Composants

Le premier composant que l'on va voir est le composant "root" C'est à dire le composant racine, celui qui sera à la base de votre application.

Si on ouvre le fichier "**app.component.html**" On verra que Angular nous a créer un template par défaut. On peut tout supprimer sauf :

```
<router-outlet/>
```

et on va ajouter un petit h1 :

```
<h1>Bienvenue sur {{title}}!</h1>
```

histoire d'avoir quelque chose dans notre page.

Ensuite ouvrons et examinons notre fichier "**app.component.ts**":

- On commence avec un import de "**Component**" depuis le noyau d'Angular.
- On trouvera aussi un import du "**RouterOutlet**"
- Puis on utilise ce **Component** en tant que **décorateur** Dans ce décorateur on va trouver au moins 2 options:
  - **selector** nous permettra de donner un nom à notre composant.
  - **standalone** indique que le composant ne dépend d'aucun module
  - **imports** indique des modules qui sont utilisées ici
  - **templateUrl** indique où trouver le html de notre composant.
  - **styleUrl** est optionnel et indique où trouver le style de notre composant.
- Enfin on aura une classe "AppComponent" où une propriété title est défini.

(sur un tout petit composant il est possible d'utiliser l'option "**template**" avec juste un string.)

```
template: `<h1>Titre</h1>`
```

Un composant peut être donc défini par "une classe" et "un template".

- il représente un morceau de votre site. Pas forcément une page entière.
- Il va échanger tout un tas d'information entre sa classe et son template.
- Un composant à un cycle de vie représenté par plusieurs méthodes :
  - **ngOnChanges** est appelé à chaque modification de notre composant, dont sa création.
  - **ngOnInit** est appelé juste après la création de notre composant.
  - **ngDoCheck** est appelé pour étendre les détections par défaut de ngOnChanges.
  - **ngAfterViewInit** est appelé juste après que le template soit initialisé.

- **ngOnDestroy** est appelé juste avant qu'un composant soit détruit et retiré du DOM. (Ce dernier pourra par exemple servir à supprimer des évènements ou des services qui n'ont plus lieu d'être sans ce composant, pour ne pas saturer la mémoire du navigateur)

le plus souvent ce sera "**ngOnInit**" et "**ngOnDestroy**" qui seront utilisé. à noté que Angular appelle ces méthodes peu importe que vous les utilisiez ou non.

Revenons à notre "**app.component.ts**" et ajoutons en propriété un tableau qui sera une liste de recette dans notre classe :

```
receteList = ["Carbonade", "Okonomiyaki", "Cannelé"];
```

Puis affichons une recette dans notre template :

```
<h2>{{receteList[1]}}</h2>
```

Ensuite on va ajouter à notre import depuis le noyau, l'interface "onInit":

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

Puis on l'implémente à notre classe :

```
export class AppComponent implements OnInit
```

Et pour que notre classe soit valide, on y ajoute la méthode demandé par l'interface :

```
ngOnInit(): void
{
    // Pour le tester on affichera un message dans la console.
    console.table(this.receteList);
}
```

Petite remarque, pourquoi utiliser "**ngOnInit**" plutôt que le "**constructor**" de la classe ?

Principalement car la classe est construite bien avant son initialisation. Angular va préparer la classe avant de lancer tout son fonctionnement.

Ajoutons une nouvelle méthode à notre classe :

```
selectRecette(nom: string): void
{
```

```
    console.log(`Vous avez sélectionné ${nom}`);
}
```

On pourra la tester dans le "**ngOnInit**"

```
this.selectRecette(this.recetteList[0])
```

Ajoutons à notre dossier "**app**" les fichiers "**Recette.ts**" et "**RecetteList.ts**";

Exercice 1

Consigne dans le fichier "**exercice-1.md**"

Correction

Importer la liste des recettes dans notre composant racine.

```
import { RECETTES } from './RecetteList';
```

Donner cette liste à notre propriété "recetteList" et la typer pour être sûr que notre propriété corresponde.

```
recetteList: Recette[] = RECETTES;
```

Modifier notre méthode "selectRecette" pour indiquer que le type de notre argument sera une "Recette".

```
import { Recette } from './Recette';
// ...
selectRecette(recette: Recette): void
{
console.log(`Vous avez sélectionné ${recette.name}`);
}
```

Corrigez les possibles erreurs typescript. Changer le template pour qu'il affiche bien le nom de la recette:

```
<h2>{{recetteList[0].name}}</h2>
```