





```
const hitMonsterIf = (w : Weapon | null, t : Target | null) : Impacted | null =>
  (w && t)
  ? new Impacted()
  : null;
```

`null` est un type littéral qui a une valeur : `null`

Si `strictNullChecks=true` dans `tsconfig.json` alors utiliser `null` est sound

Mais il y a beaucoup de confusion sémantique entre `undefined` (valeur non initialisée) & `null` (absence de valeur)

Les manipulation de type nullable ou leur expressivité est perfectible

**MODÉLISER UN ÉVÉNEMENT EN VALEUR**

TRAINERS VARNER DRONER LES ENT

MAGN2-QUALITÉ D'SI-THO MAS HAE SLÉ & DUE NINBURG

```
import * as O from 'fp-ts/Option'
import { Option } from 'fp-ts/Option'
const hitMonsterOp = (w: Option<Weapon>, t: Option<Target>): Option<Impacted> =>
  (O.isSome(w) && O.isSome(t))
    ? O.some(new Impacted())
    : O.none
```

La librairie fp-ts, fournit `Option<A>`

Dans cet exemple trivial, l'apport est limité

```

import * as O from 'fp-ts/Option'
import { Option } from 'fp-ts/Option'
const calculateDamage = (w: Weapon, t: Target): number => 42
const damageMonster = (w: Option<Weapon>, t: Option<Target>): Option<number> =>
O.chain(
  (ow : Weapon) => O.map(
    (ot : Target) =>
      calculateDamage(ow, ot)
  )(t))
(w)

```

**chain** est flatMap en TS

**chain**:  $\langle A, B \rangle (f: (a: A) \Rightarrow O.Option\langle B \rangle) \Rightarrow (ma: O.Option\langle A \rangle) \Rightarrow O.Option\langle B \rangle$

**map**:  $\langle A, B \rangle (f: (a: A) \Rightarrow B) \Rightarrow (fa: O.Option\langle A \rangle) \Rightarrow O.Option\langle B \rangle$



# MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

## TRAITER LES VALEURS OPTIONNELLES EN TS

```
import * as O from 'fp-ts/Option'
import { Option } from 'fp-ts/Option'
const calculateDamage = (w: Weapon, t: Target): number => 42
const damageMonster = (w: Option<Weapon>, t: Option<Target>): Option<number> =>
O.chain(
  (ow : Weapon) => O.map(
    (ot : Target) =>
      calculateDamage(ow, ot)
  )(t)
)(w)
```

`chain` est `flatMap` en TS

`chain`: `<A, B>(f: (a: A) => O.Option<B>) => (ma: O.Option<A>) => O.Option<B>`

`map`: `<A, B>(f: (a: A) => B) => (fa: O.Option<A>) => O.Option<B>`

# TAKE AWAY

## LES OPTION / MAYBE

A utiliser pour modéliser l'absence de valeur

Sécurisant, surtout quand on dispose d'ADT

Dans certains langages Option s'appelle  
Optional (Java) ou Maybe (Scala, Haskell)

