How FP Made Grammarly's Codebase Error-Free and Predictable

Anton Pets
@oopets

7 March 2020

Nothing about monads today!

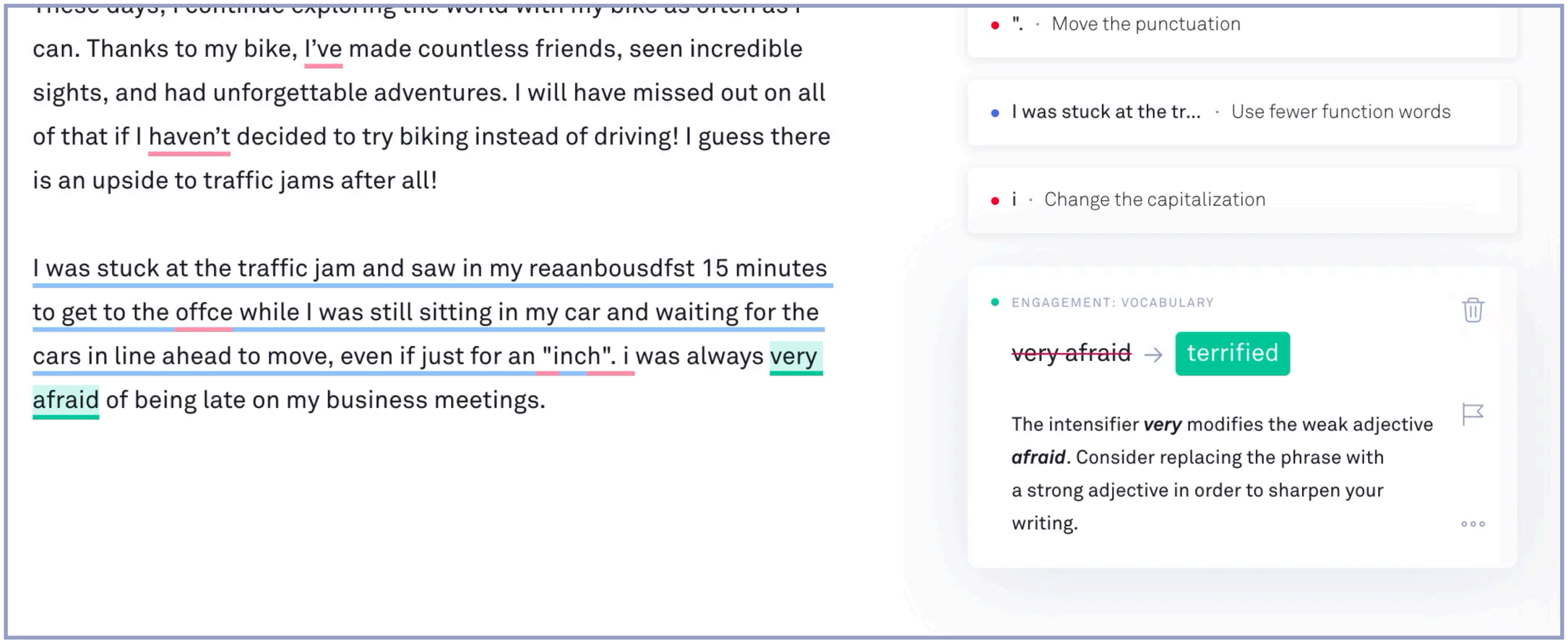
Grammarly - a writing assistant that helps make your communication clear and effective, wherever you type.

Demo document

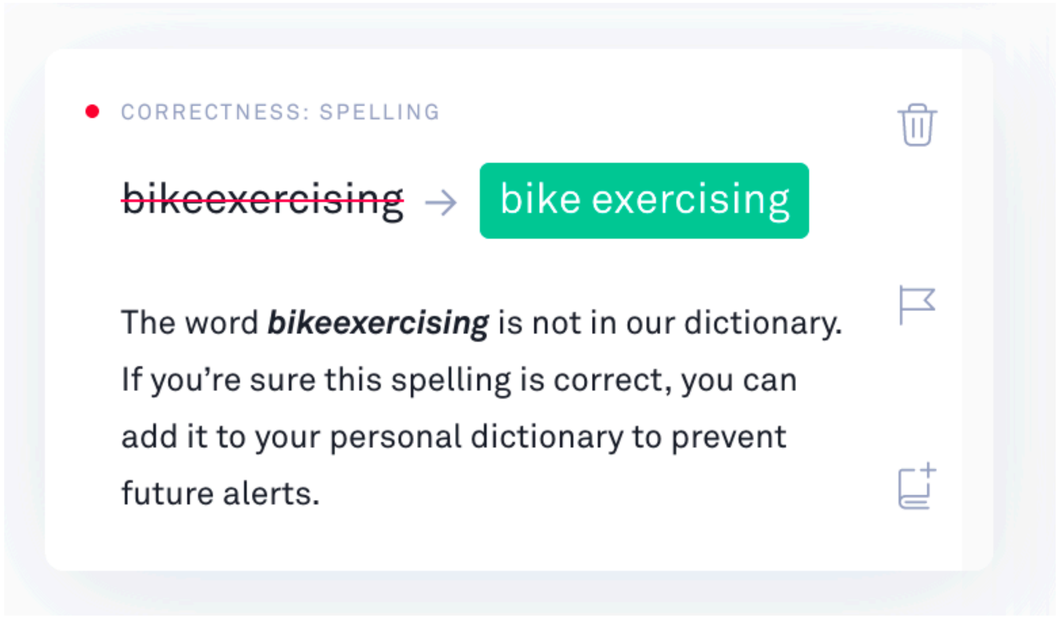
Remember when you were a careless eight year old kid riding a bike with your friends, racing each other around the neighborhood? Remember that feeling of absolute freedom as you felt the wind in your hair and the smile it put on your face? I never thought I would feel that way as a grown up, until my friends presented me a red brand-new bike. At first, I was a bit skeptical about the total idea of commuting by bike. One morning a couple of days later, I changed completely my mind. I was stuck at a traffic jam and saw in my rear mirror a man in a suit riding a classy bike with his laptop case in one hand and a handlebar in the other. I figured out it would take him about 15 minutes to get to the office while I was still sitting in my car and waiting for the cars in line ahead to move , even if just for a inch. I was always very afraid of being late for my business meetings.

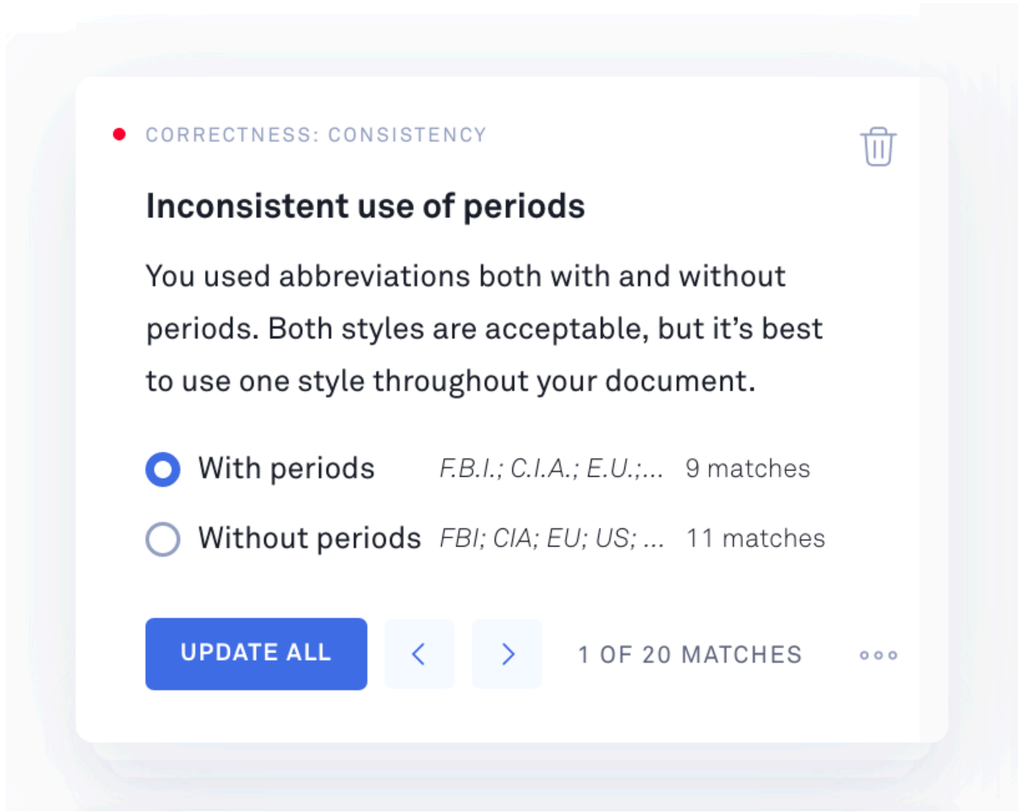
Grammarly Challenges

Text is difficult



Change is difficult



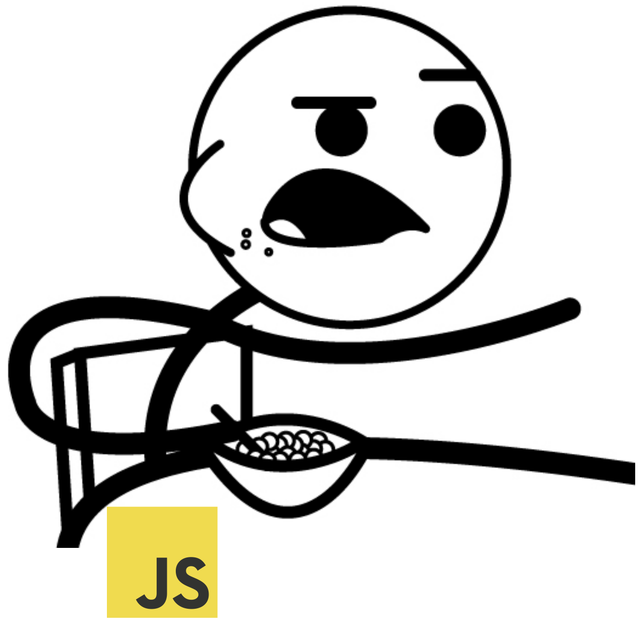


Our combo

Object
Oriented
Design

Our combo

Object
Oriented
Design


JS


TS

TypeScript is…

Args validation - free & out of the box

```
function login(user) {
    if(user &&
       user.unauthorized &&
       user.hadBreakfast &&
       user.feelsGood ) { ... }
}
```

TypeScript is…

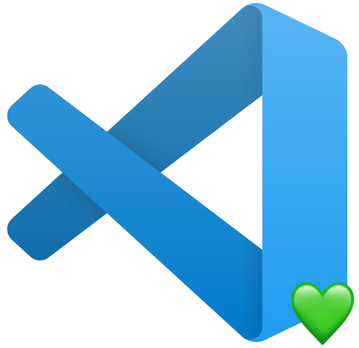
Args validation - free & out of the box

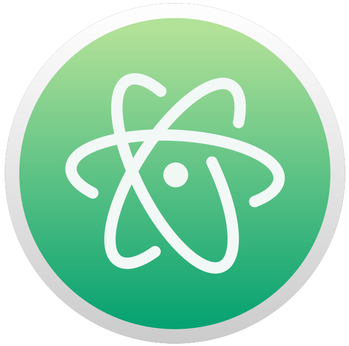
```
1 function login(user: User.Unauthorized): User.Authorized {
2  …
3 }
4
5 // later  …
6 const authorized = login(unathorized_user)
7
8 // Secure operation
9 retrieve_protected_info(authorized)
```

TypeScript is…

```
/**
* Easy to read
* Self-documenting: less JSDoc
**/
function login(user: User.Unauthorized): User.Authorized {

}
```

TypeScript is…





TypeScript is…

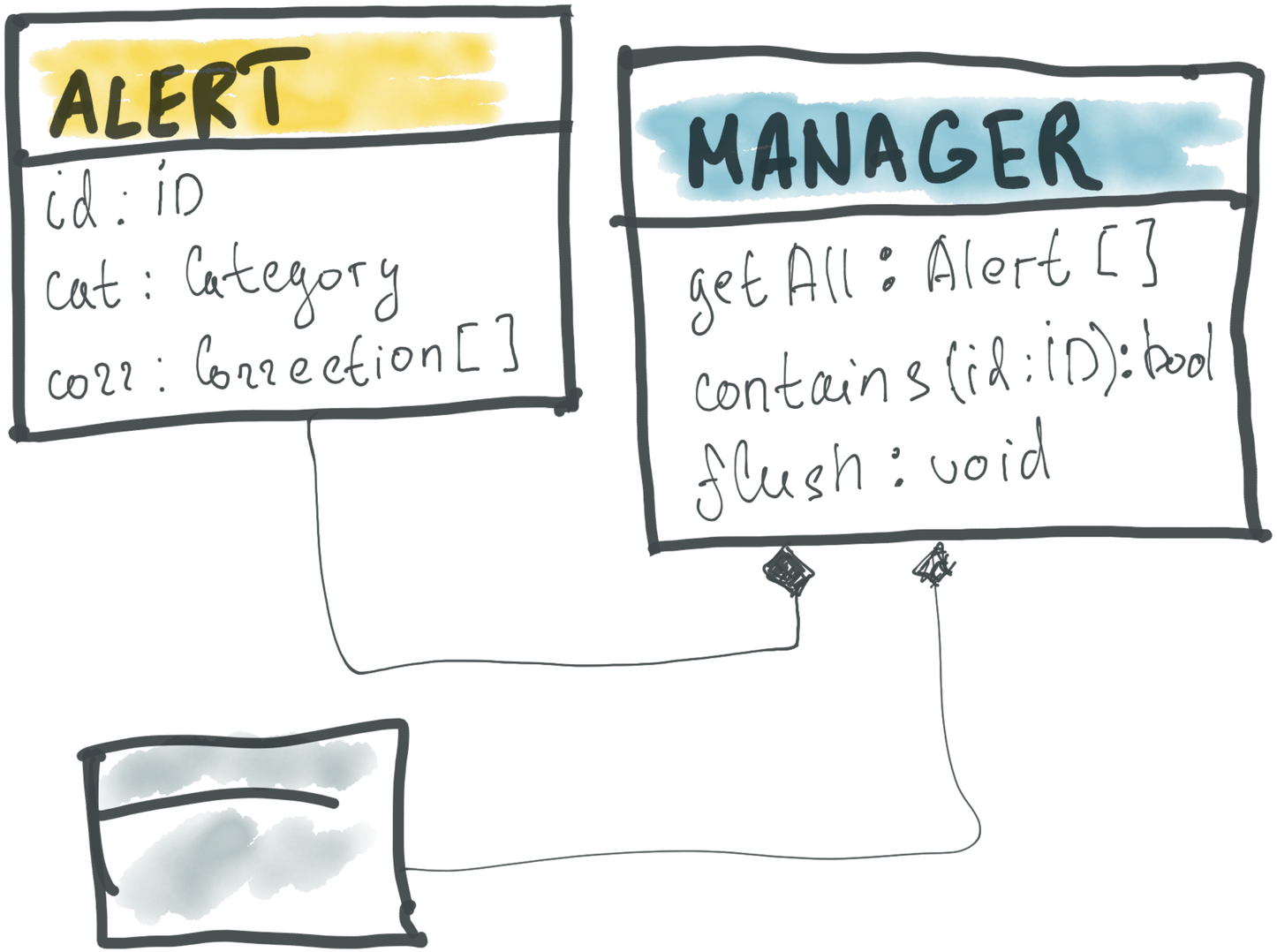
Confident refactoring

```
  | ERROR in packages/app//Users/oopets/Grammarly/denali/pacl
,84):
  | TS2339: Property 'updateAllBtn' does not exist on type 'f
pp/src/denali-editor-page/view/sidebar/card/super/full.sss"
```

TypeScript is… Interfaces!

```
1 interface Alert {
2   id: number
3   category: Category
4   corrections: Array<Correction>
5   …
6 }
7
8 interface AlertManager {
9     getAll(): Array<Alert>
10    …
11 }
```

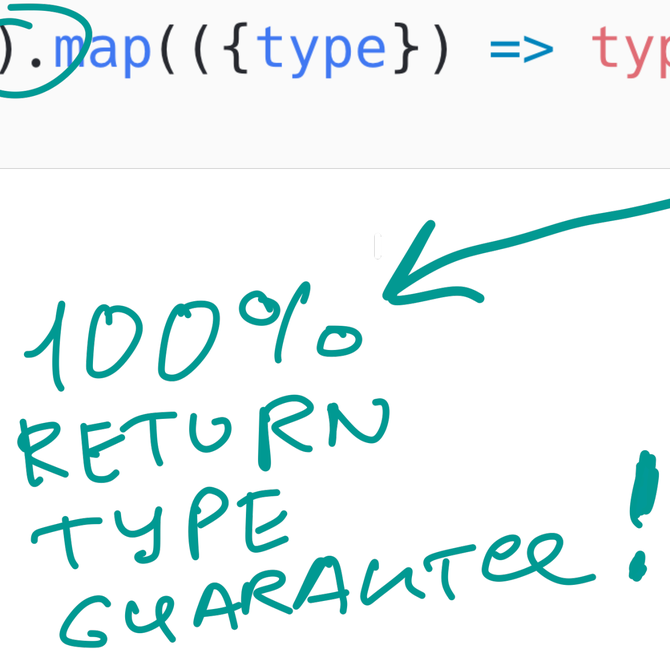
TypeScript is… Interfaces!



Types +  = ❤️

```
alertManager.getAll().filter(Alert.isFree).map(({type}) => type)
```

```
alertManager.getAll().filter(Alert.isFree).map(({type}) => type)
```



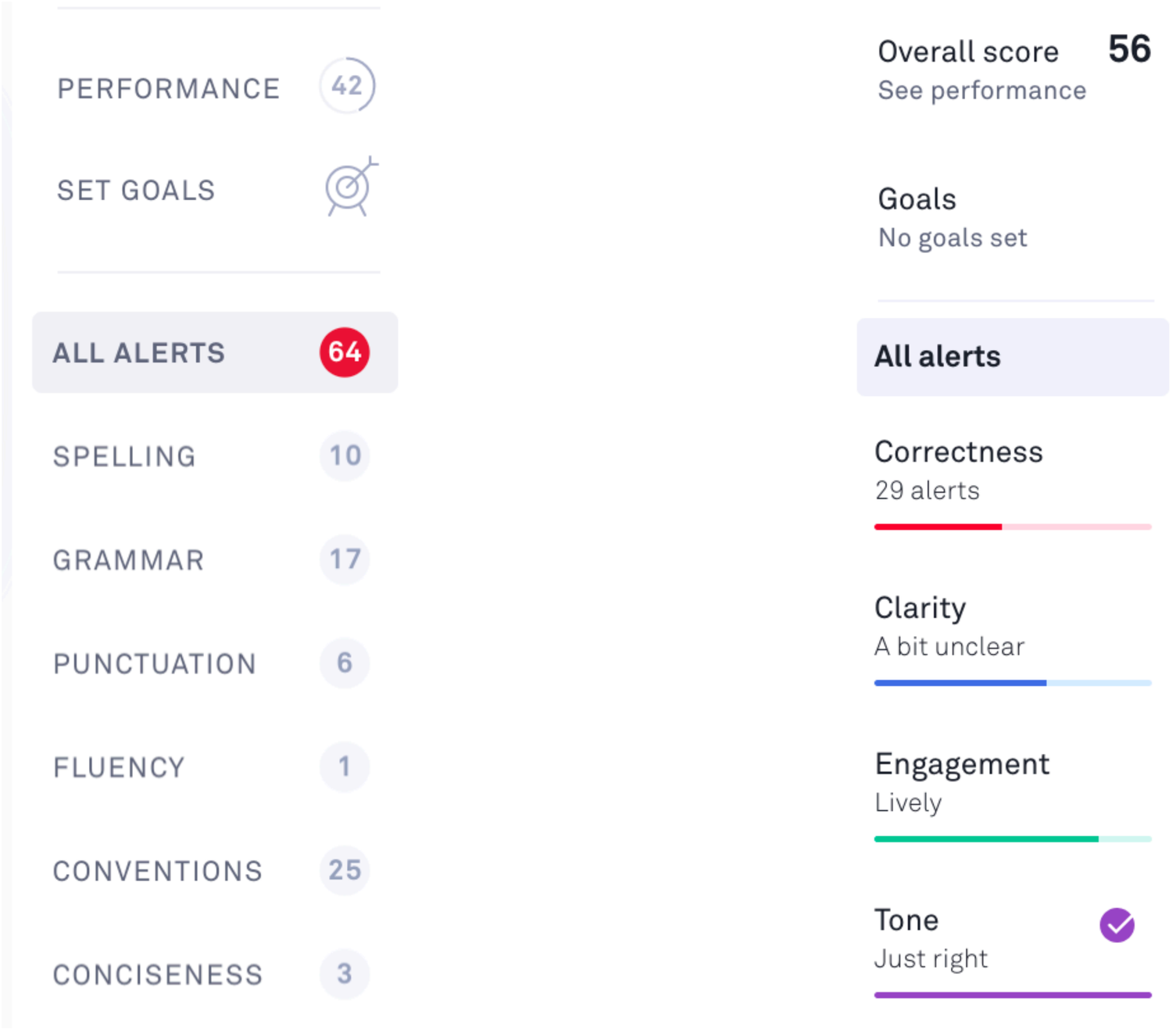
TypeScript FP libs

Funfix

fp-ts

Functional Programming: Real-Life Cases

Case 1/4


PERFORMANCE
42
SET GOALS
ALL ALERTS
64
SPELLING
10
GRAMMAR
17
PUNCTUATION
6
FLUENCY
1
CONVENTIONS
25
CONCISENESS
3
Overall score
56
See performance
Goals
No goals set
All alerts
Correctness
29 alerts
Clarity
A bit unclear
Engagement
Lively
Tone
Just right

A

PERFORMANCE 42

SET GOALS

ALL ALERTS 64

SPELLING 10

GRAMMAR 17

PUNCTUATION 6

FLUENCY 1

CONVENTIONS 25

CONCISENESS 3

50%

B

Overall score 56
See performance

Goals
No goals set

All alerts

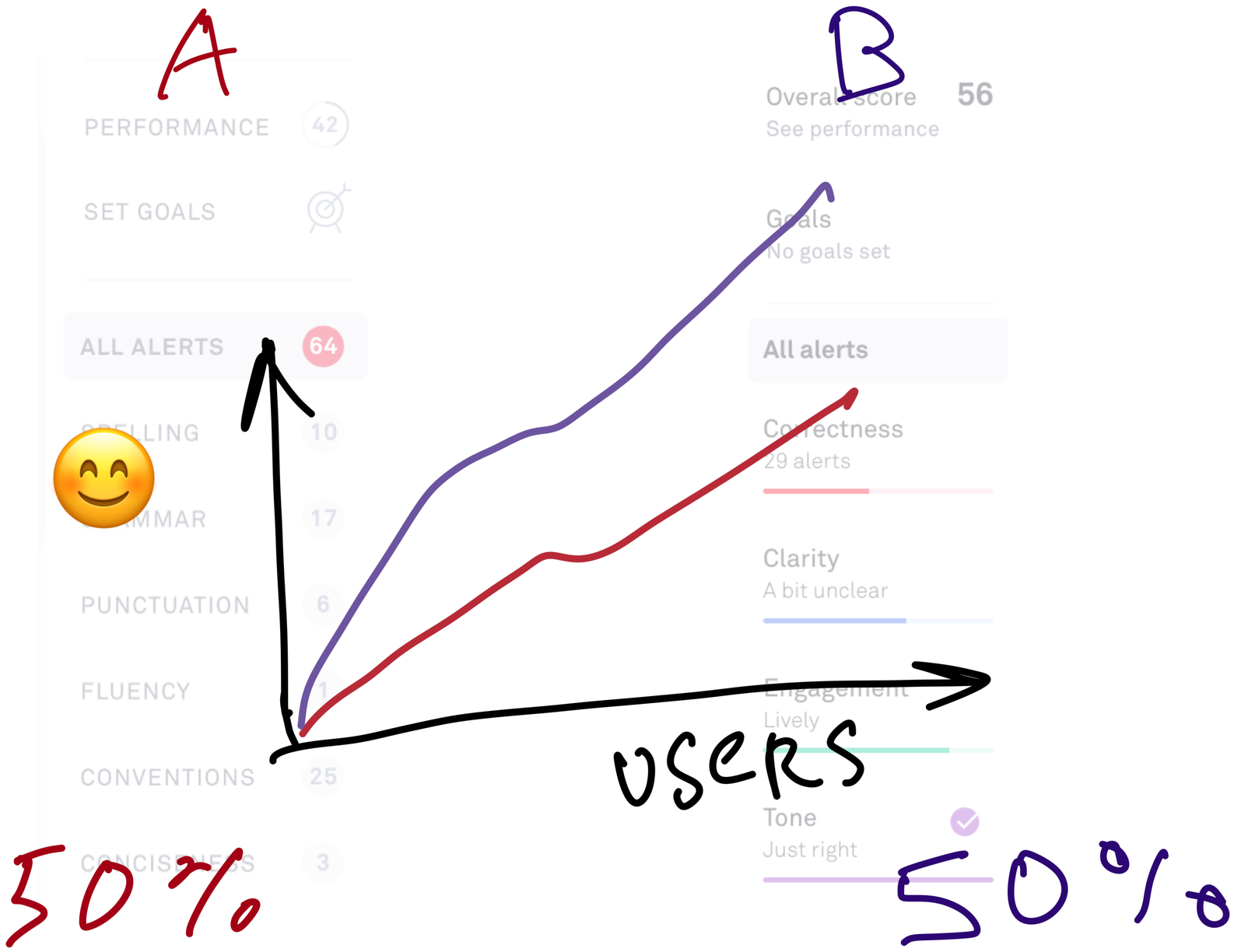
Correctness
29 alerts

Clarity
A bit unclear

Engagement
Lively

Tone
Just right

50%

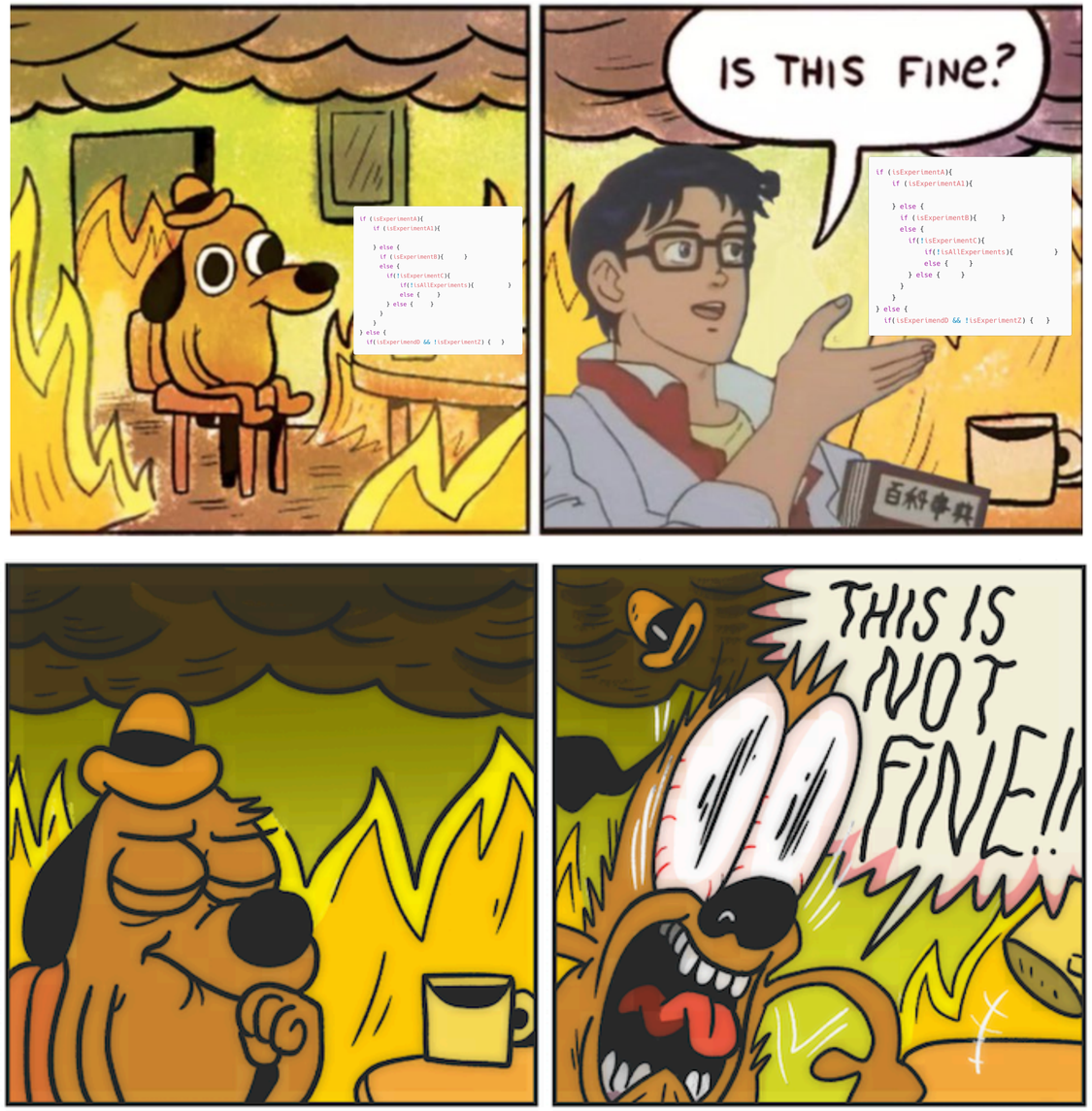

A
B
PERFORMANCE
42
SET GOALS
ALL ALERTS
64
SPELLING
10
GRAMMAR
17
PUNCTUATION
6
FLUENCY
1
CONVENTIONS
25
CONCISENESS
3
Overall score
56
See performance
Goals
No goals set
All alerts
Correctness
29 alerts
Clarity
A bit unclear
Engagement
Lively
Tone
Just right
USERS
50%
50%

How to experiment?


PERFORMANCE 42
SET GOALS
ALL ALERTS 64
SPELLING 10
GRAMMAR 17
PUNCTUATION 6
FLUENCY 1
CONVENTIONS 25
CONCISENESS 3
// Naive approach
interface Alert {
id: Id,
category: {
expA: Category
expB: NewCategory
}
}
Overall score 56
See performance
Goals
No goals set
All alerts
Correctness
29 alerts
Clarity
A bit unclear
Engagement
Lively
Tone
Just right

```
if (isExperimentB)
    categoryName = alert.category.expB
else
    categoryName = alert.category.expA
```

```
if (isExperimentA){
    if (isExperimentA1){

    } else {
      if (isExperimentB){      }
      else {
        if(!isExperimentC){
            if(!isAllExperiments){          }
            else {     }
        } else {     }
      }
    }
} else {
  if(isExperimendD && !isExperimentZ) {   }
```


IS THIS FINE?
THIS IS NOT FINE!!

Can we  better ?


λ

```
const alertManager = new AlertManager(
  …,
  transformAlertFn
)
```

```
1 const alertManager = new AlertManager(
2   ...,
3   transformAlertFn
4 )
```

```
new AlertManager(..., transformAlertFn)
```

```
const transformFn =
    isExperimentB ? changeCategoryNames : x => x

const newAlert = transformFn(alert)
```

```
compose(f, g)(x) ≡ f(g(x))
```

```
const transformFn: Alert => Alert
 = compose(
    categoryNamesExperiment,
    alertImportanceExperiment,
    …
 )
```


const transformFn: Alert => Alert
= compose(
isExperimentB ? changeCategoryNames : x => x,
isExperimentC ? changeAlertImportance : x => x
...
)
immutable
Pure functions

Immutability

```
1 // Bad 👎
2 function changeCategoryNames(alert: Alert): Alert {
3   …
4   alert.category = newName
5   return alert
6 }
7
8 // Good 👍
9 function changeCategoryNames(alert: Alert): Alert {
10     …
11     return {...alert, category: newName}
12 }
```

Pure Functions

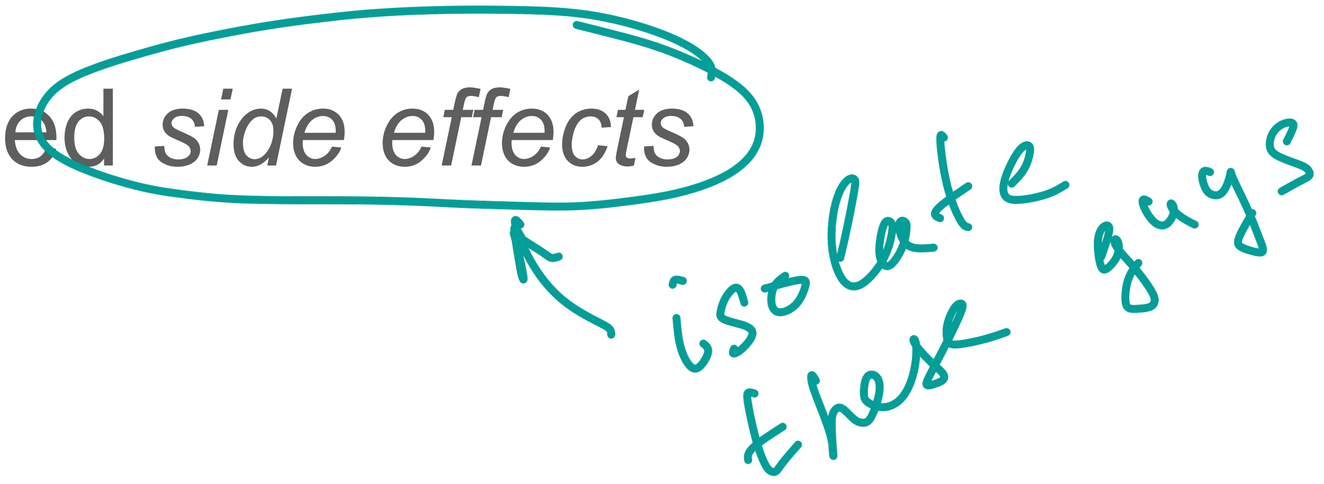
Deterministic: Same Input ➡️ Same output

No uncontrolled *side effects*

Pure Functions

Deterministic: Same Input ➡️ Same output

No uncontrolled *side effects*



Pure Functions

```
1 // Impure
2 function changeCategoryNames(alert: Alert): Alert {
3   const newName = window.localStorage.getItem("name")
4   console.log(newName)
5   return {...alert, category: newName}
6 }
7
8 // Pure 👍
9 function changeCategoryNames(alert: Alert): Alert {
10     return {...alert, category: 'Some name'}
11 }
```

Before

PERFORMANCE 42

SET GOALS

ALL ALERTS 64

SPELLING 10

GRAMMAR 17

PUNCTUATION 6

FLUENCY 1

CONVENTIONS 25

CONCISENESS 3

```
const transformFn: Alert => Alert
 = compose(
     categoryNamesExperiment,
     alertImportanceExperiment,
     ...
 )
```

After

Overall score 56
See performance

Goals
No goals set

All alerts

Correctness
29 alerts

Clarity
A bit unclear

Engagement
Lively

Tone
Just right

What we achieved

⛳ Reliable

Almost nobody knows about 🧪

⛳ Extensible

Add ∞ number of 🧪

Case 2/4

- **travelling** · Change the spelling
- **really** · Remove the phrase
- **are seeing** · Change the verb tense
- **and** · Add a comma
- **good** · Choose a different word

- I was stuck at the traffic jam ... · Use fewer function words

~~really~~

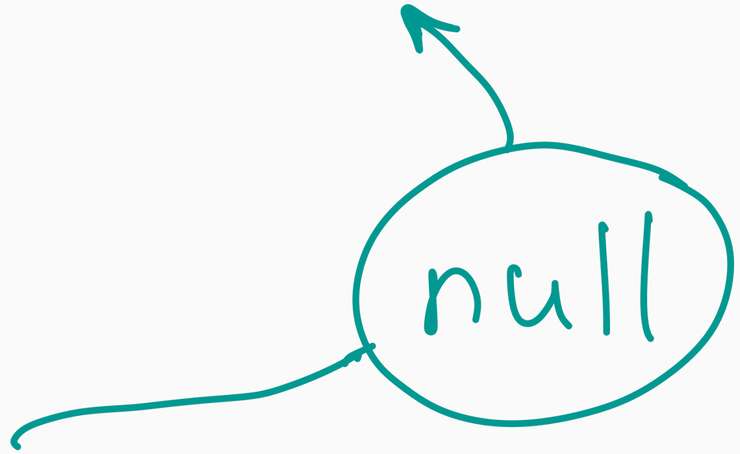
It appears that ***really*** may be unnecessary in this sentence. Consider removing it.

- all the way · Remove the phrase
- future · Remove tautology

getActiveAlert()

```
getActiveAlert(): Alert| null
```

```
const transformFn: Alert => Alert
 = compose(
    newCategoryUI ? changeCategoryNames : x => x
    ...
 )

 // Later
 transformFn(getActiveAlert())
```

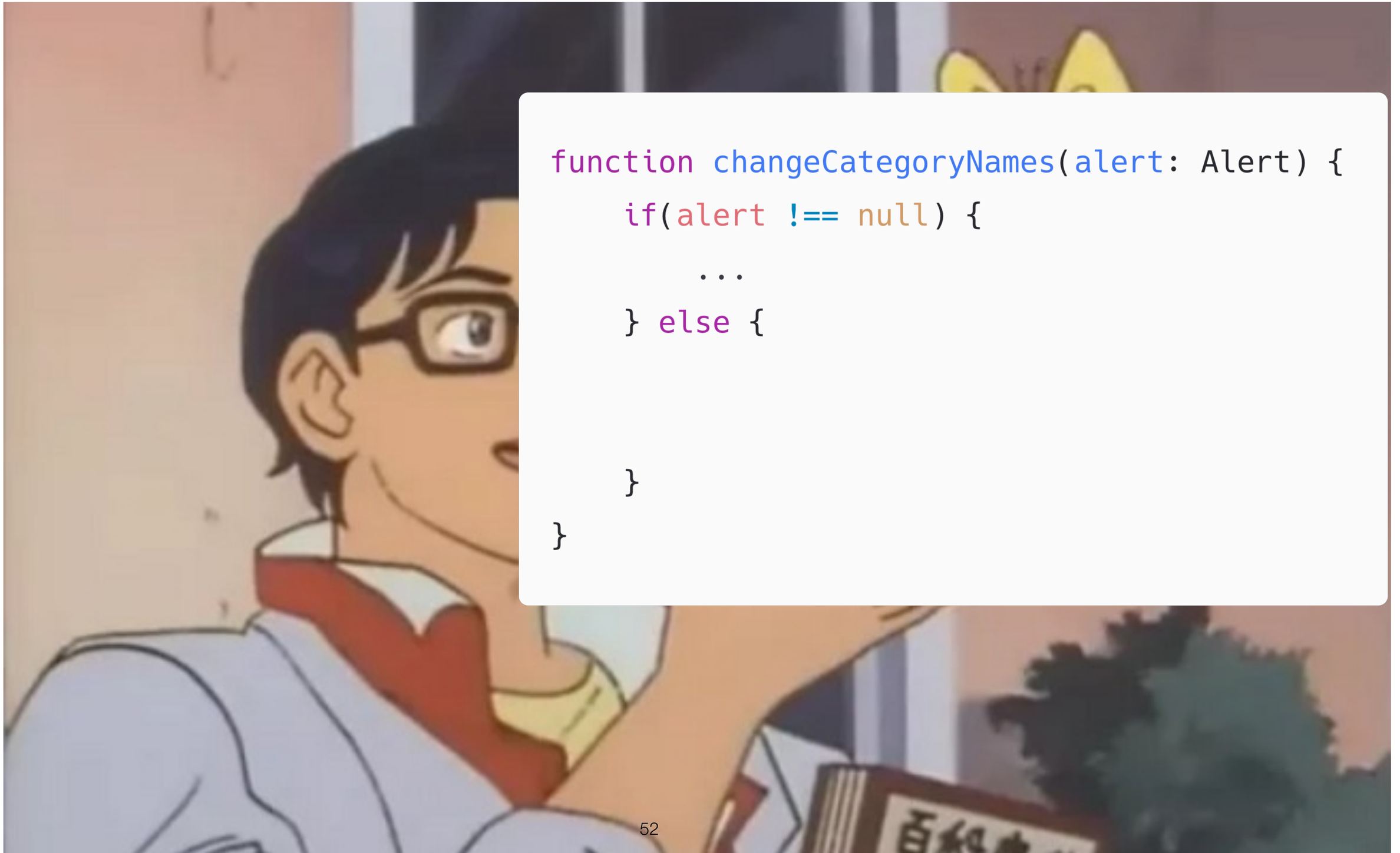


```
const transformFn: Alert => Alert
 = compose(
    newCategoryUI ? cha      oryNames : identity,
    ...
 )

// Later
transformFn(getActiveAlert())
```

Will not compile

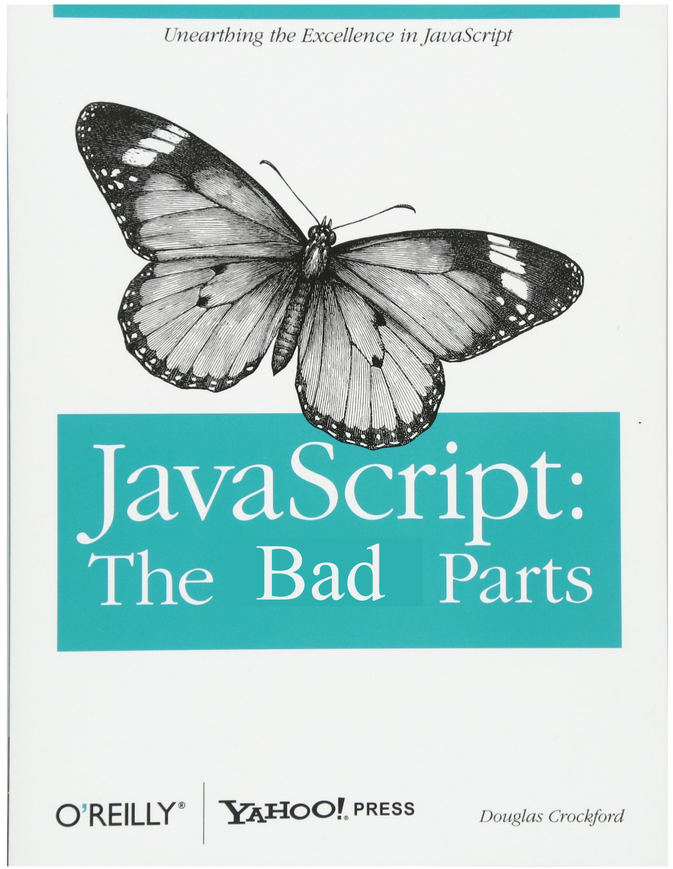
```
const transformFn: Alert => Alert
 = compose(
    newCategoryUI ? chan tegoryNames : identity,
    ...
 )

 // Later
 transformFn(getActiveAlert())
```

But how to deal with null?



```
function changeCategoryNames(alert: Alert) {
    if(alert !== null) {
        ...
    } else {



    }
}
```

```
function changeCategoryNames(alert: Alert) {
    if(alert !== null) {
        ...
    } else {
        🤔 Empty alert? Exception? …
    }
}
```



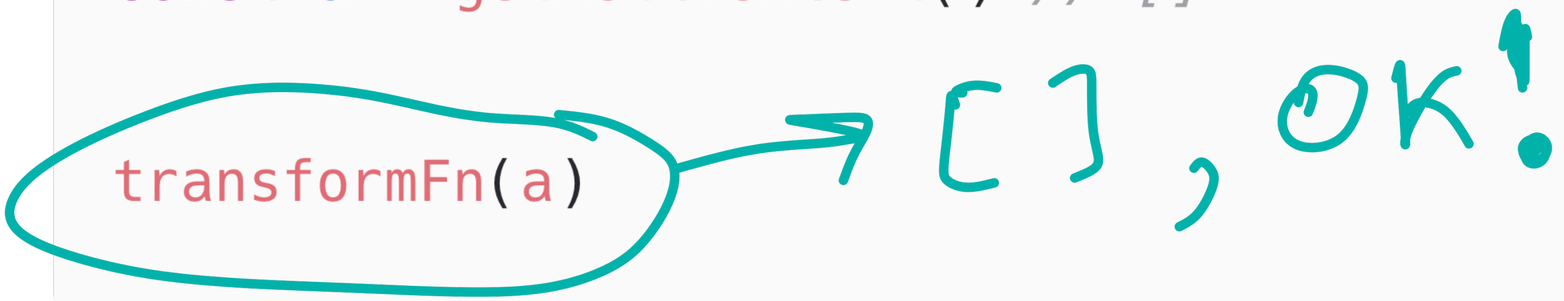
1) Null
2) Undefined


Array<Alert>

```
function changeCategoryNames(alert: Alert[]) {
    return alert.map(a => ({...a, category: "..."}))
}

const transformFn: Alert[] => Alert[] = compose(
    changeCategoryNames, …
)
```

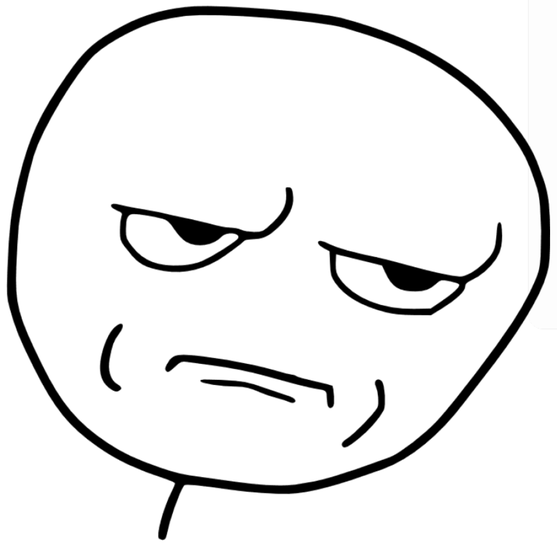
```
getActiveAlert(): Alert[]
```

```
function changeCategoryNames(alert: Alert[]) {
    return alert.map(a => a.category = "...")
}

const transformFn: Alert[] => Alert[] = compose(
    changeCategoryNames, …
)

const a = getActiveAlert() // []

transformFn(a)
```

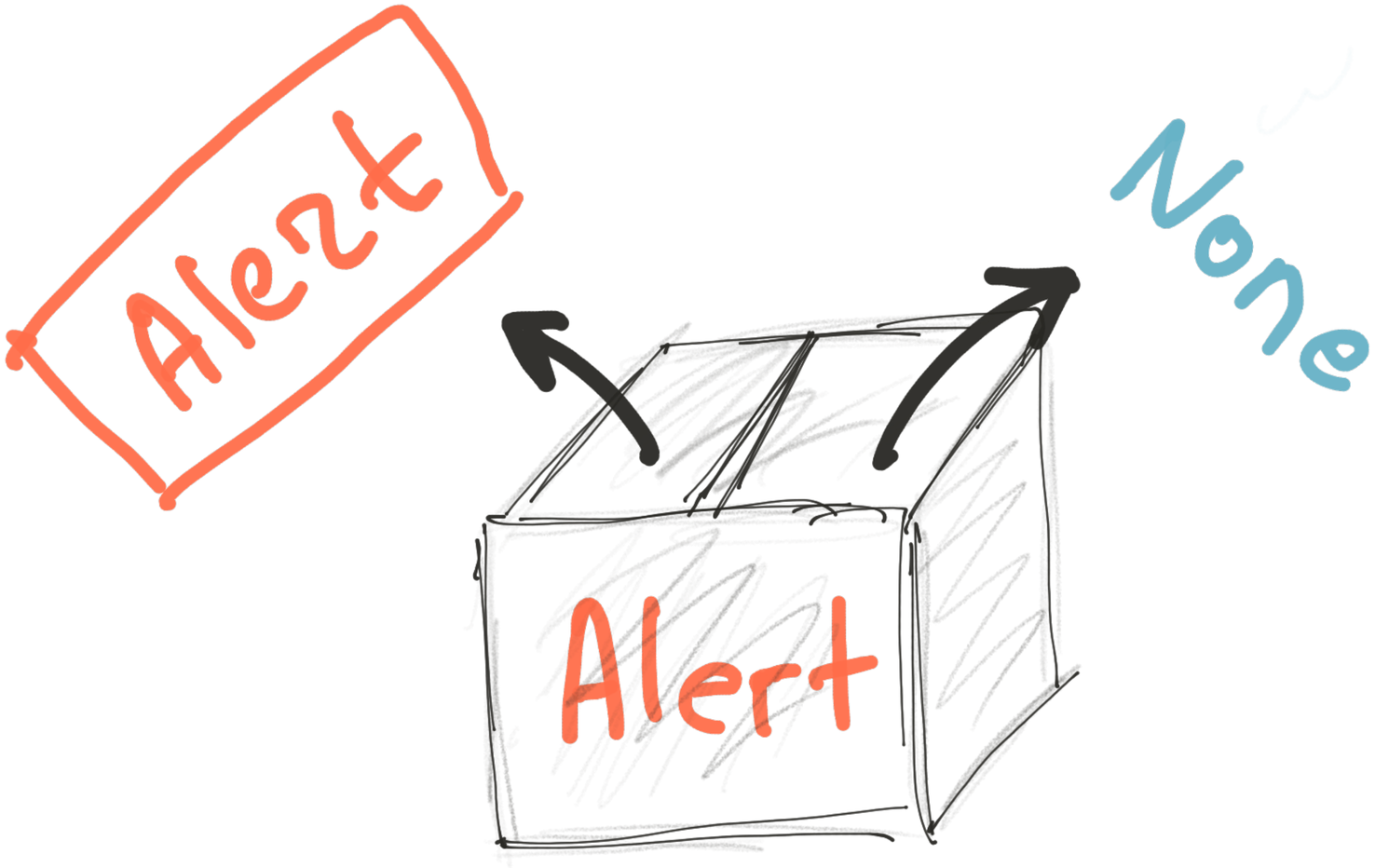


```
function changeCategoryNames(alert: Alert[]) {
    return alert.map(a => a.category = "...")
}

const t                                    pose(
    cha
)

transformFn([])
```

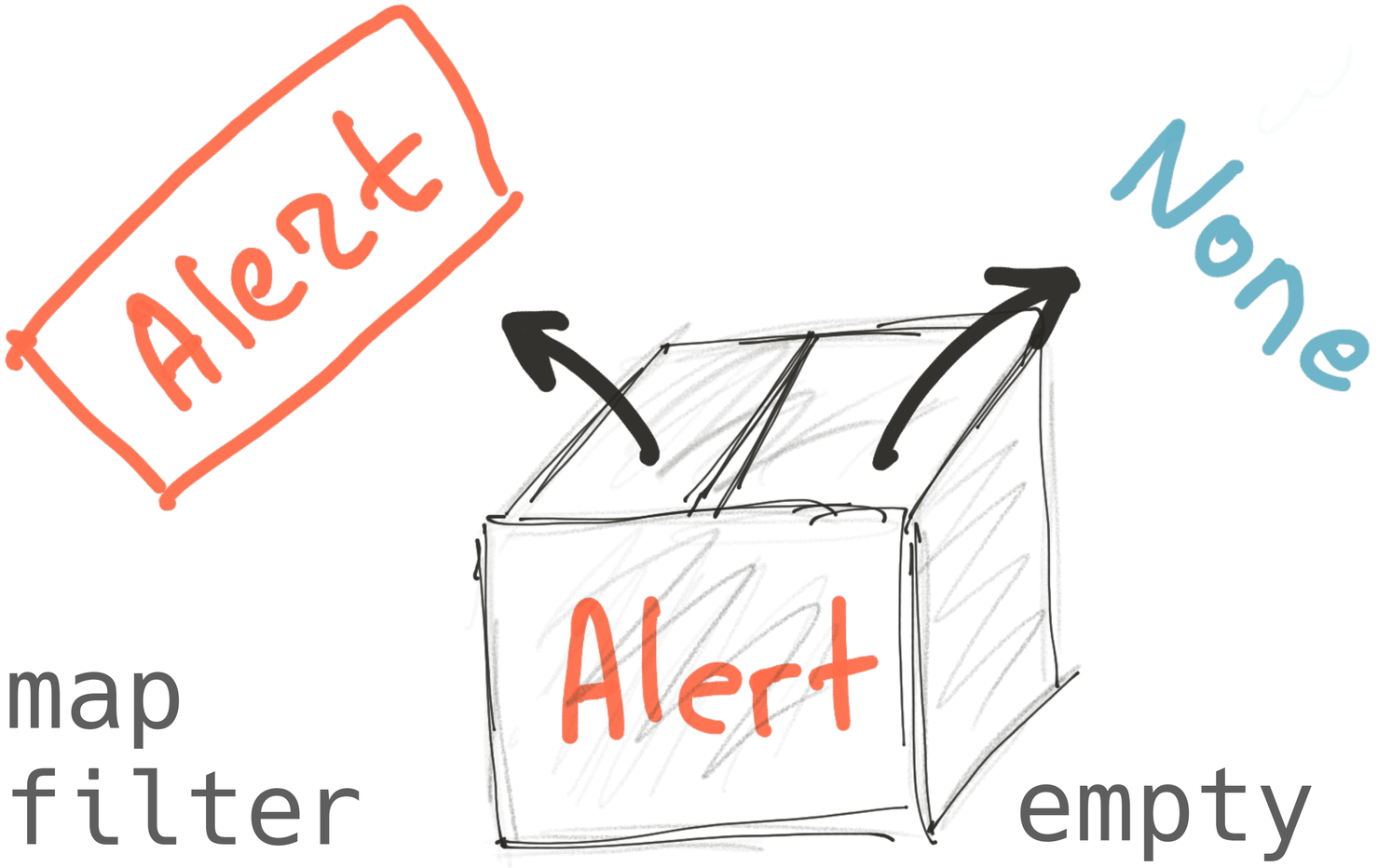
Alert[] !== Alert




λ



Option ≡ an array with 0…1 elements




```
map
filter
forEach
contains
…
```

```
empty
getOrElse
fold
…
```

```
function changeCategoryNames(alert: Option<Alert>) {
    return alert.map(a => ...)
}

const transformFn: Option<Alert> => Option<Alert> = compose(
    changeCategoryNames, …
)
```

```
const alert = Option.of(getCurrentAlert())
alert.map(a => ({...a, category: "..."}))
  .orElse(
      // Fallback
      Alert.createEmpty()
  )
```

```
function getOptionalAlert(): Option<Alert> {
  // no fallback, delegate to user
  return Option.of(getCurrentAlert())
    .map(...)
}

// Later, we handle empty value case
const alert: Alert = getOptionalAlert()
    .getOrElse(Alert.createEmpty())
```

What we achieved

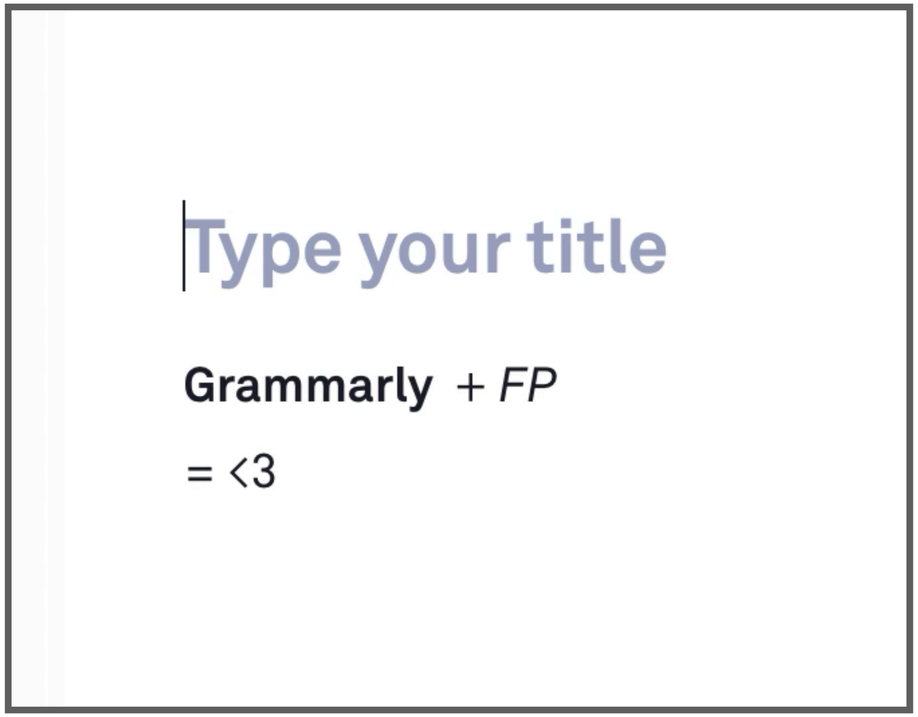
⛳ Reliable

Empty values are safe

⛳ Clean

No ifs overload, empty values handled separately

Case 3/4

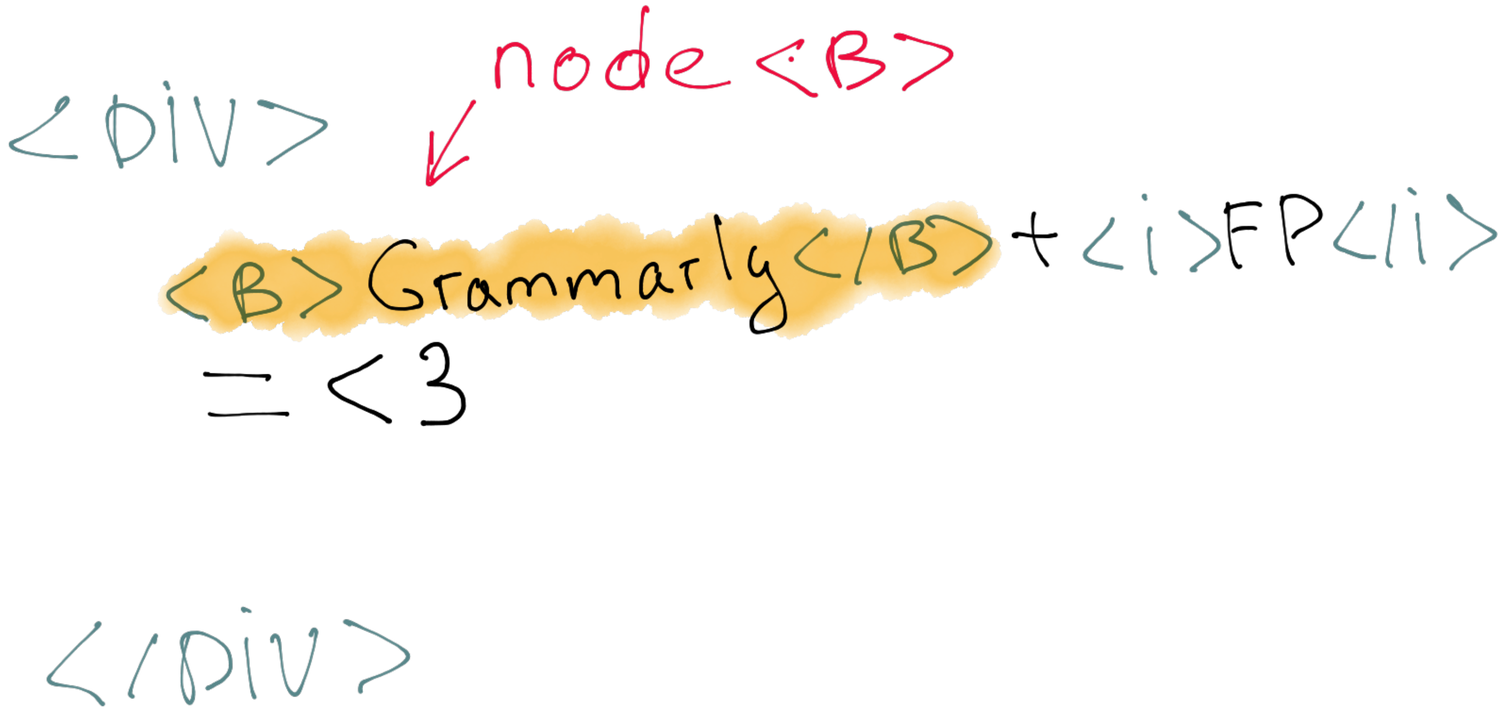

Type your title
Grammarly + FP
= <3

<DIV>

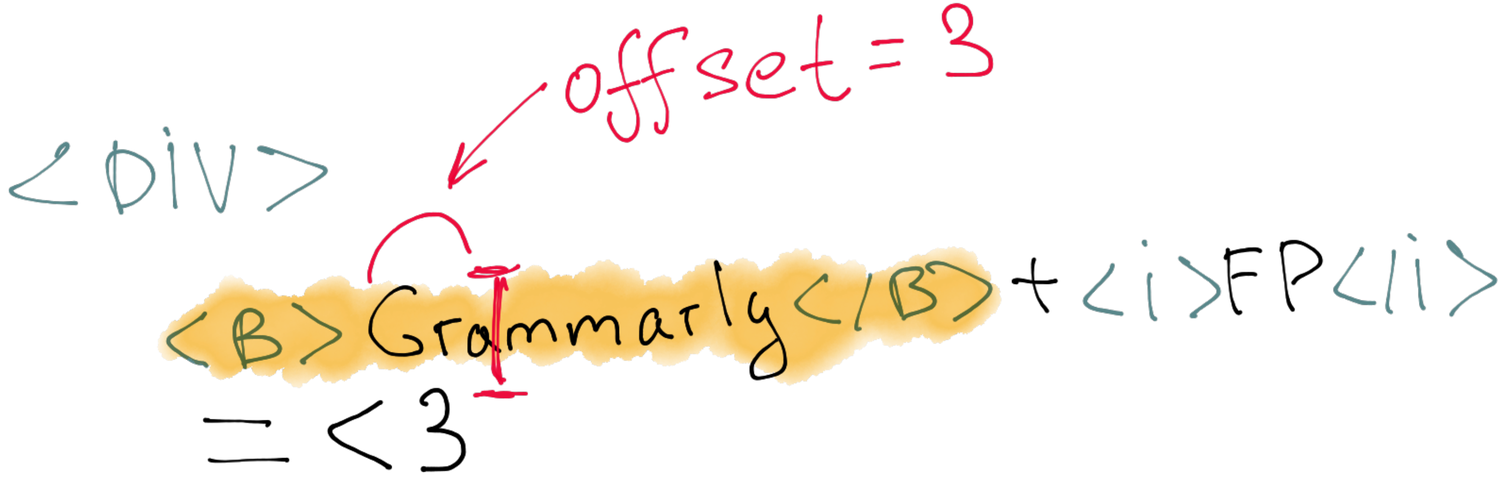
<B>Grammarly</B> + FP

=<3

</DIV>


<DIV>
node <B>
<B>Grammarly</B> + FP
=<3
</DIV>


<DIV>
<B>Grammarly</B> + FP
=<3
</DIV>


offset = 3
<DIV>
<B>Grammarly</B> + FP
=<3
</DIV>

```
const getCaretPosition = (x: number, y: number) => {
    const range = document.caretRangeFromPoint(x, y)
    return {
      node: range.startContainer,
      offset: range.startOffset
    }
}
```

```
const getCaretPosition = (x: number, y: number) => {
    const range = document.caretRangeFromPoint(x, y)
    return {
      node: range.startContainer,
      offset: range.startOffset
    }
}
```

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
`caretRangeFromPoint` ⚠	8	12	No	No	15	Yes

```
getCaretPosition(x, y).node
```

```
try {
  try {
      getCaretPosition(x, y)
  } catch(e) {   }
  try {

  } catch(e) {    }
} catch(e) {}
```

```
const getCaretPosition = (x: number, y: number) => {
        if (document.caretRangeFromPoint) {
          const range = document.caretRangeFromPoint(x, y)
          return {
            node: range.startContainer,
            offset: range.startOffset
          }
        } else
          return null
    }
```

Bad params

```
getCaretPosition(-1, -1) // null in Chrome

getCaretPosition(100, 100) // null in Firefox
```

no API

How to handle errors correctly?

```
getCaretPosition(x, y).map(({ node, offset } => {
 // use the result
}).recover(e: Error => {
 // recover from error
})
```

```
import {Try} from 'funfix'
```

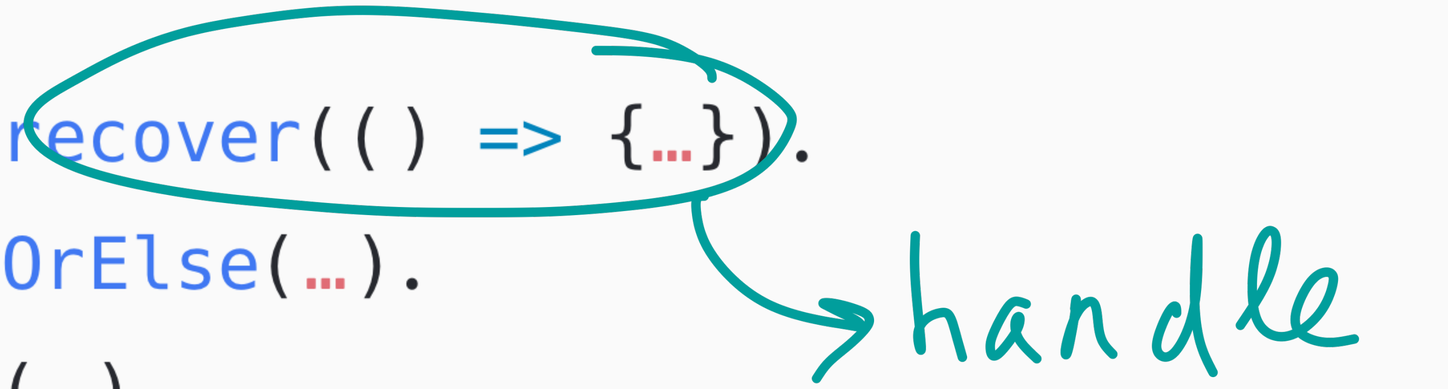
```ts
import { Try } from 'funfix'

const getCaretPosition = (x: number, y: number) =>
  Try.of(() => {
    document.caretRangeFromPoint(…).node
  })
})
```

transparent API

```
getCaretPosition(x, y).map(({ node, offset } => {
  …
}).recover(() => {…}).
getOrElse(…).
map(…).
filter(…)
```

```
Try<{ node: Node; offset: number }>
```

```
getCaretPosition(x, y).map(({ node, offset } => {
  …
}).recover(() => {…}).
getOrElse(…).
map(…).
filter(…)
```



What we achieved

⛳ Clean & Confident

No defensive programming

⛳ Singe Responsibility

Business logics is separated from error handling

Case 4/4


Ready to write something brilliant?
The Grammarly Editor helps you turn your ideas into clear, compelling writing.
START A QUICK TOUR
Skip the tour

```
import Onboarding from  "onboarding"


render: () =>
  <App>
    { isNewUser ?  <Onboarding /> : null }
  </App>
```

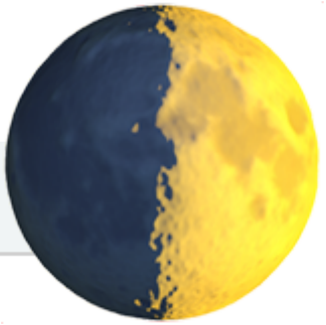
```
import Onboarding from "onboarding"

render: () =>
  <App>
    { isNewUser ? <Onboarding /> : null }
  </App>
```

many KBs of js code

React lazy loading

```
const Onboarding = React.lazy(import("/onboarding"))


render: () =>
  <App>
    { isNewUser ?  <Onboarding /> : null }
  </App>
```

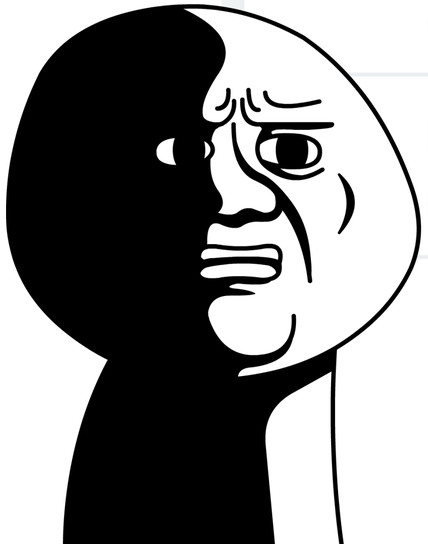
loads on demand



Messages | Aggregates

Page: 1 of 4 | Time Compare

#	exception
1	Loading chunk 12 failed. (missing: https://denali-static.grammarly.com/js/8bcd6f884fa390b07a8d/new-e.js)
2	Loading chunk 12 failed. (missing: https://denali-static.grammarly.com/js/61dcf2e401b255c92980/new-e.js)
	Loading chunk 12 failed. (missing: https://denali-static.grammarly.com/js/5043354ee6aeebc74daf/new-e.js)



How to retry when React lazy fails

Guilherme Oenning Nov 19 '18 *Updated on Dec 04, 2018* · 1 min read

#react #javascript

React 16.6 has been released and it's now easier than ever to do code split within our React applications by using lazy and Suspense.

If you don't know what I'm talking about, you should definitely read this https://reactjs.org/blog/2018/10/23/react-v-16-6.html

After a few days monitoring a production application that is using lazy, I noticed some client-side errors like this:

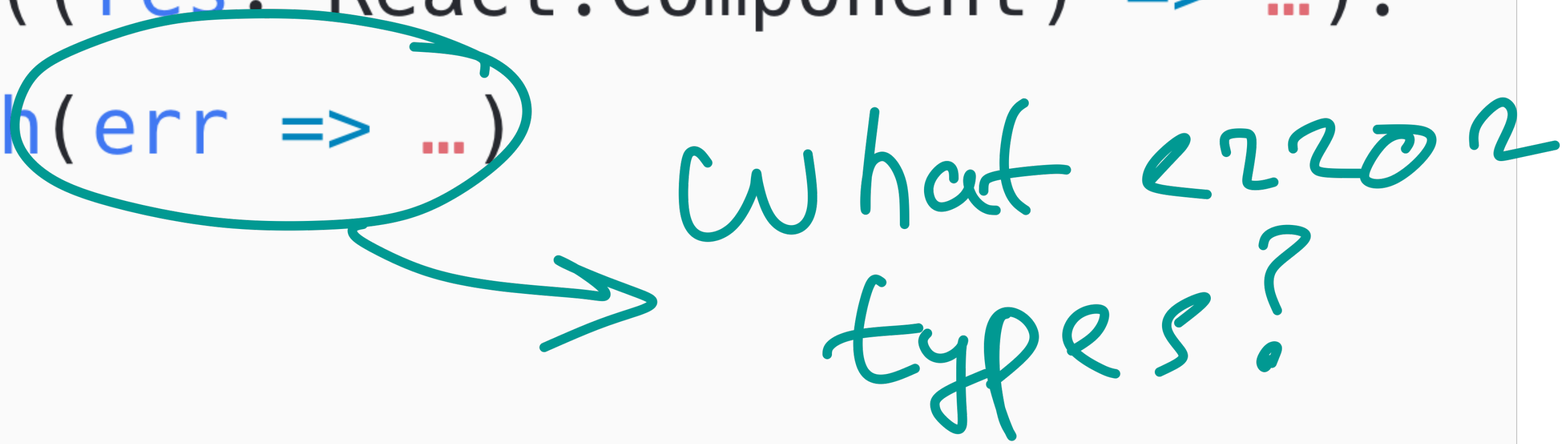
```
Loading chunk 6 failed. (error: https://.../6.4e464a072cc0e5e27a07.js)
Loading CSS chunk 6 failed. (https://.../6.38a8cd5e9daba617fb66.css)
```

```
retry(fn: () => Promise<Component>, retriesLeft = 5, interval = 1000)
```

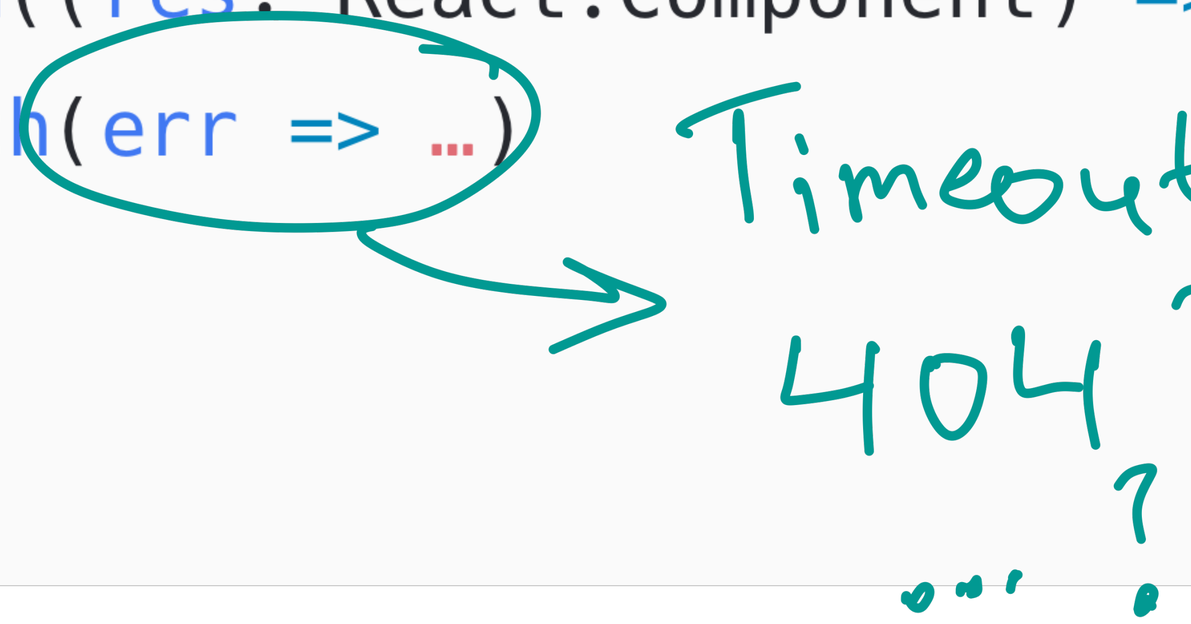
```
const somePromise = import("/onboarding")
```

```
React.lazy(
  () =>
     retry(() => somePromise).
          then((res: React.Component) => …).
          catch(err => …)

)
```

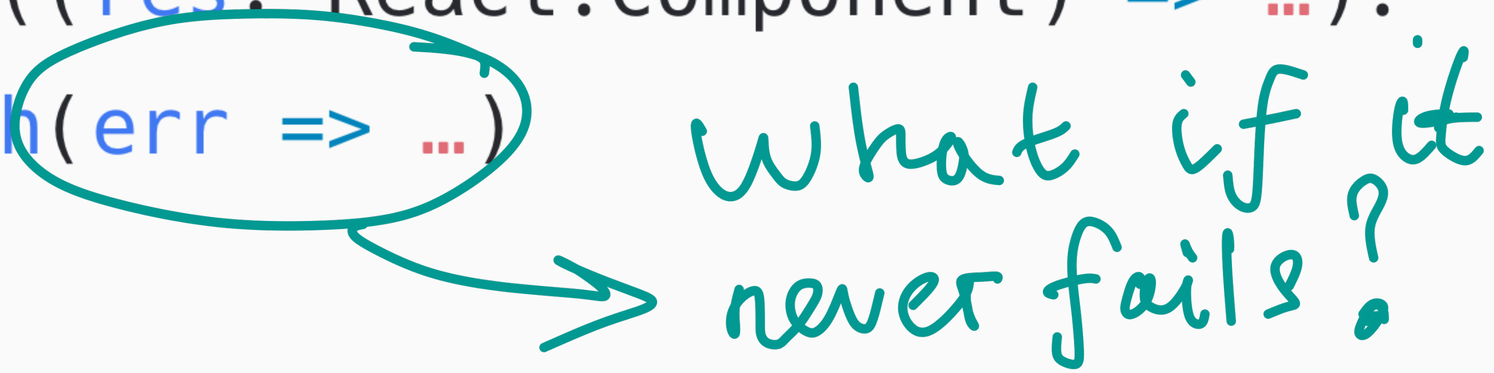
```
React.lazy(
  () =>
    retry(() => somePromise).
      then((res: React.Component) => …).
      catch(err => …)
)
```



```
React.lazy(
  () =>
    retry(() => somePromise).
      then((res: React.Component) => …).
      catch(err => …)
)
```



```
React.lazy(
  () =>
    retry(() => somePromise).
      then((res: React.Component) => …).
      catch(err => …)
)
```



```
React.lazy(
  () =>
    retry(() => somePromise, 5, 1000)
)
```

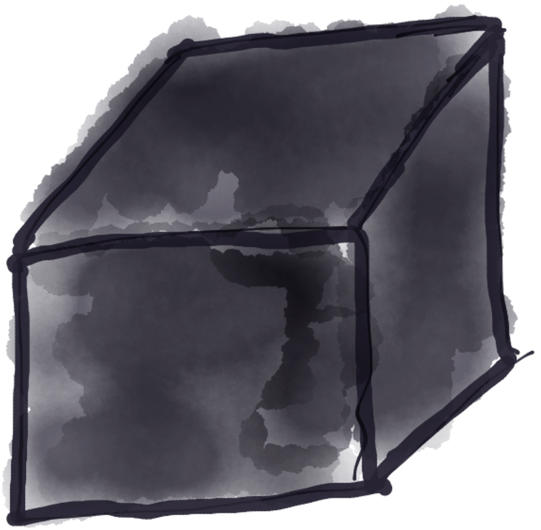
what if we want some custom logic?

```
React.lazy(
  () =>
    retry(() => somePromise, 5, 1000)
)
```

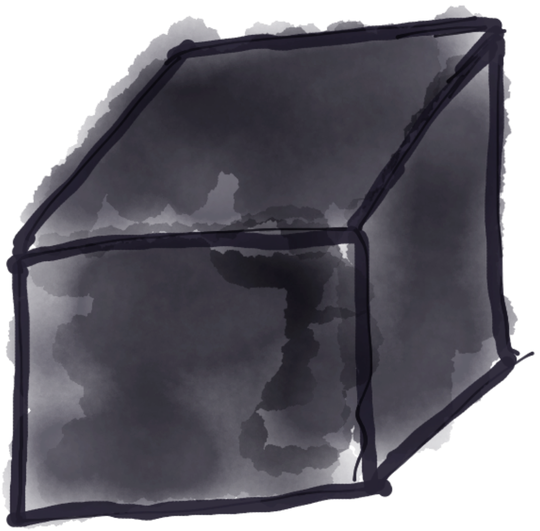
what if we want to abort early or log something?

How to build a *good* retry function?


ЗНАТОКИ
ТЕЛЕЗРИТЕЛИ
0
0
ЧТО? ГДЕ? КОГДА?



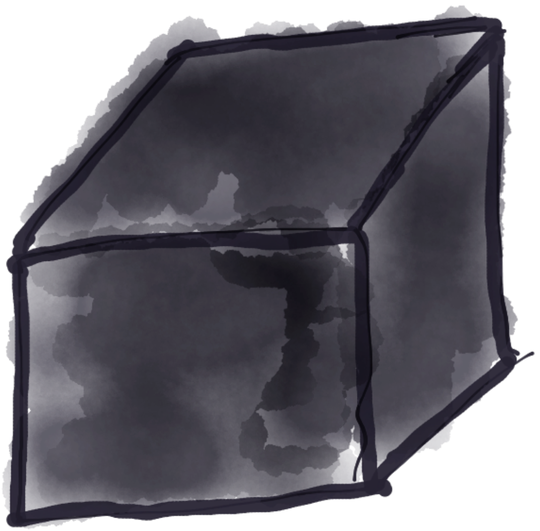
what we want



```
retry(…)
```

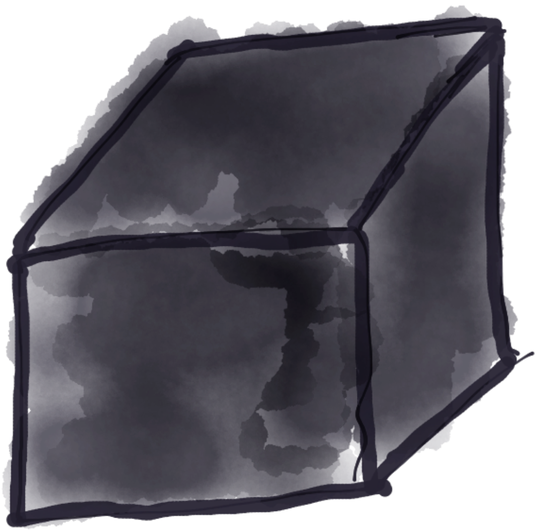
what we want

Aborting



what we want

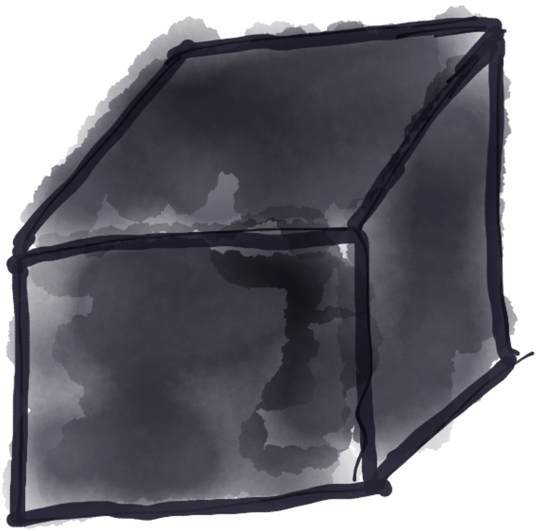
Aborting



Log

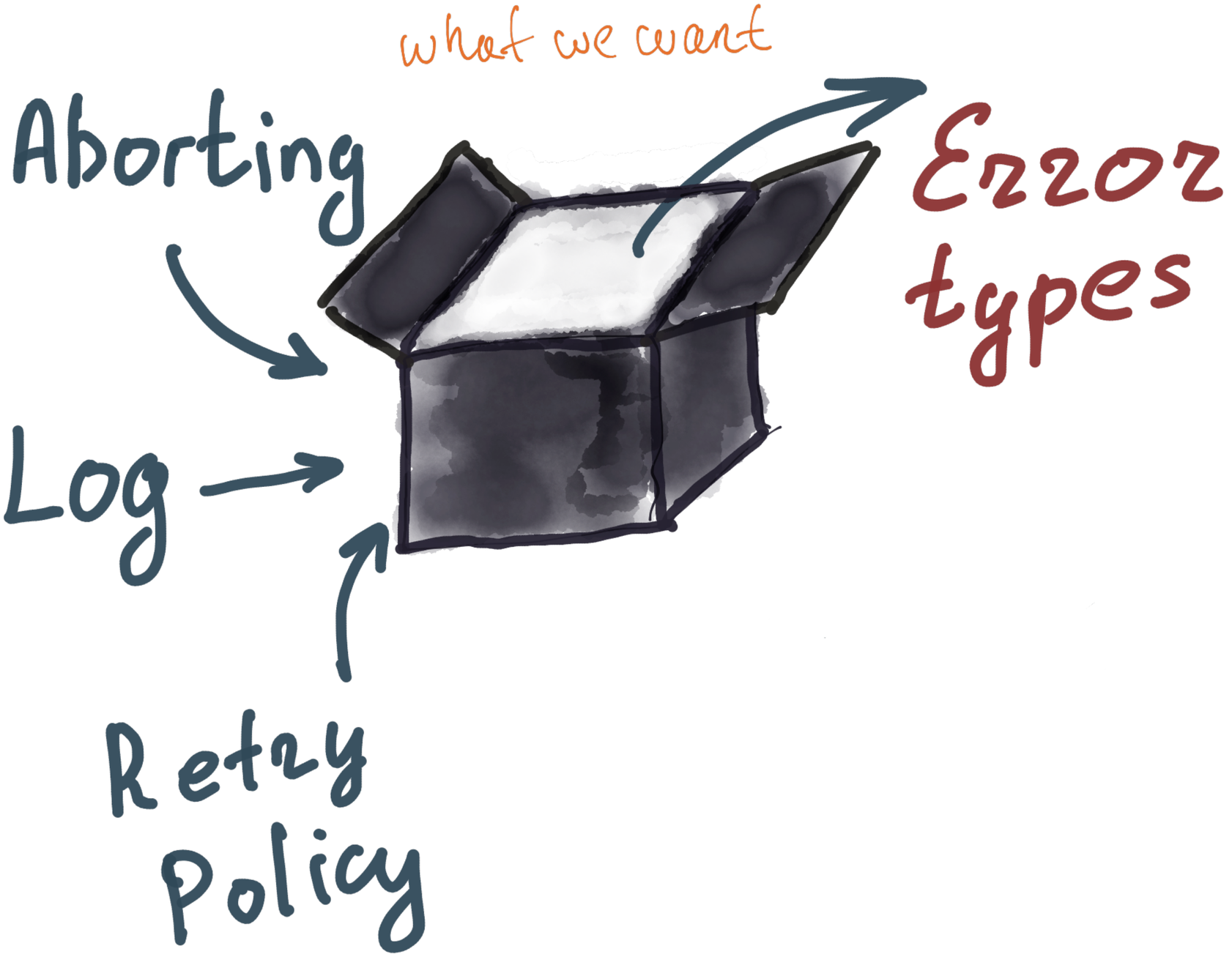
what we want

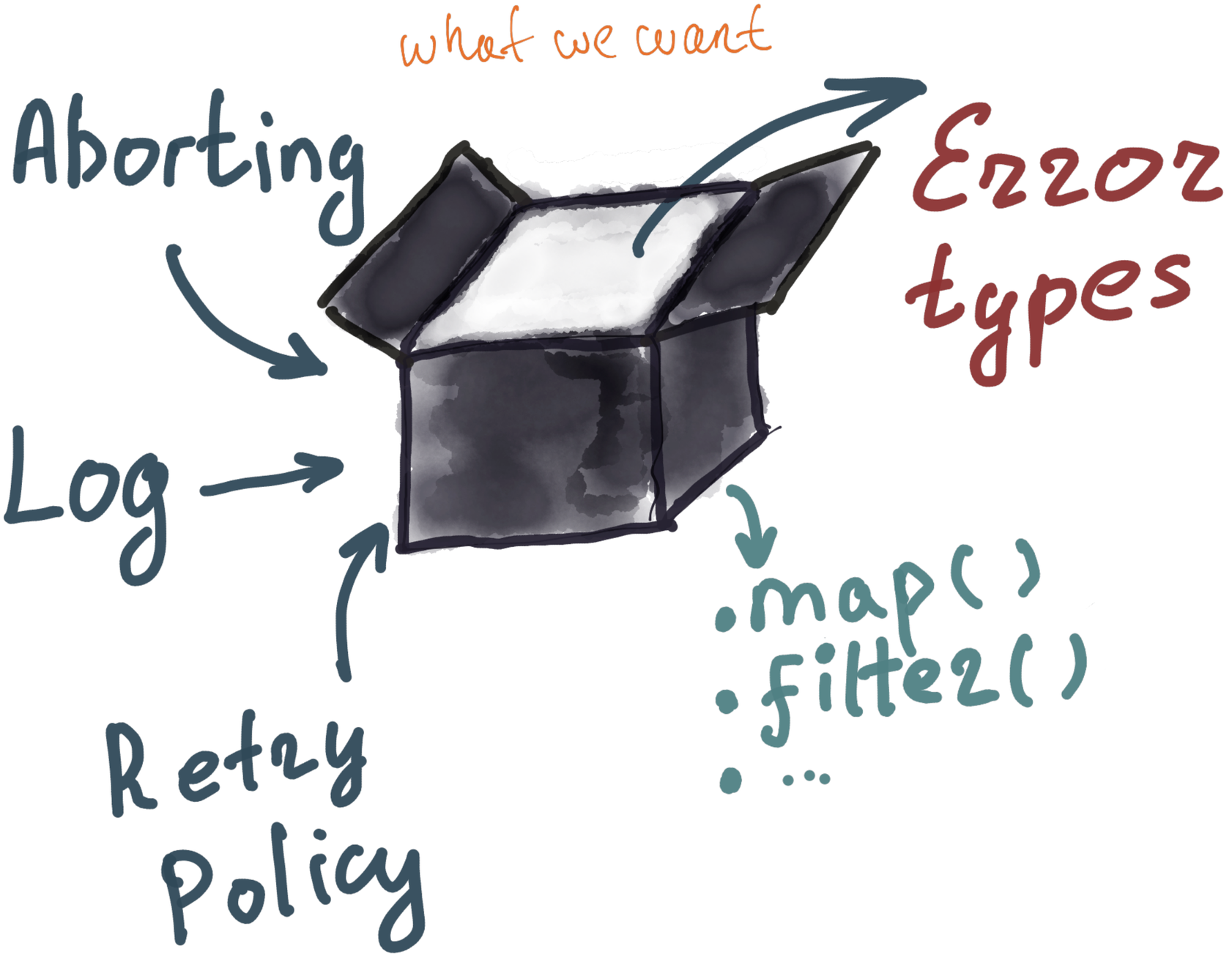
Aborting



Log

Retry Policy


what we want
Aborting
Log
Retry Policy
Error types

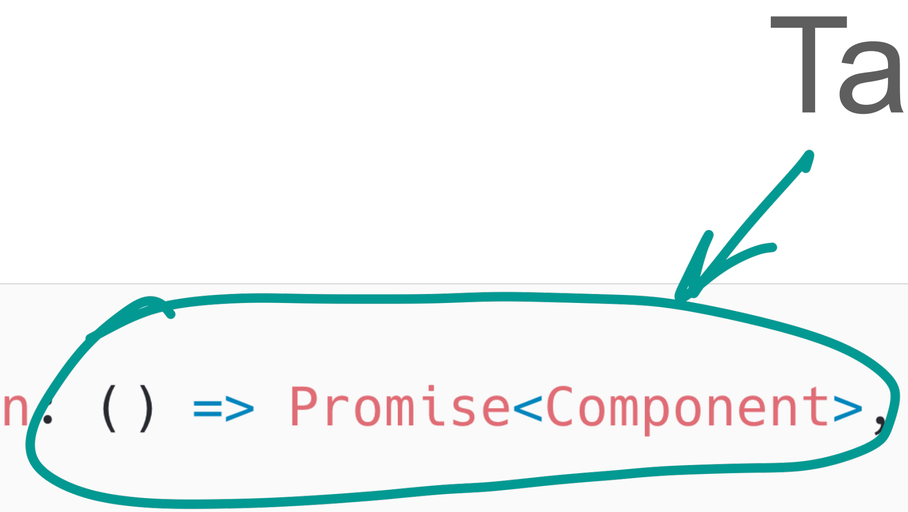

what we want
Aborting
Log
Retry Policy
Error types
.map()
.filter()
...

Lets check github.com/gcanti/**fp-ts**

Promise in FP

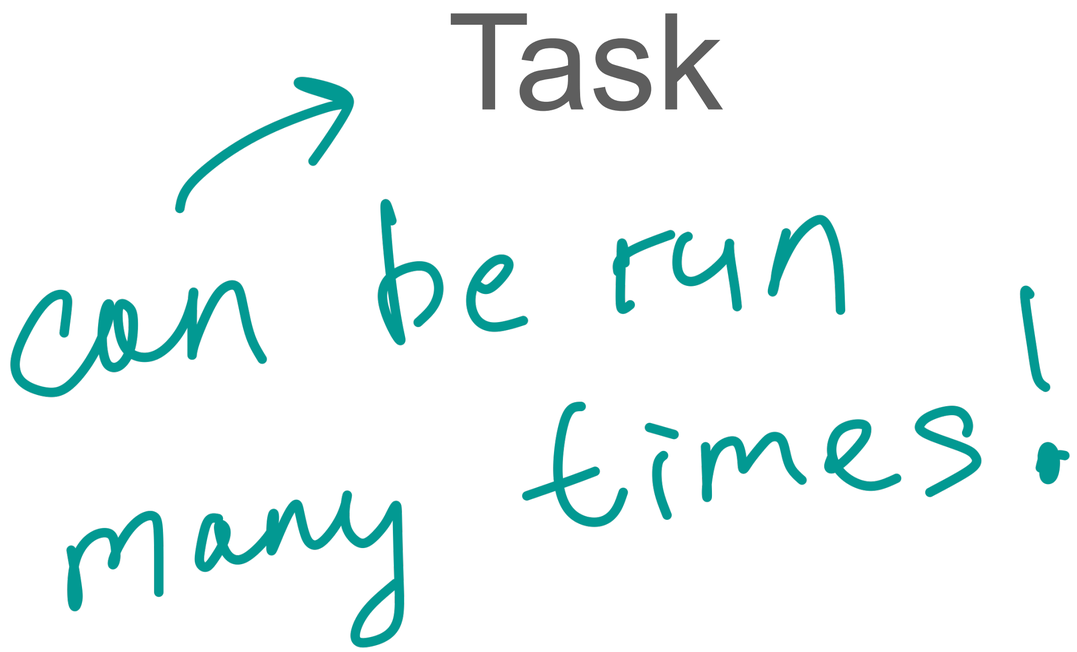
Task

Promise in FP




```
retry(fn: () => Promise<Component>, retriesLeft = 5, interval = 1000)
```

Promise in FP



Error + Result container

Either<Error, Result>

Error + Result container

Either<Error, Result>

like Try type, but knows the Error type

Error + Result container

Either<Error, React.Component>

where type Error = TimeoutError | 404Error

Task returning Either

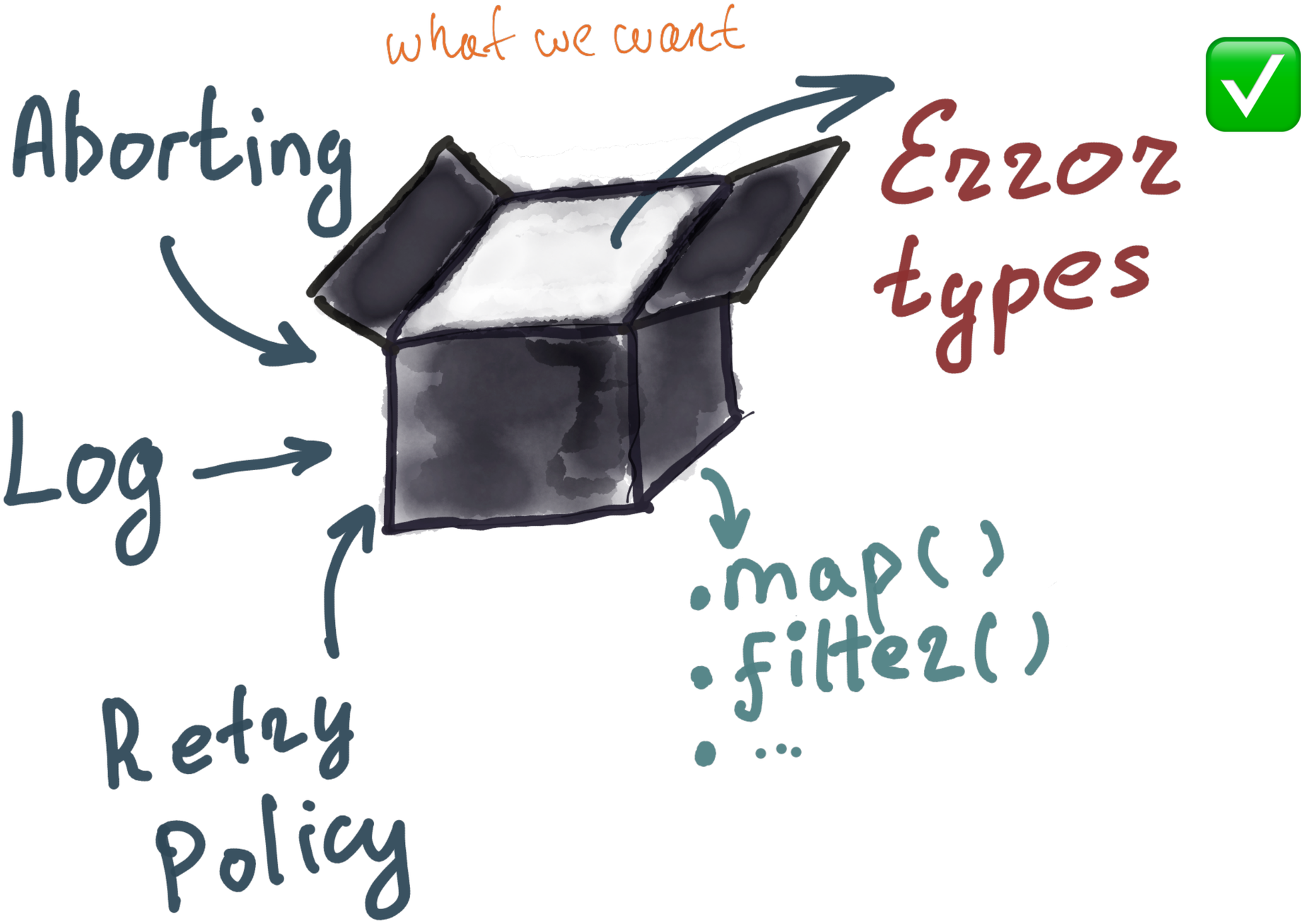
Task<Either<Error, Result>>

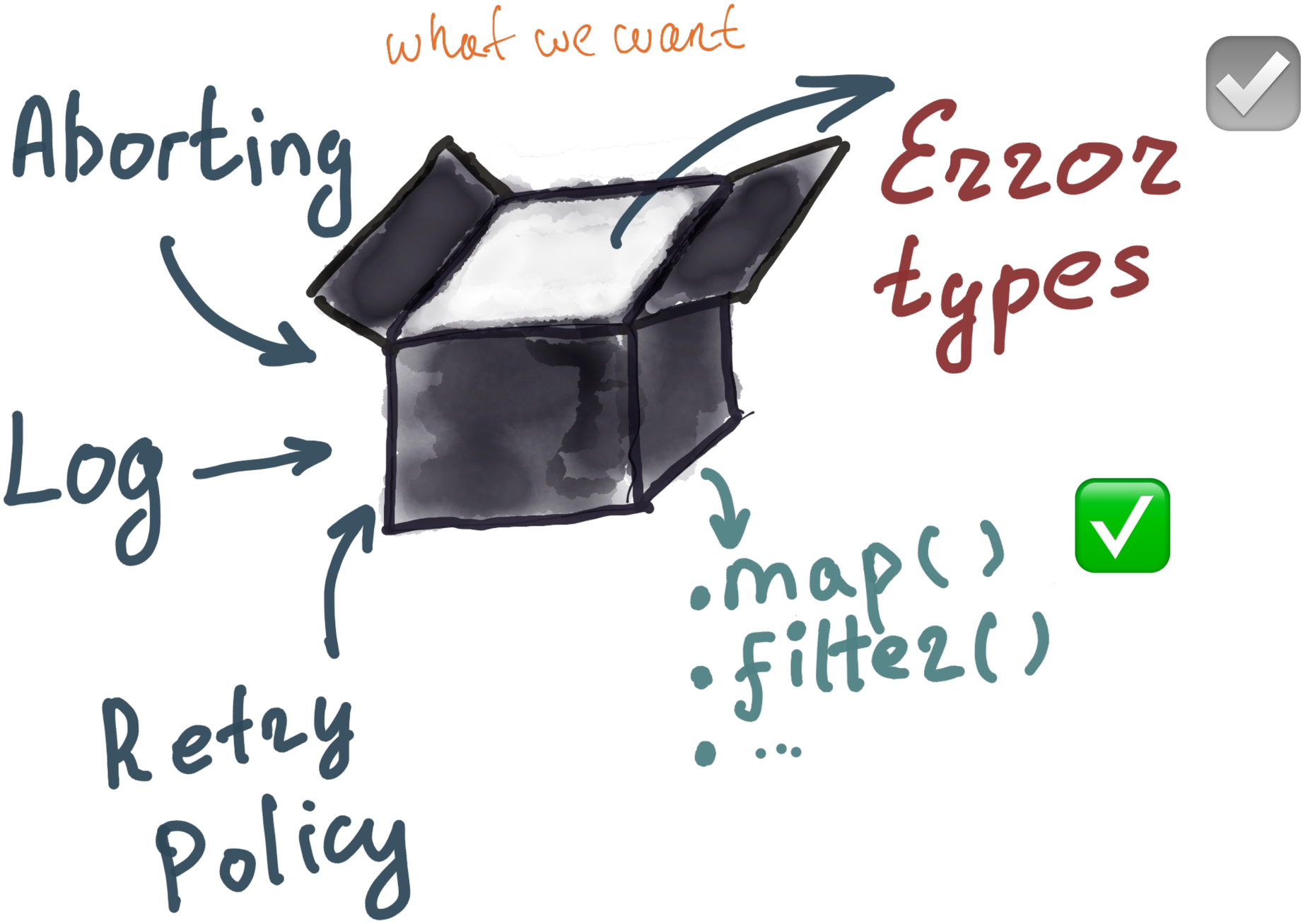
alias: TaskEither<Error, Result>

Task returning Either

TaskEither<Error, Result>

() => Promise< Either<Error, Result> >


what we want
Aborting
Log
Retry Policy
Error types
.map()
.filter()
...


what we want
Aborting
Log
Retry
Policy
Error
types
.map()
.filter()
...

github.com/gcanti/**retry-ts**

by the author of fp-ts

```
retrying( policy: Policy, action: Action, check: Check ): Task<T>
```

```
retrying( policy: Policy, action: Action, check: Check ): Task<T>
```

```
Either<Error, Component>
```

```
retrying( policy: Policy, action: Action, check: Check )
```

You can’t just stop

speaking about FP

```
retrying( policy, action, check )
```

```
retrying( policy, action, check )
```

```
action: (status: RetryStatus) => TaskEither<Error, Result>
```

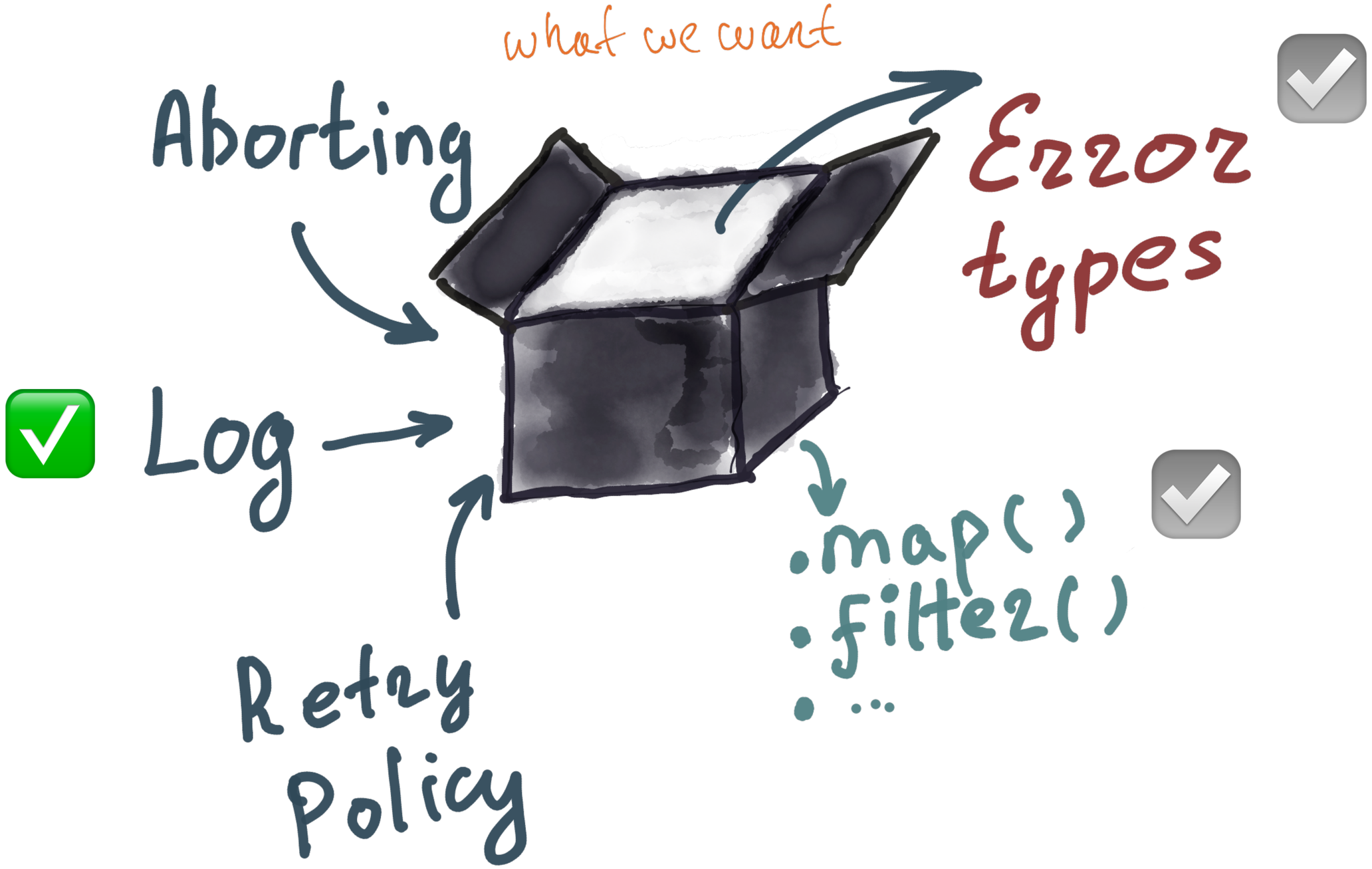
```
const taskEither = tryCatch<TimeoutError, Component>(
  () => import("/onboarding"),
  reason => new TimeoutError(…)
)

const action = status => {
  return taskEither
}
```

```
const taskEither = tryCatch<TimeoutError, Component>(
  () => import("/onboarding"),
  reason => new TimeoutError(…)
)

const action = status => {
  return taskEither
}
```

Promise to Task Either

```
const taskEither = tryCatch<TimeoutError, Component>(
  () => import("/onboarding"),
  reason => new TimeoutError(…)
)

const action = status => {
  if (status.previousDelay.isSome())
    log("Houston, we have a problem with this component")
  return taskEither
}
```

```
retrying( policy, action, check )
```
✅


what we want
Aborting
Log
Retry Policy
Error types
.map()
.filter()
...

```
retrying( policy, action, check )
```

```
retrying( policy, action, check )
```

Left Right

```
function check(either: Either<Error, Component>): boolean {
  return isLeft(either)
}
```

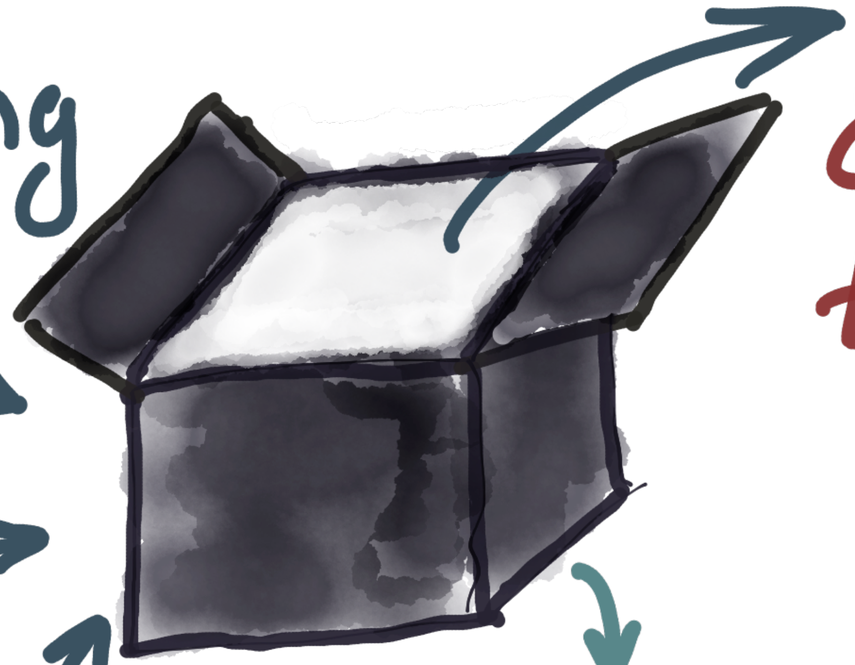
```
retrying( policy, action, check )
```

```
function check(either: Either<Error, Component>): boolean {
  return isLeft(either) && !(either.left instanceof Error404)
}
```

```
retrying( policy, action, check )
```

what we want

✅ Aborting



Error types ☑

☑ Log →

Retry Policy

.map()
.filter()
...

☑

```
retrying( policy, action, check )
```

```
interface RetryPolicy {
    (status: RetryStatus): Option<number>
}
```

```
import { limitRetries } from 'retry-ts'

const policy = limitRetries(50)
```

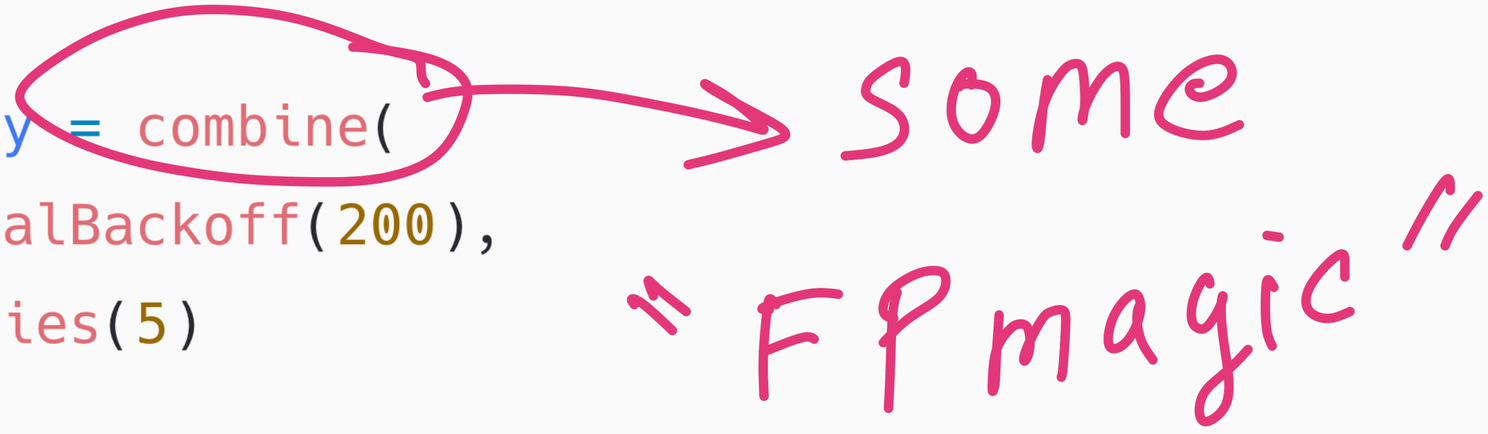
```
retrying( policy, action, check )
```

```
import { limitRetries, exponentialBackoff, combine } from 'retry-ts'

const policy = combine(
  exponentialBackoff(200),
  limitRetries(5)
)
```

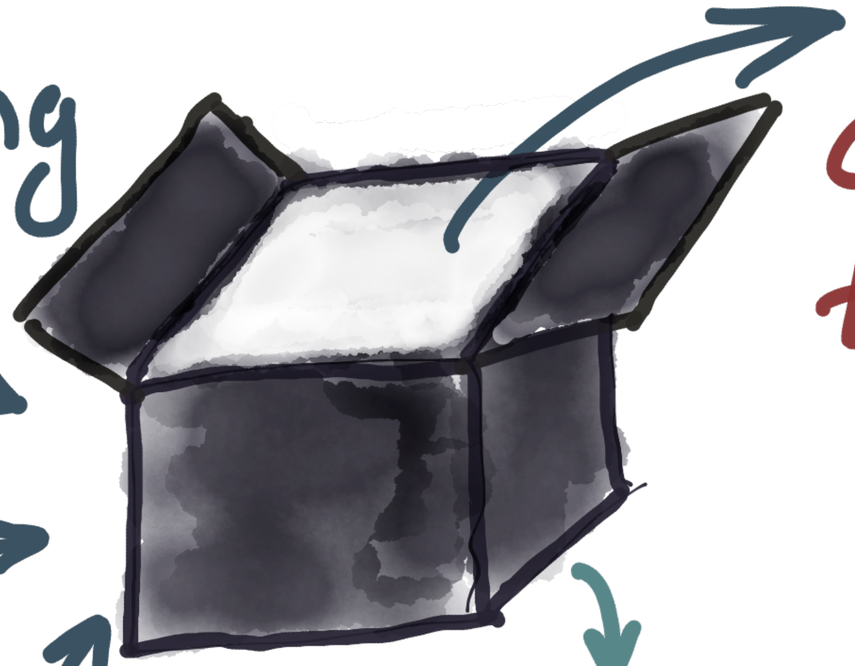
↗ increase delay, start from 200 ms

```
retrying( policy, action, check )
```

```
import { limitRetries, exponentialBackoff, combine } from 'retry-ts'

const policy = combine(
  exponentialBackoff(200),
  limitRetries(5)
)
```



what we want



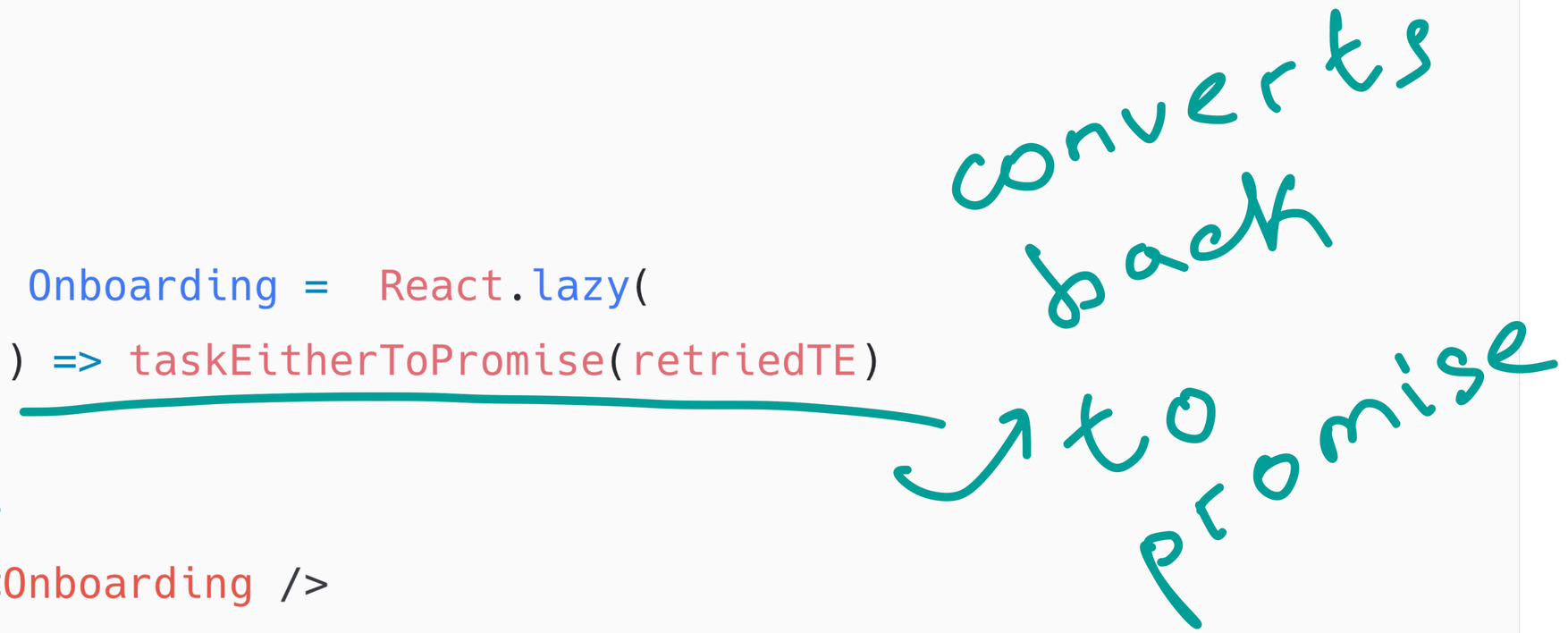
- Aborting
- Log
- Retry Policy

Error types

- .map()
- .filter()
- ...

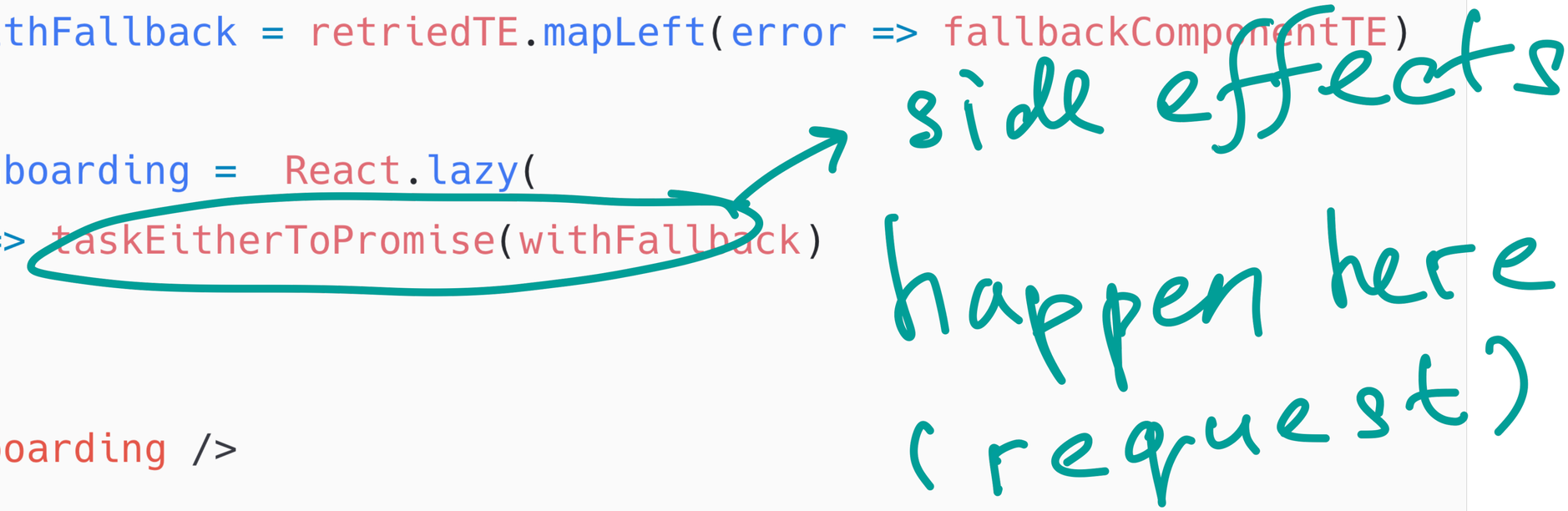
```
const retriedTE: TaskEither<ServerError, Component> = retrying(
    policy,
    action, // includes import("/onboarding")
    check
)



const Onboarding =  React.lazy(
    () => taskEitherToPromise(retriedTE)
)
<App>
    <Onboarding />
</App>
```

```
const retriedTE: TaskEither<ServerError, Component> = retrying(
    policy,
    action, // includes import("/onboarding")
    check
)



const Onboarding =  React.lazy(
    () => taskEitherToPromise(retriedTE)
)
<App>
    <Onboarding />
</App>
```



```
const retriedTE: TaskEither<ServerError, Component> = retrying(
    policy,
    action, // includes import("/onboarding")
    check
)
// Let's add a fallback
const withFallback = retriedTE.mapLeft(error => fallbackComponentTE)

const Onboarding =  React.lazy(
    () => taskEitherToPromise(withFallback)
)
<App>
    <Onboarding />
</App>
```

```
const retriedTE: TaskEither<ServerError, Component> = retrying(
    policy,
    action, // includes import("/onboarding")
    check
)
// Let's add a fallback
const withFallback = retriedTE.mapLeft(error => fallbackComponentTE)

const Onboarding =  React.lazy(
    () => taskEitherToPromise(withFallback)
)
<App>
    <Onboarding />
</App>
```



Code examples: done

What we achieved

⛳ Flexible

Combine different retry policies

⛳ Predictable

Know your errors in advance

⛳ Isolated side effects

Combine complex tasks without execution

Problem - Solution

Empty values ➡️ **Option**

Problem - Solution

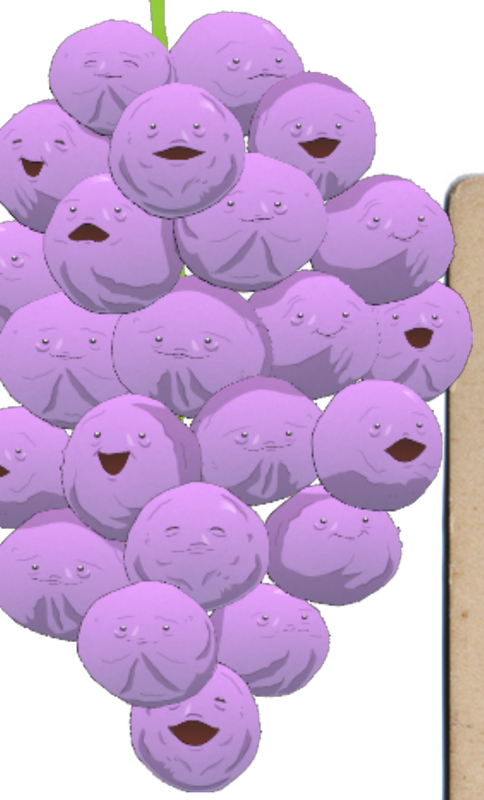
Empty values ➡️ **Option**

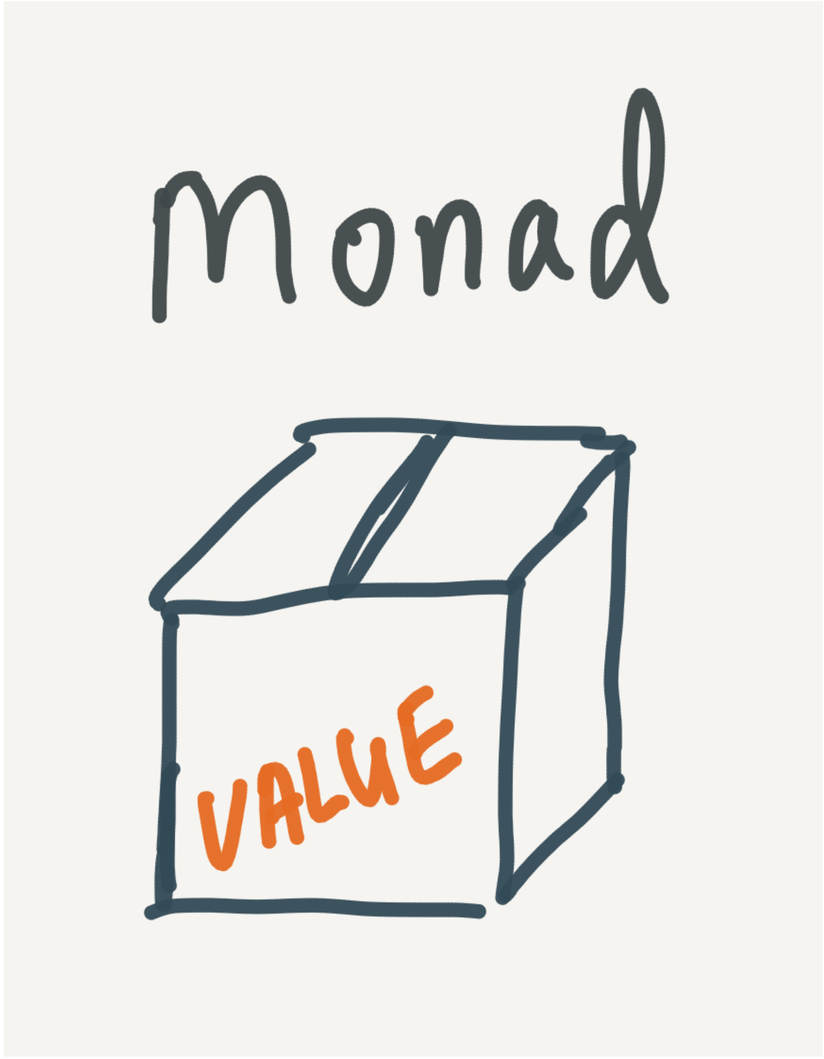
Exceptions ➡️ **Try** (or Either)

Problem - Solution

Empty values	➡️	**Option**
Exceptions	➡️	**Try** (or Either)
Async side effects	➡️	**Task / TaskEither**

Remember?





Option

Try, Either

Task, TaskEither

One last thing

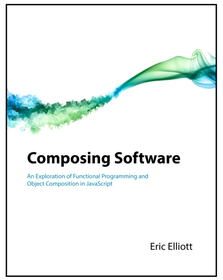
FP is about *composition:*

- Function composition
- Monadic composition

Conclusions

Twitter: @oopets

The last thing



Composing Software: The Book
E. Elliot

Functional Programming in Scala
P. Chiusano, R. Bjarnason

Blog: https://dev.to/gcanti (fp-ts)