ERREURS

FP-TS

MIAGE M2 - QUALITÉ DU SI - THOMAS HAESSLÉ

```ts
import { pipe } from 'fp-ts/function';
import * as E from 'fp-ts/Either';
type State = E.Either<Error, Idle | Moving>
const initialState: State = pipe(idle(), E.right);
//⚠️  Redux is js first, so state parameter must by a subtype of ` any | undefined`
const reducer: Reducer<State, Action> = (state: State | undefined, action: Action) => {
    if (!state) return initialState //mandatory by Redux
    // ⚠️ `chain` is `flatmap` or `bind` name in fp-ts;
    // be carefull bind is js Function.prototype.bind ... naming are hard
    return E.chain(
        // 😟 TS inferance is bad, you will often need to help the typer
        (state: Idle | Moving): E.Either<Error, Idle | Moving> => {
            switch (state.type) {
                case "idle": {
                    // ⚠️ Redux action MUST be tagged on a `type` attribute
                    switch (action.type) {
                        case "face": return pipe(idle(), E.right); // idle () |> E.right
                        case "start": return pipe(moving(), E.right);
                        case "stop": return pipe(new Error("Illegal Action from Idle")
                                                , E.left);// 😎
                    }
                }
                case "moving": {
                    switch (action.type) {
                        case "stop": return pipe(idle(), E.right);
                        default: return pipe(new Error("Illegal Action from Moving"),
                                             E.left);// 😎
                    }
                }
            }
        }
    )(state)
}
```

Design simpliste pour exemple

Que faire quand le state est en erreur ?

Action Init?

Reinit auto ?

Conserver le state avant erreur ?

Dans un state plus complexe, on ne veut pas forcément tout dans un Either : c'est ok!

ERREURS

FP-TS

Design simpliste pour exemple

Que faire quand le state est en erreur ?

Action Init?

Reinit auto ?

Conserver le state avant erreur ?

Dans un state plus complexe, on ne veut pas forcément tout dans un Either : c'est ok!

```ts
import { pipe } from 'fp-ts/function';
import * as E from 'fp-ts/Either';
type State = E.Either<Error, Idle | Moving>
const initialState: State = pipe(idle(), E.right);
//⚠️  Redux is js first, so state parameter must by a subtype of ` any | undefined`
const reducer: Reducer<State, Action> = (state: State | undefined, action: Action) => {
    if (!state) return initialState //mandatory by Redux
    // ⚠️ `chain` is `flatmap` or `bind` name in fp-ts;
    // be carefull bind is js Function.prototype.bind ... naming are hard
    return E.chain(
        // 😟 TS inferance is bad, you will often need to help the typer
        (state: Idle | Moving): E.Either<Error, Idle | Moving> => {
            switch (state.type) {
                case "idle": {
                    // ⚠️ Redux action MUST be tagged on a `type` attribute
                    switch (action.type) {
                        case "face": return pipe(idle(), E.right); // idle () |> E.right
                        case "start": return pipe(moving(), E.right);
                        case "stop": return pipe(new Error("Illegal Action from Idle")
                                                , E.left);// 😎
                    }
                }
                case "moving": {
                    switch (action.type) {
                        case "stop": return pipe(idle(), E.right);
                        default: return pipe(new Error("Illegal Action from Moving"),
                                             E.left);// 😎
                    }
                }
            }
        }
    )(state)
}
```

ÉCOSYSTÈME REACT

NOTRE ENVIRONNEMENT DE TP

Une application plus complexe nécessite de pouvoir chainer les actions et de gérer des actions asynchrones

Nous utiliserons:

- **redux-loop qui implémente le pattern Elm dans une vision puriste**
- **fp-ts pour disposer des types Option et Either**
- **Fast Check pour les PBT**

Interdiction d'utiliser l'état local !