```
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 UNE ABSENCE POTENTIELLE DE VALEUR

TRAITER LES VALEURS OPTIONNELLES EN TS

MIAGE M2 - QUALITÉ DU SI - THOMAS HAESSLÉ & QUENTIN BURG

```ts
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

```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

```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>
```

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)