Jug Summer Camp

-enjoy it-

PROGRAMMATION FONCTIONNELLE 😱 EN JAVASCRIPT ❤️

Speaker : Philippe Charrière -

@GitHub

@k33g

@k33g_org

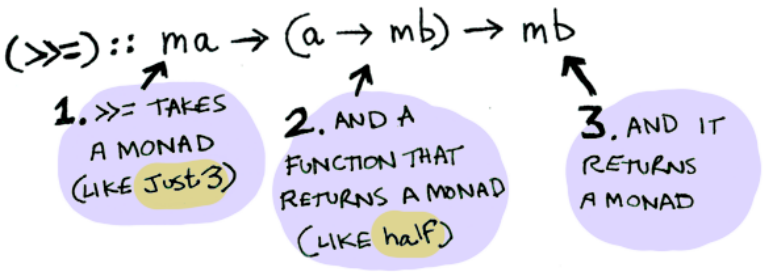
mix-IT

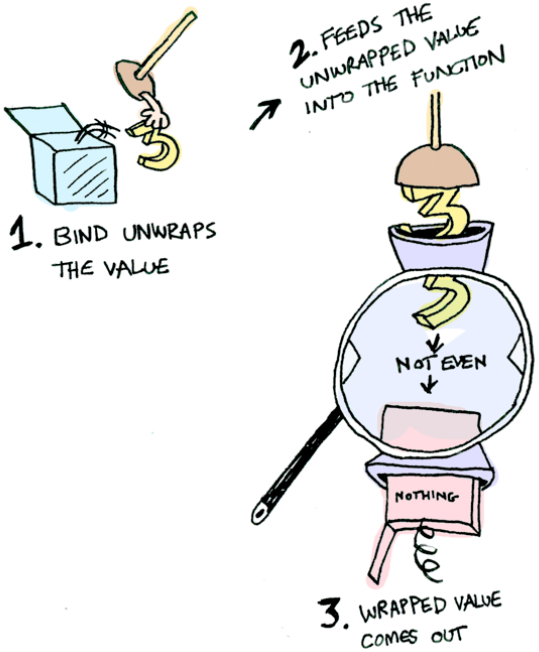
SERLI elastic CloudBees The Enterprise Jenkins Company sonarsource GitHub Developpez.com Club des développeurs et IT pro

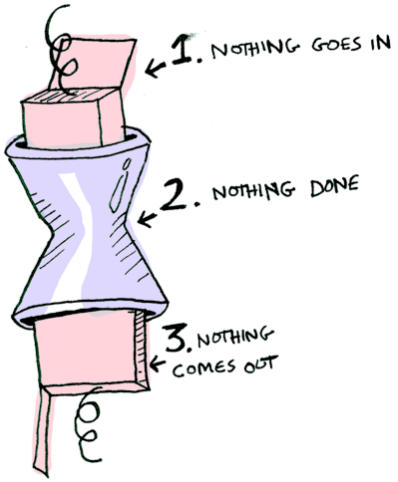
Pourquoi ce talk?

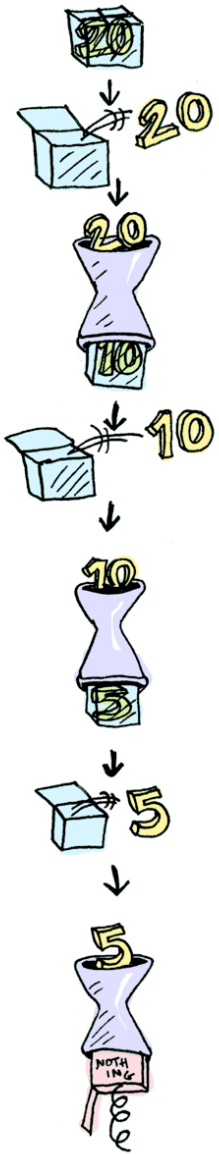

la disjonction logique du functor Either va te permettre de caractériser avec davantage de précision la nature de l'interruption ...
... mais bien sûr
GitHub

http://adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html

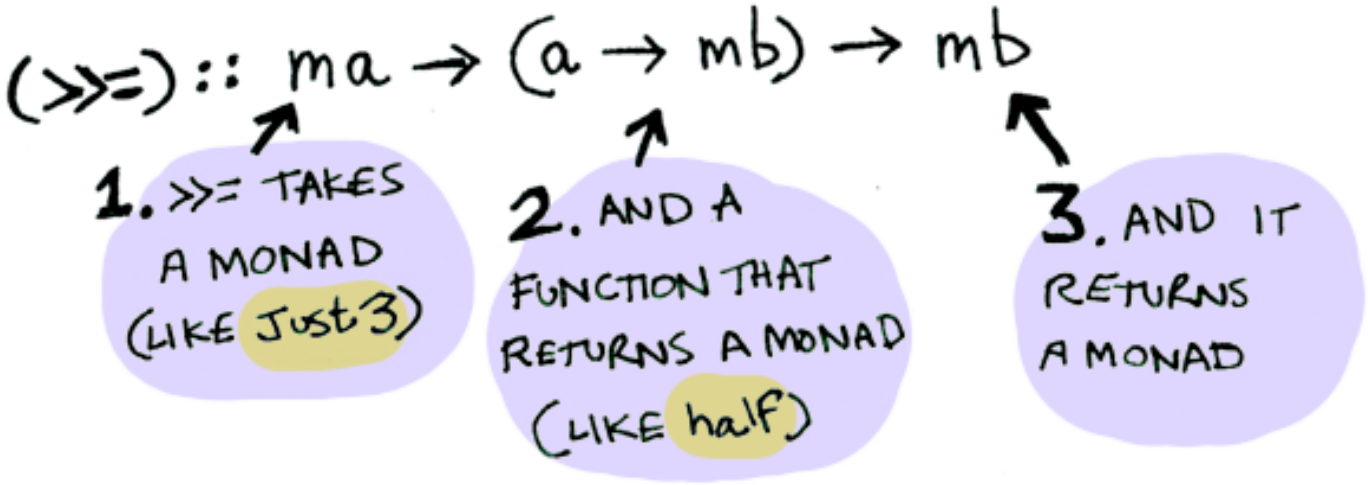








http://adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html



Sondage

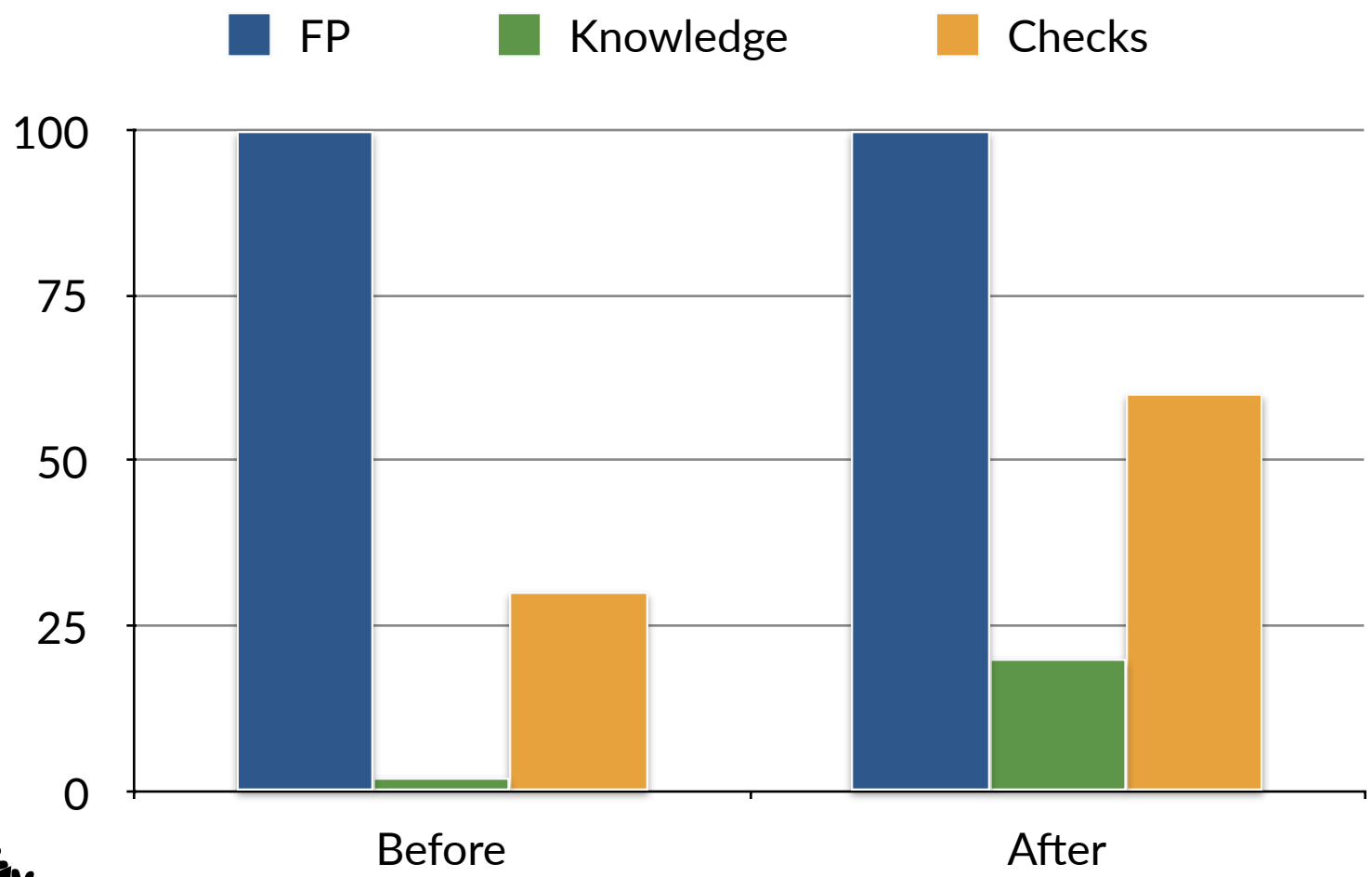
- combien de dev fonctionnels dans la salle?
- qui fait du Scala? 😜
- du Erlang ou Haskell? 😱
- du Java?
- du JavaScript?
- autre?

W.I.P. 🚧

> vocabulaire(s) fonctionnel(s)
> implémentations incomplètes
> échanges de points de vue:
https://github.com/k33g/stools/issues

Vous n'êtes pas obligés d'être d'accord

W.I.P.



Programmation fonctionnelle ?

Définition

> paradigme -> function == 👑
> façon de coder (mindset)
> buzz
> rendre son code safe

développeurs fonctionnels

Pourquoi en JS ❓


PAAAAARSKEUUUUU !!!

Nous allons voir

> Rapidement: quelques “key points”
> Container
> Functor
> Monad
> Maybe
> Either
> Validation ⚠️🆘

Key points

Key points

⚠️ ES2015

> imperative vs functional
> pure function
> function as input & output
> no iteration
> (im)mutability

samples

```
samples/000-func-vs-imp.js
samples/001-pure-func.js
samples/002-func-as-input-output.js
samples/003-no-iterations.js
samples/004-mutability.js
samples/005-immutability.js
```

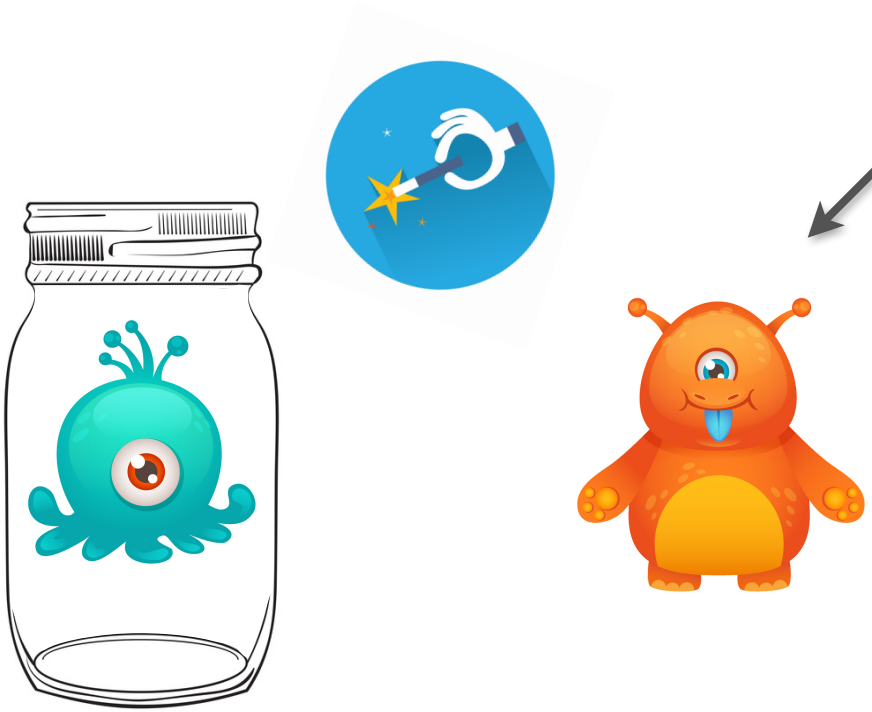
Containers

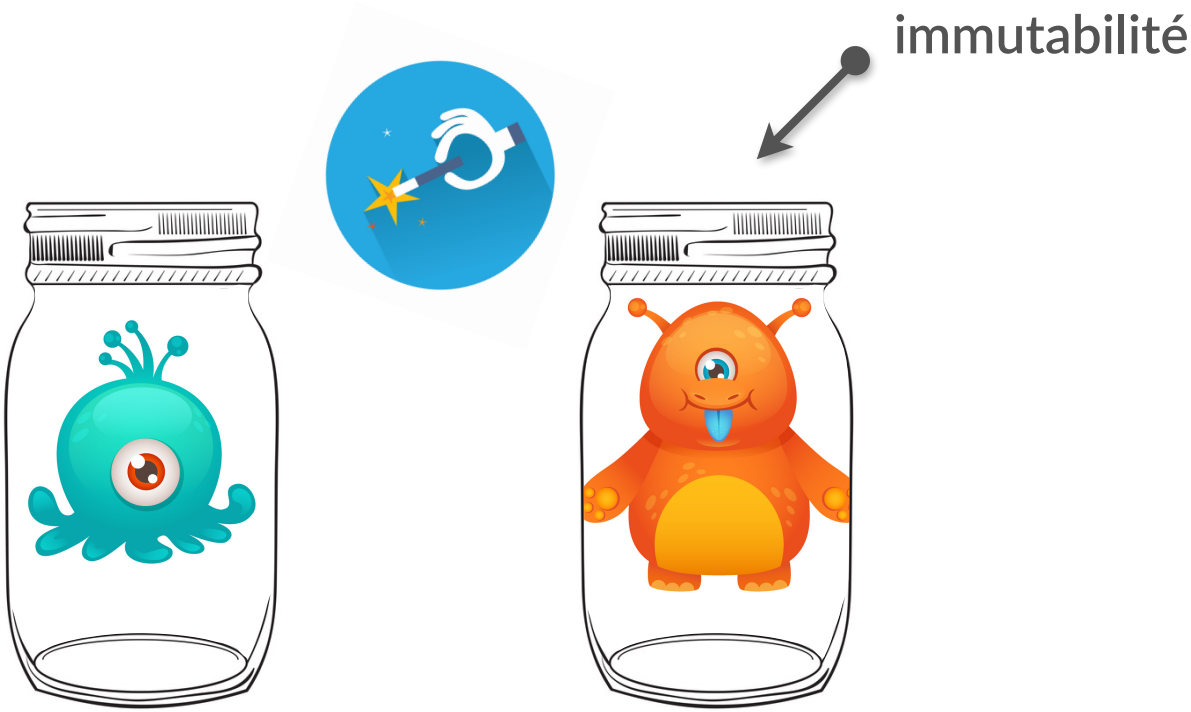
```
class Container {
  constructor(x) {
    const value = x;
    Object.defineProperty(this, "value", { get: () => value })
  }

  static of(x) {
    return new Container(x);
  }

}
```

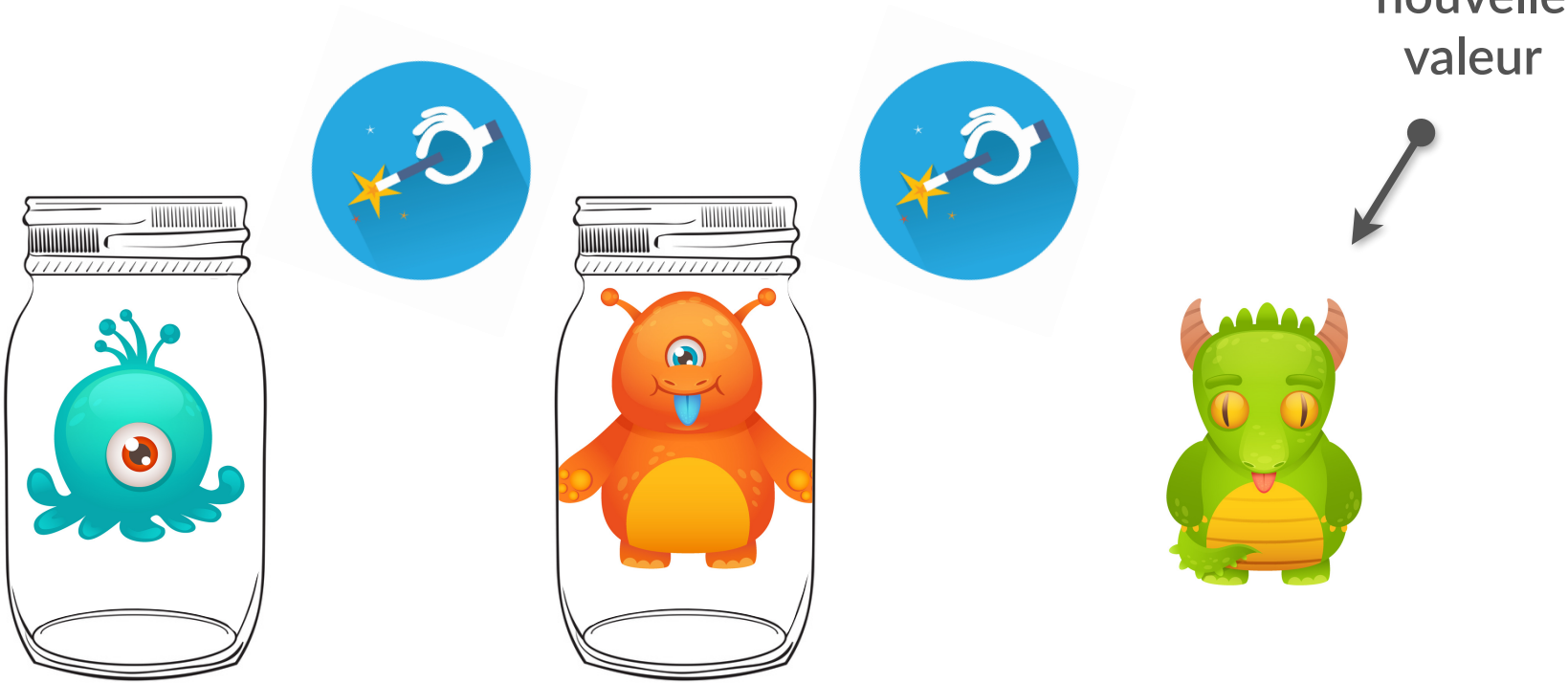
```
let bob = Container.of('Bob Morane')
bob.value == 'Bob Morane'
```

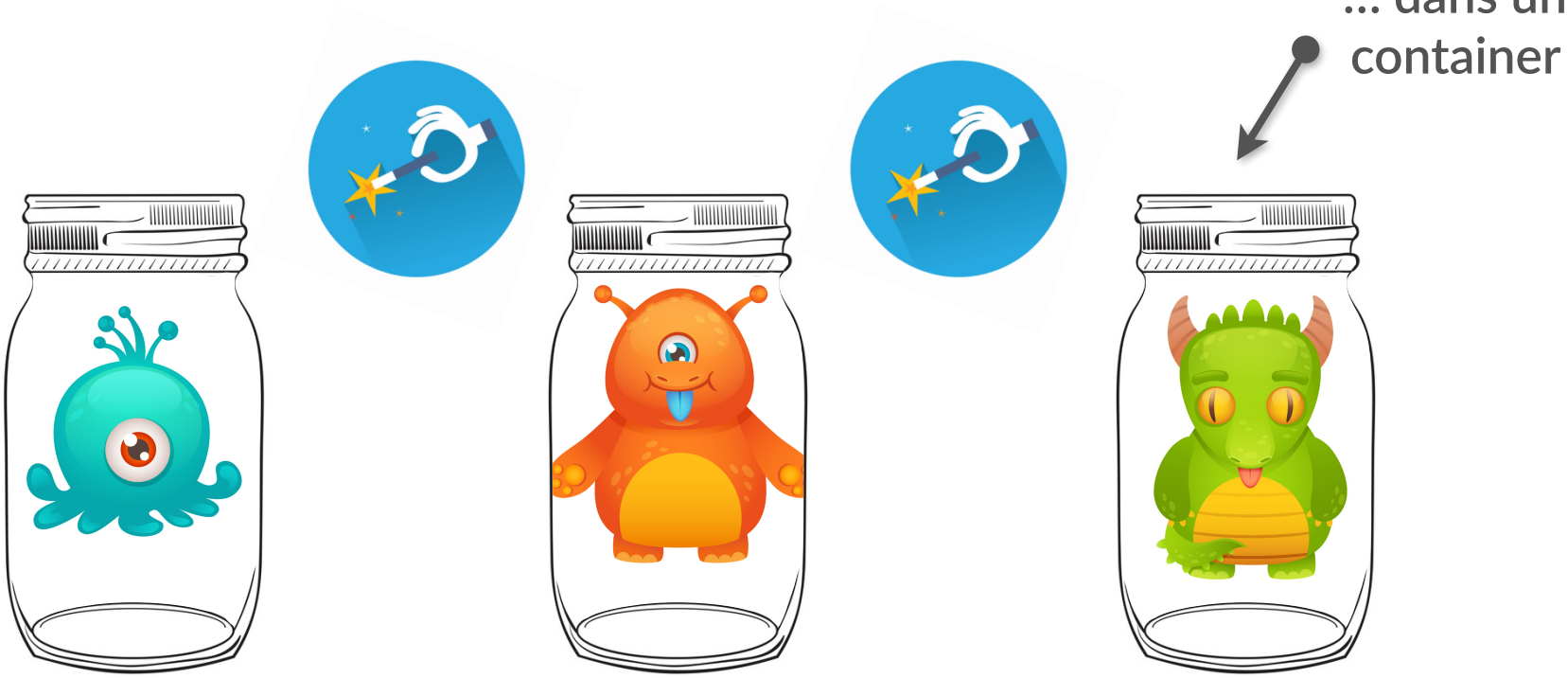
Functor

opération


nouvelle
valeur
reste dans
le bocal


immutabilité


nouvelle
valeur


... dans un
container

```
class Functor extends Container {
  constructor(x) {
    super(x);
  }

  static of(x) {
    return new Functor(x);
  }
  /*
    A map function: used for chaining operations on Container
  */
  map (fn) {
    //return Functor.of(fn(this.value));
    return new this.constructor(fn(this.value));
  }
}
```

nouveau Functor

applique fn(x) à value

I'm a Functor!

```
map(function)
```

samples

samples/020-functor.js

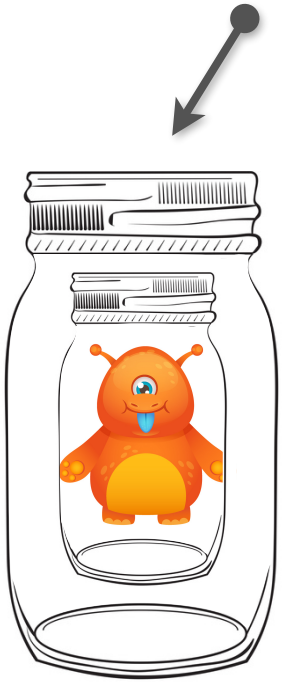
Monad

`map(function)`

```
map(function)
```

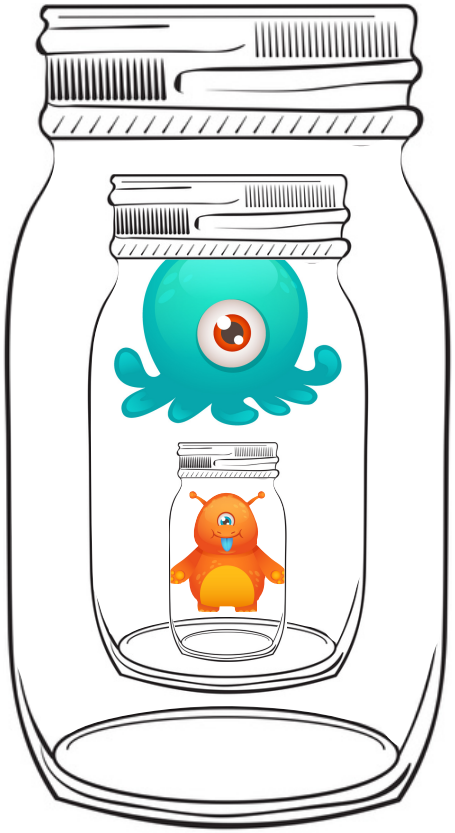
functor

Et là ...



samples

samples/021-functor-of-functor.js



“aplatir”

bind() <-‘attacher’

```
class Monad extends Functor {
  constructor(x) {
    super(x);
  }

  static of(x) {
    return new Monad(x);
  }
  /*
  map (fn) {
    return Monad.of(fn(this.value));
  }
  */

  /* So, I'm a monad because I have a bind method */
  bind (fn) {
    return fn(this.value);
  }
}
```

samples

samples/030-monad.js

@k33g @k33g_org

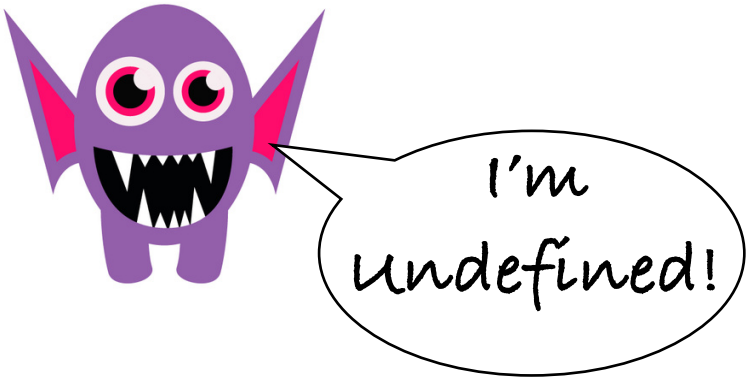
I'm a Monad!

```
bind(function)
```

```
map(function)
```

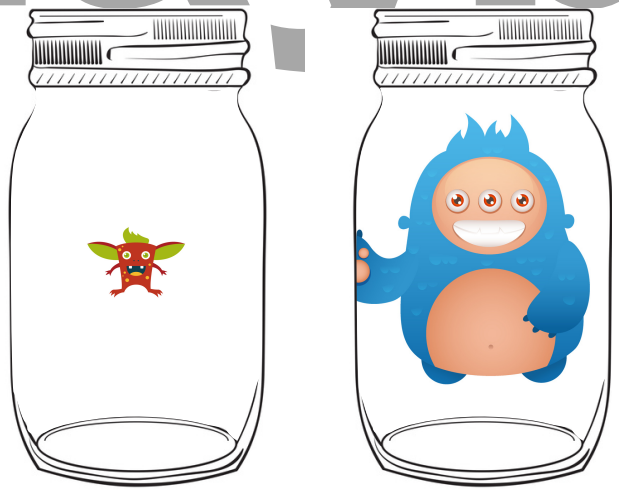
Maybe

Pain points



Maybe

None Some



```
class Maybe {

  static Some(x) { return new Some(x); }

  static None() { return new None(); }

  static fromNullable(x) {
    return (x !== null && x !== undefined)
      ? Maybe.Some(x)
      : Maybe.None();
  }

  static of(x) { return Maybe.Some(x); }
}
```

```
class None extends Monad {
  constructor() { super(null) }
  // overrides Functor.map
  map() { return this; }
  // overrides Monad.map
  bind (fn) { return this.value; }

  getOrElse(value) {
    return value;
  }

  isNone() {
    return true;
  }
  isSome() {
    return false;
  }

}
```

```
class Some extends Monad {
  constructor(x) { super(x); }
  // overrides Functor.map
  map(fn) { // ⚠️⚠️⚠️
    let res = fn(this.value);
    if(res == null || res == undefined) {
      return new None()
    }
    return new Some(res);
  }

  getOrElse() { return this.value; }

  isNone() { return false; }

  isSome() { return true;}
}
```

```
Maybe.fromNullable(42)
// -> == Some(42)

Maybe.fromNullable(null)
// -> == None()

Maybe.fromNullable(42).getOrElse(“Quarante-deux")
// -> == 42

Maybe.fromNullable(null).getOrElse(“Quarante-deux")
// -> == “Quarante-deux”
```

samples

samples/041-maybe.js
samples/042-maybe.js

I'm a Maybe!

J'ai 2 sous types

None

Some

```
bind(function)
```

```
map(function)
```

Either

```
let githubCli = new GitHubClient({
  baseUri: "http://github.at.home/api/v3",
  token:process.env.TOKEN_GHE_27_K33G
})

githubCli.fetchUserRepositories({handle:”k33g"})
.then(repositories => {
  console.log("🤖:", repositories);
}).catch(error => {
  console.log("😡", error.message, error);
})

githubCli.fetchOrganizationRepositories({organization:"AC ME”})
.then(repositories => {
  console.log("🤖:", repositories);
}).catch(error => {
  console.log("😡", error.message, error);
})
```

😡 HttpException { Error: HttpException
    at HttpException (/Users/k33g/dev.js/stools/samples/GitHubClient.js:8:5)
    at fetch.then.response (/Users/k33g/dev.js/stools/samples/GitHubClient.js:41:15)
    at process._tickCallback (internal/process/next_tick.js:103:7)
  status: 404,
  statusText: 'Not Found',
  url: 'http://github.at.home/api/v3/orgs/AC ME/repos' }
🤖: [{ id: 40,
    name: 'cuckoo',
    full_name: 'k33g/cuckoo',

Either

Left **Right**

```
class Either {

  static Left(x) { return new Left(x); }

  static Right(x) { return new Right(x); }


  static fromNullable(x) {
    return (x !== null && x !== undefined)
      ? Either.Right(x)
      : Either.Left();
  }


  static of(x) { return Either.Right(x); }
}
```

ou Monad

```
class Left extends Functor {
  constructor(err) {
    super(err)
  }

  map() { return this; }

  getOrElse(value) {return value; }


  isRight() { return false; }
  isLeft() { return true; }

  cata(leftFn, rightFn) { return leftFn(this.value); }
}
```

```
class Right extends Functor {
  constructor(x) {
    super(x);
  }

  map(fn) { return new Right(fn(this.value)); }

  getOrElse() { return this.value; }

  isRight() { return true; }
  isLeft() { return false; }

  cata(leftFn, rightFn) { return rightFn(this.value); }
}
```

samples

samples/051-either.js
samples/052-either.js

I'm an Either!

J'ai 2 sous types

Left

Right

`bind(function)` ou pas

`map(function)`

```
cata(
  left function,
  right function
)
```

Mais on ne trace pas toutes les erreurs

Validation

Validation

Fail **Success**

subtilité

```
class Validation {

  static Success(x) { return new Success(x); }

  static Fail(errList) { return new Fail(errList); }

  static of(x) { return Validation.Success(x); }
}
```

```
class Success extends Monad {
  constructor(x) {
    super(x);
  }

  isSuccess() { return true; }

  isFail() { return false; }

  ap(otherContainer) {
    return otherContainer instanceof Fail
      ? otherContainer
      : otherContainer.map(this.value)
  }

  cata(failureFn, successFn) { return successFn(this.value); }
}
```

```
class Fail extends Monad {
  constructor(err) {
    super(err)
  }
  // override
  map() { return this; }

  isSuccess() { return false; }
  isFail() { return true; }

  ap(otherContainer) {
    return otherContainer instanceof Fail
      ? new Fail(this.value.concat(otherContainer.value))
      : this
  }

  cata(failureFn, successFn) { return failureFn(this.value); }
```

subtilité

```
Validation.Success(40).cata(
  error => {console.log("😡", error)},
  success => {
    console.log("😛", success)
  }
)

Validation.Fail("Oups I did it again!").cata(
  error => {console.log("😡", error)},
  success => {
    console.log("😛", success)
  }
)
```

samples

samples/061-validation.js
samples/062-validation.js

I'm a validation!

J'ai 2 sous types

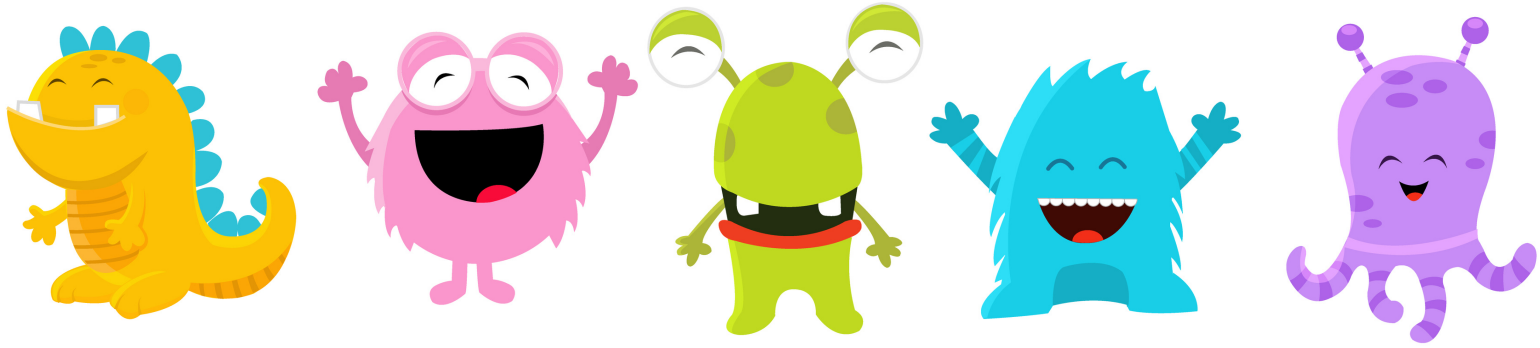
Fail

Success

```
ap(validation)
```

```
map(function)
```

```
cata(
  failureFn function,
  successFn function
)
```



Paysage Fonctionnel côté JS

Quelques libs

> Monet.js: https://cwmyers.github.io/monet.js/
> Ramda.js: http://ramdajs.com/0.21.0/index.html
> Underscore: http://underscorejs.org/
> Lodash: https://lodash.com/
> Folktale: http://folktalejs.org/
> Fantasy Land: https://github.com/fantasyland

Ressources

Lectures

Mostly Adequate Guide (Brian Lonsdorf): https://www.gitbook.com/book/drboolean/mostly-adequate-guide/details

Functional Programming in Java (Pierre-Yves Saumont): https://www.manning.com/books/functional-programming-in-java

Functional Programming in JavaScript (Luis Atencio): https://www.manning.com/books/functional-programming-in-javascript

Functional Programming in Scala (Paul Chiusano and Rúnar Bjarnason): https://www.manning.com/books/functional-programming-in-scala

Le code source de Golo:

https://github.com/eclipse/golo-lang/blob/master/src/main/golo/errors.golo

https://github.com/eclipse/golo-lang/blob/master/src/main/java/gololang/error/Result.java

Les monades en PHP: http://mcamuzat.github.io/blog/2015/11/11/les-monades-en-php-cest-possible-dot/

THE MARVELLOUSLY MYSTERIOUS JAVASCRIPT MAYBE MONAD: http://jrsinclair.com/articles/2016/marvellously-mysterious-javascript-maybe-monad/

Talks

Gérer les erreurs avec l'aide du système de types de Scala !
(David Sferruzza):
https://www.youtube.com/watch?v=TwJQKrZ23Vs

TDD, comme dans Type-Directed Development
(Clément Delafargue):
https://www.youtube.com/watch?v=XhcgCF0xXRs

Remerciements

Merci

> à vous
> Loïc Descotte @loic_d
> Julien Ponge @jponge
> Yannick Loiseau @yannick_loiseau
> Thierry Chantier @titimoby
> **Etienne Issartial** | CommitStrip
> Clément Delafargue @clementd
> @JugSummerCamp

https://github.com/k33g/stools

Questions

https://github.com/k33g/stools/issues