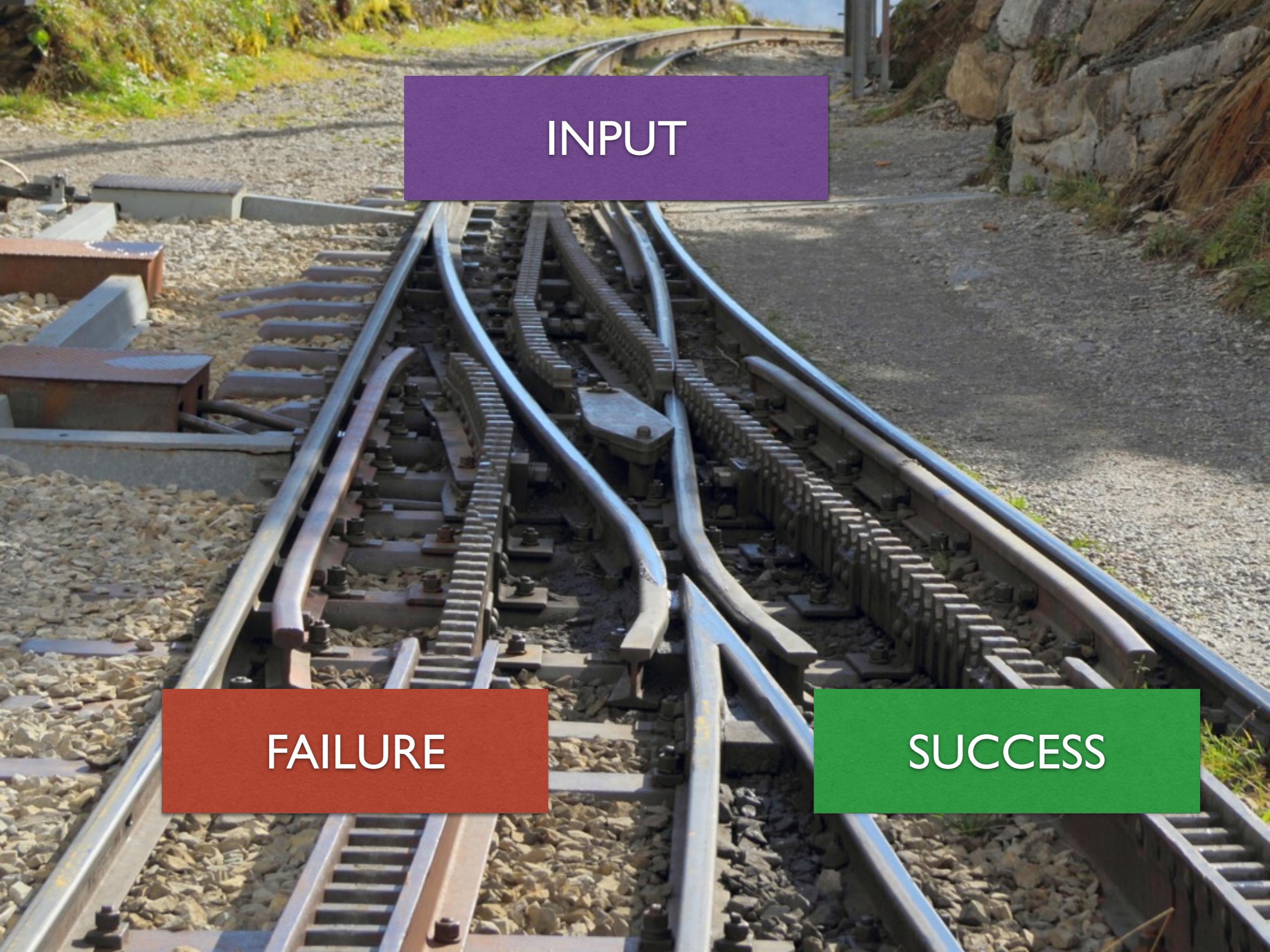
you

me

railways programming

other stuff





INPUT

FAILURE

SUCCESS

REQUEST



401



404



400

RESPONSE

railway programming

Part of the "A recipe for a functional app" series ([more](#))

Railway oriented programming

A recipe for a functional app, part 2

UPDATE: [Slides and video from a more comprehensive presentation available here](#) (and if you understand the Either monad, [read this first!](#))

In the previous post, we saw how a use case could be broken into steps, and all the errors shunted off onto a separate error track, like this:

```
graph LR; Request --> Validate[Validate]; Validate -- Success --> Update[Update]; Update -- Success --> Send[Send]; Send -- Success --> Response; Send -- Failure --> Failure[Failure]
```

In this post, we'll look at various ways of connecting these step functions into a single unit. The detailed internal design of the functions will be described in a later post.

Designing a function that represents a step

Let's have a closer look at these steps. For example, consider the validation function. How would it work? Some data goes in, but what comes out?

Well, there are two possible cases: either the data is valid (the happy path), or something is wrong, in which case we go onto the failure path and bypass the rest of the steps, like this:

```
graph LR; Input --> Validate[Validate]; Validate -- Success --> Output[Output]; Validate -- Failure --> Failure[Failure]
```

But as before, this would not be a valid function. A function can only have one output, so we must use the `Result` type we defined last time:

```
type Result<'TSuccess, 'TFailure> =
| Success of 'TSuccess
| Failure of 'TFailure
```

And the diagram now looks like this:

SHARE THE LOVE
F# |> I ❤️

SLIDES/VIDEOS
Functional Design Patterns
A functional approach to Domain Driven Design
A functional approach to error handling (Railway Oriented Programming)
Property-based testing
Enterprise Tic-Tac-Toe
Dr Frankenkaster and the Monadster
Designing with Capabilities
Understanding parser combinators

SERIES
Why use F#?
Thinking functionally
Expressions and syntax
Understanding F# types
Object-oriented programming in F#
Porting from C#

Bind example

```
bind nameNotBlank
bind name50
bind emailNotBlank
```

use "Bind" to convert to 2-track

Scott Wlaschin - Railway Oriented Programming – error handling in functional languages

From NDC Conferences [\[more\]](#) 2 years ago | more

Follow

More from NDC Cor

Autoplay on

Scott Wlaschin - Railway Oriented Programming – error handling in functional languages

182,948 views

Connecting switches

2 people liked this slide

Domain Driven the F# type S Functional L Scott Wlaschin

Functional Pr Patterns (NDC Scott Wlaschin

Dr Frankenkaster and the Monadster Scott Wlaschin

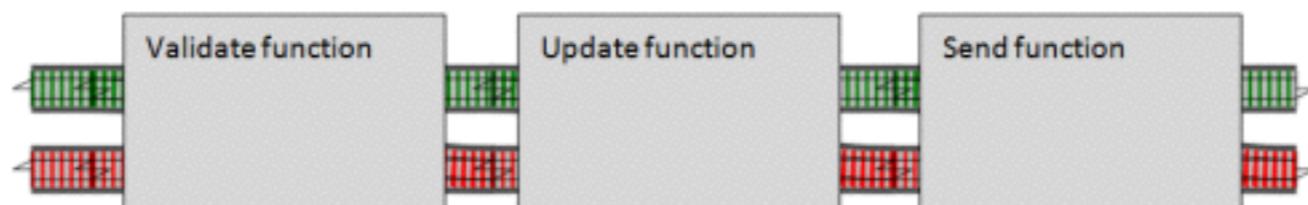
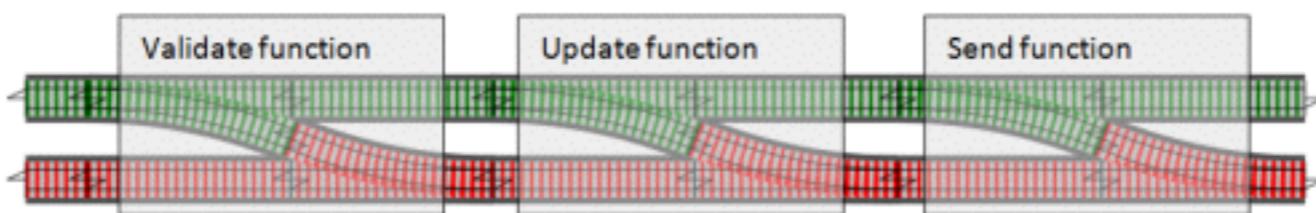
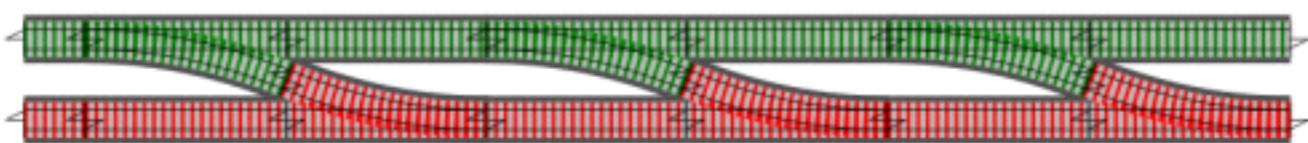
Designing with Capabilities Scott Wlaschin

An Introductio based testing Scott Wlaschin

O-conference Scott Wlaschin

Enterprise Tic Scott Wlaschin

<https://fsharpforfunandprofit.com/posts/recipe-part2/>



railway programming

- Treat your Failures
- Type your failures
- Failures: “run-time system does not know what to do” *
- Errors: “programmer does not know what to do” *
 - Errors handled side-band *

* http://erlang.org/download/armstrong_thesis_2003.pdf

railway programming

- non distributed programming
- for distributed transactions see Sagas

<http://www.cs.cornell.edu/~andru/cs711/2002fa/reading/sagas.pdf>

```
byte[] bytes = customer
    .getAddress()
    .getStreet()
    .substring(0, 3)
    .toLowerCase()
    .getBytes();
```

```
Identity<byte[]> idBytes = new Identity<>(customer)
    .map(Customer::getAddress)
    .map(Address::getStreet)
    .map((String s) -> s.substring(0, 3))
    .map(String::toLowerCase)
    .map(String::getBytes);
```

```
byte[] bytes = customer
    .getAddress()
    .getStreet()
    .substring(0, 3)
    .toLowerCase()
    .getBytes();
```

```
Identity<byte[]> idBytes = new Identity<>(customer)
    .map(Customer::getAddress)
    .map(Address::getStreet)
    .map((String s) -> s.substring(0, 3))
    .map(String::toLowerCase)
    .map(String::getBytes);
```

```
public static void happy(  
    java.util.List<String> stringList) {  
  
    for (String s : stringList) {  
        int i = (50 / Integer.valueOf(s)) - 1;  
        System.out.println(" -> " + i);  
    }  
  
}
```

```
public static void sad(java.util.List<String> stringList) {  
    try {  
        try {  
            for (String s : stringList) {  
                int parsed = Integer.valueOf(s);  
                int div = 50 / parsed;  
                int x = div - 1;  
                System.out.println(" --> " + x);  
            }  
        } catch (ArithmetcException ae) {  
            // What do we do ?!  
        }  
    } catch (NumberFormatException nfe) {  
        // What do we do ?!  
    }  
}
```

```
static final Function<String, Try<Integer>> toInt =  
    x -> Tries.to(() -> Integer.valueOf(x));
```

```
static final Function<Integer, Try<Integer>> divide50 =  
(x) -> Tries.to(() -> 50 / x);
```

```
static final Function<Integer, Integer> minusOne =  
    x -> x - 1;
```

```
public static void grownupAttempt1(List<String> stringList) {  
    final Function<String, Try<Integer>> pipe = x ->  
        toInt.apply(x)  
        .flatMap(divide50)  
        .map(minusOne);  
    final List<Try<Integer>> tryList = stringList.map(pipe);  
}
```

```
static final Function<String, Try<Integer>> toInt =  
    x -> Tries.to(() -> Integer.valueOf(x));
```

```
static final Function<Integer, Try<Integer>> divide50 =  
(x) -> Tries.to(() -> 50 / x);
```

```
static final Function<Integer, Integer> minusOne =  
    x -> x - 1;
```

```
public static void grownupAttempt1(List<String> stringList) {  
    final Function<String, Try<Integer>> pipe = x ->  
        toInt.apply(x)  
            .flatMap(divide50)  
            .map(minusOne);  
    final List<Try<Integer>> tryList = stringList.map(pipe);  
}
```

```
static final Function<String, Try<Integer>> toInt =  
    x -> Tries.to(() -> Integer.valueOf(x));
```

```
static final Function<Integer, Try<Integer>> divide50 =  
(x) -> Tries.to(() -> 50 / x);
```

```
static final Function<Integer, Integer> minusOne =  
    x -> x - 1;
```

```
public static void grownupAttempt1(List<String> stringList) {  
    final Function<String, Try<Integer>> pipe = x ->  
        toInt.apply(x)  
            .flatMap(divide50)  
            .map(minusOne);  
    final List<Try<Integer>> tryList = stringList.map(pipe);  
}
```

```
public static void sad(java.util.List<String> stringList) {  
    try {  
        try {  
            for (String s : stringList) {  
                int parsed = Integer.valueOf(s);  
                int div = 50 / parsed;  
                int x = div - 1;  
                System.out.println(" --> " + x);  
            }  
        } catch (ArithmetcException ae) {  
            // What do we do ?!  
        }  
    } catch (NumberFormatException nfe) {  
        // What do we do ?!  
    }  
}
```

```
static final Function<String, Try<Integer>> toInt =  
    x -> Tries.to(() -> Integer.valueOf(x));
```

```
static final Function<Integer, Try<Integer>> divide50 =  
(x) -> Tries.to(() -> 50 / x);
```

```
static final Function<Integer, Integer> minusOne =  
    x -> x - 1;
```

```
public static void grownupAttempt1(List<String> stringList) {  
    final Function<String, Try<Integer>> pipe = x ->  
        toInt.apply(x)  
            .flatMap(divide50)  
            .map(minusOne);  
    final List<Try<Integer>> tryList = stringList.map(pipe);  
}
```

```
static final Function<String, Try<Integer>> toInt =  
    x -> Tries.to(() -> Integer.valueOf(x));
```

```
static final Function<Integer, Try<Integer>> divide50 =  
(x) -> Tries.to(() -> 50 / x);
```

```
static final Function<Integer, Integer> minusOne =  
    x -> x - 1;
```

```
public static void grownupAttempt1(List<String> stringList) {  
    final Function<String, Try<Integer>> pipe = x ->  
        toInt.apply(x)  
            .flatMap(divide50)  
            .map(minusOne);  
    final List<Try<Integer>> tryList = stringList.map(pipe);  
}
```

```
final String[] arr = new String[]{"1", "2", "0", "jamon"};
```

```
final String[] arr = new String[]{"1", "2", "0", "jamon"};
```

```
Success{s=49}
Success{s=24}
Failure{t=java.lang.RuntimeException: java.lang.ArithmetricException: / by zero}
Failure{t=java.lang.RuntimeException: java.lang.NumberFormatException: For input string: "jamon"}
```

```
def railwayOperation(i: String) = {  
    for {  
        valid <- parse(i)  
        parsed <- authorize(valid)  
        persisted <- persist(parsed)  
        response <- forAUser(persisted)  
    } yield response  
}
```

```
import scala.concurrent.ExecutionContext.Implicits.global

def railwayOperationSync(impl: Impl[Try])(i: String) = {
  for {
    valid <- impl.parse(i)
    parsed <- impl.authorize(valid)
    persisted <- impl.persist(parsed)
    response <- impl.forAUser(persisted)
  } yield response
}

def railwayOperationAsync(impl: Impl[Future])(i: String) = {
  for {
    valid <- impl.parse(i)
    parsed <- impl.authorize(valid)
    persisted <- impl.persist(parsed)
    response <- impl.forAUser(persisted)
  } yield response
}
```

```
import scala.concurrent.ExecutionContext.Implicits.global

def railwayOperationSync(impl: Impl[Try])(i: String) = {
    for {
        valid <- impl.parse(i)
        parsed <- impl.authorize(valid)
        persisted <- impl.persist(parsed)
        response <- impl.forAUser(persisted)
    } yield response
}

def railwayOperationAsync(impl: Impl[Future])(i: String) = {
    for {
        valid <- impl.parse(i)
        parsed <- impl.authorize(valid)
        persisted <- impl.persist(parsed)
        response <- impl.forAUser(persisted)
    } yield response
}

val asyncImpl: Impl[Future] = new PromiseImpl
val syncImpl: Impl[Try] = new TryImpl

val inputs = List("2", "17", "18", "19", "200", "asdf")

inputs.map(railwayOperationSync(syncImpl)).foreach(println)
inputs.map(railwayOperationAsync(asyncImpl)).foreach(println)
```





Sergi GP
@SergiGP



Following

"Estoy probando el JAVI PATH" @JavierCane

View translation

RETWEET

1

LIKES

5



6:02 PM - 27 Apr 2016



1



5

...



Ignasi Marimon-Clos

@ignasi35

@SergiGP @JavierCane export
JAVI_PATH=/usr/bin/unicorns;/usr/bin/rainbows
??

View translation

LIKE

1





Sergi GP

@SergiGP



Following

@ignasi35 @JavierCane el javi path es no testear
los recover de los futures xD

View translation

LIKE

1



8:48 AM - 5 May 2016



1

...

Content

you

me

railways programming

final remarks



Choose Life.

Choose a job.

Choose a career.

Choose a family.

Choose a fucking big
4K monitor, choose SSD,
choose more RAM,
choose an i7.

Choose SOLID.

Choose TDD.

Choose BDD.

Choose CI, CD, auto provision,
self healing, autoscalable.

Choose dragging yourself around
the city to attend a coding dojo.

Choose your colleagues.

Choose your laptop bag
and matching sneakers.

Choose a Star Wars t-shirt.

Choose your future.

Choose to

Choose to
manage

Choose to
manage your

Choose to
manage your errors.

Choose life.

End of presentation

Questions ?

references

<https://billysteele60.files.wordpress.com/2015/09/trainspotting-04.jpg>

<http://static.srcdn.com/wp-content/uploads/Trainspotting-2-Renton.jpg>

https://pmcvariety.files.wordpress.com/2015/12/rexfeatures_1661889a-e1449278924296.jpg

http://cdn.skim.gs/image/upload/v1456344227/msi/trainspotting_tied_to_tracks_lrhjrl.jpg

http://67.media.tumblr.com/18ccff12fa65b095ad1eee950cb2ca65/tumblr_nkf57f9hIN1tr18qlo5_r1_1280.jpg

<https://s-media-cache-ak0.pinimg.com/originals/ee/85/30/ee85301458448e149b422b40629e9729.jpg>

<http://www.beyng.com/images/PlatocaveSmall.jpg>

<https://www.flickr.com/photos/hobart65/8390975307>

<https://twitter.com/SergiGP/status/725354554385174528>

<http://slides.com/julientournay/a-monad-is-just-a-monoid-in-the-category-of-endofunctors-what-s-the-problem/fullscreen#/6>

<https://timedotcom.files.wordpress.com/2014/11/night-shift.jpg?quality=85&w=1100>

<https://mindstepsinc.com/wp-content/uploads/pinky-promise.jpg>