TD Formation Angular Débutant

# TD 1. Exécution d'un code typescript

Dans ce TD, vous allez être capable de récupérer un projet depuis un dépôt git, ensuite tranpiller du code typescript et rajouter une fonctionnalité.

1. Installez nodejs sur votre ordinateur

https://nodejs.org/en/download/current

2. Récupérez le code depuis le repos gitlab

**git clone git@gitlab.com:formation-angular-d-butant/mini-increment.git**

3. Ouvrez le dépôt récupéré avec VS Code

Analysez la structure du projet.
Exécutez la commande **npm install** pour installer les paquets

4. Lancez le projet

Ouvrez le fichier **index.html** via votre navigateur. Que se passe-t-il?

5. Transpillez le code typescript

Pour transpiler le code, lancez la commande
**tsc**
Rechargez votre page, que se passe-t-il?

PS: si vous rajoutez **–watch** à **tsc** il va surveiller les modifications des fichiers.

Pour en savoir plus sur les options de compilation:
https://www.typescriptlang.org/docs/handbook/compiler-options.html

6. Ajoutez une nouvelle fonctionnalité

Vous allez rajouter, un code qui permet de décrémenter la valeur affichée lorsque l'on clique sur le bouton [-]

# TD 2. Manipulation des composants

### T2.1. Affichage d'une liste d'éléments

Dans ce TD, vous allez être capable d'afficher une liste d'éléments en utilisant une directive structurelle qui va boucler sur un composant enfant afin d'avoir des informations.

1. Récupérez les sources depuis le repo

**git clone git@gitlab.com:formation-angular-d-butant/agent-app.git**

2. Ouvrez les sources avec VS Code

3. Créez un composant **header** et déplacez partiellement le code de la vue du composant **AppComponent** dans le composant header (ainsi que le style ayant **.toolbar** comme nom de classe dans le fichier de style de header)

```
<!-- Toolbar -->
<div class="toolbar" role="banner">
  <img
    width="40"
    alt="Angular Logo"
    src="
  />
  <span>Formation Angular</span>
</div>
<!-- end Toolbar -->
```

4. Insérez le composant **header** dans la vue de **AppComment** et visualiser.

5. Affichez la liste des éléments (**agentList**) du composant **AppComponent** dans la vue

- ***firstname lastname : age***

6. Créez un composant **agent** qui va prendre en paramètre un agent et l'afficher

7. Intégrez ce composant dans **AppComponent**

8. Créez une directive **adultBold** qui va mettre en gras les valeurs qui sont supérieures ou égales à 18.

```
// récupérer le contenu(valeur) de l'élément HTML
this.el.nativeElement.innerText

// modification de la police
this.el.nativeElement.style.fontWeight = 'bold';
```

9. Insérez le sélecteur de cette directive sur l'élément **span** de votre composant agent

### T2.2. Communication entre composant

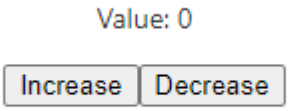
Dans ce TD, vous allez être capable de mettre en place une communication entre composants afin d'afficher la valeur d'un nombre et pouvoir l'augmenter ou le baisser.

1. Créez un nouveau projet angular (vous donnez un nom au choix)

2. Lancez l'application générée.

3. Nettoyez le code générer par angular par défaut et laisser juste le header (bleu)

4. Dans la vue du composant principale **AppComment**, vous allez réaliser la vue suivante:




5. Vous allez créer 2 composants:

- un composant **display**

Qui va afficher le message de la valeur actuelle

- un composant **button**

Qui va représenter un bouton

6. Vous allez mettre à jour la vue du composant **AppComponent** avec vos nouveaux composants: une fois **display** et 2 fois **button** et visualisez.

7. Vous allez créer une variable dans **displayedValue** valant 0 dans votre composant **AppComponent**. Puis la transmettre au composant **display** pour l'afficher

8. Vous allez ajouter des événements dans **button** qui écoute le click et renvoyer au composant **AppComponent**

9. Dans le composant **AppComponent** vous allez écrire des fonctions (increase, decrease) qui sont se brancher sur ces événements afin d' incrémenter la valeur si le bouton concerné est une augmentation et baisser la valeur dans le cas contraire.

10. Ajoutez maintenant un moyen de protéger les clics, par exemple on ne peut pas cliquer sur un bouton pour augmenter la valeur si celle-ci est 20 et 0 dans le cas ou on baisse la valeur.

# TD 3. Création d'une application permettant d'afficher des posts

Dans ce TD, vous allez être capable de créer une application frontend qui se connecte à une API (**JSONPlaceholder**) ou serveur distant afin de manipuler des données.

PS. Dans ce TD, utilisez le bootstrap pour le design du site.
PS. Tous vos appels vers l'api doivent être écrits dans un service que vos composants vont utiliser.

1. Créez un nouveau projet Angular (nom au choix)

2. Nettoyez l'application créée (interface par défaut de Angular)

3. Créez une page qui affiche la liste des **posts**

Vous allez utiliser le endpoint**(*)** ci-dessous pour récupérer les posts depuis l'api.
Method: GET
Url: https://jsonplaceholder.typicode.com/posts

Cette api retourne un tableau de posts.

```
interface post {
    userId: number,
    id: number,
    title: string,
    body: string
}
```

Cette page doit:
- Avoir un titre de la page
- Afficher la liste des posts (affichage sous forme de liste ou card à votre choix): title et body
- Les posts doivent avoir un lien pour afficher les détails du posts (url à ajouter dans un second temps)

4. Créez une page de détails d'un **post**

Vous allez utiliser le endpoint**(*)** ci-dessous pour récupérer les détails d'un post depuis l'api.
Vous devez passer l'id du post donc vous souhaitez récupérer les données.
Method: GET
Url: https://jsonplaceholder.typicode.com/posts/1

Cette api retourne un post.

```
interface post {
    userId: number,
    id: number,
```

```
    title: string,
    body: string
}
```

Cette page doit:

- Avoir un titre de la page
- Afficher les informations du post
- Avoir un lien permettant de revenir sur la page des posts

5. Ajout de l'affichage **commentaires** d'un post

Vous allez ajouter dans la page de détails d'un post, la liste de commentaires associée à ce post.

Créer un composant qui va afficher la liste des commentaires grâce à un **input** qu'il va recevoir. Ce composant sera appelé depuis le composant détails d'un post.

Vous allez utiliser le endpoint**(*)** ci-dessous pour récupérer les commentaires d'un post depuis l'api. Vous devez passer l'id du post donc vous souhaitez récupérer les commentaires
Method: GET
Url: https://jsonplaceholder.typicode.com/posts/1/comments

Cette api retourne les commentaires d'un post.

```
interface comment{
    postId: number,
    id: number,
    name: string,
    email: string,
    body: string
}
```

Cette liste de commentaire doit:

- Afficher l'email de de la personne qui a commenté
- Afficher le commentaire

6. Créez la page pour la création d'un post

Vous allez créer une page qui permet de créer un  post.

Vous allez envoyer les données du formulaire sur cette api via le endpoint **(*)** ci-dessous.
Method: POST
Url: https://jsonplaceholder.typicode.com/posts
payload: {
  title: “my title”,
  body: “my body”,
  userId: 1
}

PS. le payload dans une requêtes HTTP c’est le contenu (l’objet ayant des données) de la requête

- Créez une page de création d’un post.
- Ajoutez un lien sur la page de liste des post permettant d’arriver sur la page de création.
- Sur la page de création d’un post , créez un formulaire (TDF ou DDF) ayant les champs suivants: title, body, userId (ayant la valeur par défaut 1). Tous les champs sont obligatoires.

**(*)** Sur une API, **un endpoint** correspond à l'extrémité d'un canal de communication. En pratique, il s'agit du point par lequel une API entre en contact avec d'autres systèmes avec lesquels elles communiquent. Un endpoint peut, par exemple, contenir l'URL d'un serveur auquel l'API envoie ses requêtes.