XEBIA NATIS

Angular

Michael

XEBIA NATIS

XEBIA NATIS

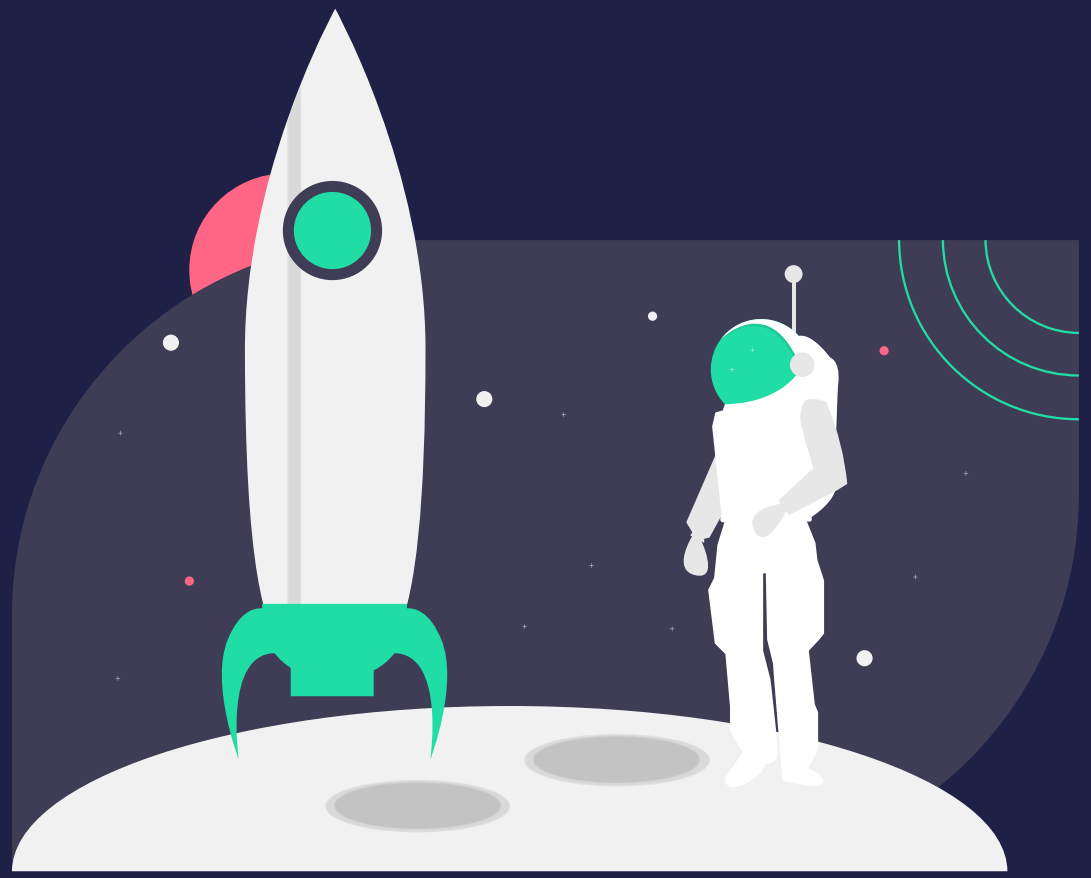
Compétence visée :

Comprendre l'articulation de TypeScript pour le développement front-end avec Angular

XEBIA NATIS

1. Data bindings
2. Control flow
3. Inversion of Control
4. L'injection de dépendances

XWIKI NATIS



XWIKI SAS

XNATIS

DATA BINDINGS

XEBIANATIS

Les symboles {{ }} permettent de d'effectuer un text interpolation pour binder la valeur d'une variable à cette endroit. Si la valeur de la variable change, la vue changera aussi.

```
<span class="fistname">{{ firstname }}</span>
```

XEBIANATIS

[(ngModel)] permet d'effectuer un two-way binding, donc une synchronisation parfaite entre la variable bindée et la vue

```
<input type="text" [(ngModel)]="firstname" />
```

XWIKI SAS

XNATIS

CONTROL FLOW

XEBIA NATIS

*ngIf vous permet de conditionner la création de l'élément dans le DOM en vous basant sur un prédicat

```
<div *ngIf="age >= 18">Vous êtes majeur(e)</div>
<div *ngIf="age < 18">Vous êtes mineur(e)</div>
```

XEBIANATIS

*ngFor vous permet de dupliquer l'élément en vous basant sur un tableau

```
<div *ngFor="let entry of entries">{{ entry }}</div>
```

XWIKI SAS

XEBIA NATIS

INVERSION OF CONTROL

XKNATIS

Une architecture logicielle avec cette conception inverse le contrôle par rapport à la programmation procédurale traditionnelle :

1. Dans la programmation traditionnelle, c’est le code qui fait appel à des objets pour prendre en charge les tâches
2. Avec l'inversion du contrôle, c'est le système qui appelle ces objets

XWIKI NATIS

L'inversion de contrôle est utilisée pour augmenter la modularité du programme et le rendre extensible.

XWIKI SAS

XKNATIS

INJECTION DE DEPENDANCES

XEBIA NATIS

L'injection de dépendances (dependency injection en anglais) est un mécanisme qui permet d'implémenter le principe de l'inversion de contrôle.

XWIKI NATIS

Il consiste à créer dynamiquement (injecter) les dépendances entre les différents objets en s'appuyant sur une description (fichier de configuration ou métadonnées) ou de manière programmatique.

XKNATIS

Ainsi les dépendances entre composants logiciels ne sont plus exprimées dans le code de manière statique mais déterminées dynamiquement à l'exécution.

XEBIA NATIS

apiService: ApiService
provider: Provider

CONTAINER

class ….

constructor(private apiService: ApiService)

class ….

constructor(private provider: Provider)

XEBIANATIS

apiService: ApiService
provider: Provider

CONTAINER

class ….

constructor(private apiService: ApiService)

class ….

constructor(private provider: Provider)

XPNATIS

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import Entity from '../models/Entity';

@Injectable({
    providedIn: 'root'
})
export class ApiService {

    constructor(private http: HttpClient) {

    }

    public list() {
        return this.http.get<Entity[]>('http://localhost:4000');
    }

    public search(text: string) {
        return this.http.post<Entity[]>('http://localhost:4000', {text: text});
    }
}
```

XWIKI SAS

XWIKI NATIS

Pour aller plus loin

https://en.wikipedia.org/wiki/Dependency_injection