Débug Angular

I.

Je viens de créer mon projet avec ng new monProjet. Je me rends dans app.ts et j'ajoute une nouvelle propriété dont j'aurais besoin. Problème : elle est soulignée et j'ai des erreurs en console lorsque je build :

app.ts

```
export class App {
  protected readonly title = signal('debug-test');

  private maPropriete: string;

  faireUnTruc() {
    this.maPropriete = "Nouvelle valeur";
  }

}
```

console

```
✗ [ERROR] TS2564: Property 'maPropriete' has no initializer and is not definitely assigned in the constructor. [plugin angular-compiler]

    src/app/app.ts:13:10:
      13 │   private maPropriete: string;
         ╵           ~~~~~~~~~~~
```

Expliquez le problème et proposez deux solutions.

On a à notre disposition trois solutions :

- *Premièrement, l'erreur nous dit que la propriété n'a pas de valeur par défaut à son initilisation. On peut donc commencer par résoudre ce problème en lui mettant simplement une valeur par défaut.*
- *Deuxièmement, on a aussi l'option de déclarer le paramètre dans la **signature** du constructeur. On parle plutôt alors d'**injection**.*
- *Troisième solution : enlever les piles du détecteur de fumée en mettant l'attribut « strictPropertyInitialization : false »*
- *Quatrième solution : Avec les caractères ? ou ! on peut dire à TypeScript d'admettre la valeur null*

II.

Toujours dans app.ts, j'ai mon tableau d'import qui est souligné en jaune bizarre :

src > app > TS app.ts > ...

```ts
import { Component, signal } from '@angular/core';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet],
  templateUrl: './app.html',
  styleUrl: './app.scss'
})
```

Expliquez le problème et proposez une solution.

*Le problème survient lorsque l'on importe des éléments qui ne sont **pas utilisés** dans le template. Il suffit alors de ne plus les importer et l'alerte s'en ira. Voici le message d'erreur qui indique le problème en console :*

```
Application bundle generation complete. [0.056 seconds]

▲ [WARNING] TS-998113: All imports are unused [plugin angular-compiler]

    src/app/app.component.ts:9:11:
      9 |    imports: [BarDirective, FooComponent],
```

III.

Je viens de corriger ma propriété dans app.ts :

```ts
private maPropriete: string = 'Hello World :) ';
```

Maintenant, je veux l'afficher dans mon template comme ceci, mais ça ne fonctionne pas :

src > app > app.html > div.container

Go to component

```html
<div class="container">
    {{ maPropriete }}
</div>
```

Pareil avec cette méthode :

src > app > app.html > div.container

Go to component

```html
<div class="container">
    {{ this.maPropriete }}
</div>
```

Expliquez le problème et proposez une solution.

*Le problème ici est la **visibilité** de la variable. En effet, elle est déclarée en **privée**, ce qui veut dire qu'elle n'est accessible que dans la classe ou elle est déclarée (ou éventuellement les classes « enfants » de cette classe).*

Rappel :

- *Visibilité **publique** : Visibilité par défaut : La propriété est visible partout, dans sa classe, mais aussi en dehors on appelant la propriété depuis un objet avec la syntaxe objet.propriete*
- *Visibilité **privée** : La propriété n'est accessible que dans sa classe avec le mot clé this.propriete. Si on souhaite l'afficher dans le template ou dans une autre classe, il faudra alors développer une méthode getPropriété() permettant d'en retourner la valeur.*
- *Visibilité **protégée** : La propriété n'est accessible que dans sa classe avec le mot clé this.propriete et ne doit pas être appelée ailleurs. Les getters / setters ne fonctionneront pas sur une propriété protégée*

IV.

```ts
export class App {
  protected readonly title = signal('debug-test');

  maPropriete: string = 'Hello World :) ';

  faireUnTruc() {
    this.maPropriete = "Nouvelle valeur";
  }
}
```

```html
<div class="container">
    {{ maPropriete }}
    <button (click)="faireUnTruc()">Faire un truc ! </button>
</div>
```

Que fait ce code dans app.ts et app.html respectivement ?

Le template prévoit un event binding sur le bouton pour déclencher la méthode « faireUnTruc() » sur l'évènement « click ». Quand la méthode est appelée, maPropriete change de valeur et il s'affiche donc sa nouvelle valeur.

V.

```ts
import { Component, signal } from '@angular/core';

@Component({
  selector: 'app-signal-basics',
  template: `
    <div>
      <h2>Compteur avec Signal</h2>
      <p>Valeur actuelle: {{ compteur() }}</p>
      <button (click)="incrementer()">Ajouter 1</button>
      <button (click)="decrementer()">Soustraire 1</button>
      <button (click)="reinitialiser()">Réinitialiser</button>
    </div>
  `
})
export class SignalBasicsComponent {
  compteur = signal(0);

  incrementer() {
    this.compteur.set(this.compteur() + 1);
  }

  decrementer() {
    this.compteur.update(valeur => valeur - 1);
  }
  reinitialiser() {
    this.compteur.set(0);
  }
}
```

Que fait ce code ?

Ce code nous permet de changer dynamiquement la valeur de la propriété « compteur », déclarée avec signal(). De l'event binding sur les trois boutons nous permet respectivement d'incrémenter, décrémenter ou de réinitilaiser la valeur de la propriété, qui se réaffichera par la même occasion puisqu'elle est déclarée avec signal()

VI.

Admettons la structure de Tache suivante :

```ts
interface Tache {
  id: number;
  titre: string;
  terminee: boolean;
  priorite: 'haute' | 'moyenne' | 'basse';
}
```

Expliquez la ligne concernant la propriété « priorite ».

*Cette ligne consiste en une **enumération** de valeurs possibles pour la propriété « priorite ». On peut parfois parler de type **implicite** (ici 'string') lorsque l'on a une liste, puisque TypeScript ne va plus vraiment vérifier les types mais plutôt les valeurs. Le type implicite de cette liste est donc « string », même si il n'est pas précisé.*

VII.

Dans mon template, je vais boucler sur mes tâches pour en faire une liste :

```
<ul>
  @for (tache of taches(); track tache.id) {
    <li [class.terminee]="tache.terminee">
      <input type="checkbox" [checked]="tache.terminee"
             (change)="toggleTerminee(tache.id)">
      <span>{{ tache.titre }}</span>
      <span class="priorite" [class]="tache.priorite">
        {{ tache.priorite }}
      </span>
      <button (click)="supprimer(tache.id)">X</button>
    </li>
  }
</ul>
```

Que fait la ligne suivante ?

```
<li [class.terminee]="tache.terminee">
```

Cette ligne fait dépendre l'attribution de la classe « terminée » de la valeur booléene de la propriété « terminee » de mon objet « tache ». Pour cela, on utilise l'attribute binding sur l'attribut « class » on lui précisant avec la syntaxe [class.classeAAjouter] quelle classe on veut conditionner.

VIII.

```
export class PanierService {
  private articles = signal<Article[]>([]);
```

Expliquez-moi ce que fait cette deuxième ligne. Expliquez-moi l'intérêt d'utiliser signal pour déclarer des propriétés.

*Cette ligne nous permet de créer une propriété privée nommée **articles**. Cette propriété va être de type Article[], c'est-à-dire un **tableau** contenant des objets de la classe **Article**. Sa valeur par défaut déclarée est **[]**, c'est-à-dire un tableau vide.*

*Puisque nous déclarons cette propriété avec **signal**, cela veut dire que les modifications apportées à cette propriété seront **diffusées**, **propagées** à tous les composants qui utilisent cette propriété, ce qui entraînera automatiquement leur mise à jour dans le template.*

IX.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-poupipou',
  template: `
    <div>
      <h2>Compteur Cassé</h2>
      <p>Valeur: {{ valeur }}</p>
      <button (click)="incrementer()">+1</button>
    </div>
  `
})
export class PoupipouComponent {
  valeur: number = 0;

  incrementer() {
    this.valeur = this.valeur + 1;
  }
}
```

Lorsque je clique sur mon bouton +1, la fonction incrementer() se lance bien mais je ne vois pas mon chiffre changer dans la balise <p>. Pourquoi ?

Si la propriété ne se met pas à jour dans le template après une modification de valeur, alors c'est qu'il faut la déclarer avec signal()

X.

Voici ma classe qui gère mon panier. Je veux pouvoir boucler sur tous les produits pour en afficher le montant total

```ts
export class MonPanierProduit {
  produits = signal<Produit[]>([
    { id: 1, nom: 'Laptop', prix: 1200 },
    { id: 2, nom: 'Souris', prix: 25 },
    { id: 3, nom: 'Clavier', prix: 80 }
  ]);

  augmenterPrix(id: number) {
    this.produits.update(items =>
      items.map(p => p.id === id ? { ...p, prix: p.prix + 1 } : p)
    );
  }
}
```

Dans mon template, j'utilise ceci :

```html
<ul>
  @for (produit of produits()) {
    <li>
      {{ produit.nom }} - {{ produit.prix }}€
      <button (click)="augmenterPrix(produit.id)">+1€</button>
    </li>
  }
</ul>
```

Pourquoi mon @for est souligné ?

Il manque la syntaxe track, qu'on pourrait noter comme ceci :

@for(produit of produits() ; track produit.id)

La syntaxe track doit admettre un identifiant unique par itération. En cas de modification d'un élément du tableau, cela permet à Angular de ne recharger que l'élément qui a changé sans devoir réexécuter toute la boucle.