MODÉLISER UNE ERREUR POTENTIELLE

EITHER = RESULT

```ts
type Left<E> = { kind: "Left", left: E }
type Right<A> = { kind: "Right", right: A }
type Either<E, A> = Left<E> | Right<A>
const isNat = (x: number): Either<string, number> =>
  x > 0
    ? { kind: "Right", right: x }
    : { kind: "Left", left: "Not a natural number" }

const foldNumber = (y: Either<string, number>, defaultValue: number): number => {
  switch (y.kind) {
    case "Right": return y.right;
    case "Left": return defaultValue;
  }
}
```

Either est en réalité plus générique car représente n’importe quel arbre binaire

Par convention, il est souvent utilisé avec Right (correct) contenant la valeur ok, et Left (abandon) contenant la valeur error

MODÉLISER UNE ERREUR POTENTIELLE

AVEC FP-TS

```ts
import * as E from 'fp-ts/Either'
import { Either } from 'fp-ts/Either'
import { pipe } from "fp-ts/lib/function";

const isNat = (x: number): Either<string, number> =>
  x > 0
    ? E.right(x)
    : E.left("Not a natural number")

const foldNumber = (y: Either<string, number>, defaultValue: number): number =>
  pipe(
    y,
    E.match(
      (_: string) => defaultValue, // onLeft handler
      (value: number) => value // onRight handler
    )
  )
```