Les Pipes

Les pipes permettent d'améliorer la mise en forme des éléments que l'on affiche par interpolation. Cela peut être particulièrement utile pour traiter des données brute comme l'exemple le plus courant, celui des dates. Actuellement, les dates sur nos recettes ne sont pas très jolie à lire. On va de ce fait utiliser un Pipe présent nativement en Angular, le pipe "**date**" Pour cela on va placer l'opérateur de pipe à droite de la propriété à modifier "**|**", puis le pipe que l'on souhaite utiliser.

Dans "**app.component.html**" :

```
{{rec.createdAt | date}}
```

On a maintenant à la place du gros texte indigeste une date sous la forme "**Jul 31, 2022**" Il existe d'autres pipe présent nativement que l'on peut retrouver dans le documentation.

- https://angular.io/api?type=pipe

Il est possible d'utiliser plusieurs pipes en même temps, utilisons par exemple "**uppercase**"

```
{{rec.createdAt | date | uppercase}}
```

Maintenant ma date est bien en majuscule. mais supprimons uppercase qui nous est inutile ici. Certains pipes peuvent avoir des options.

```
{{rec.createdAt | date:"dd/MM/yyyy"}}
```

Pour voir en détail les options des différents pipes, je vous invite à lire la documentation.

Créer un Pipe

On peut évidement créer ses propres pipes, pour cela on va encore une fois faire attention à ce que notre terminal soit dans le bon dossier puis taper :

```
ng generate pipe type-color
# type-color étant le nom que j'ai donné à mon nouveau pipe
```

Comme pour les directives, il a créé le fichier pour notre pipe, (plus le spec si necessaire) et modifié "**app.module.ts**" pour y inclure notre pipe.

Encore une fois on y retrouvera :

- des imports depuis le noyau

- un décorateur pour nommer notre pipe ainsi qu'indiquer le standalone
- Une classe qui cette fois implémente une interface.

Et ce qui nous intéresse une méthode **transform** pré-faite, on ne la gardera pas comme telle.

```
transform(type: string): string {}
/*
    le premier argument correspondra à l'élément à gauche du pipe "|"
    Si on souhaite ajouter des argumants à droite de notre pipe, ce seront ceux à
partir du second argument.
*/
```

Ensuite nous allons faire un switch sur les différents cas possible et leur attribuer une couleur:

```
let color:string="";
switch(type){
    case "dessert":
        color = "pink";
        break;
    case "plat":
        color = "brown";
        break;
    case "entrée":
        color = "green";
        break;
}
return color;
```

Il ne nous reste plus qu'à importer notre pipe dans notre component:

```
imports: [otherImports, TypeColorPipe]
```

Ainsi qu'à utiliser ce pipe dans notre html :

```
<!-- Sur une directive : -->
<div class="recette" *ngFor="let rec of recetteList" appBorderCard="
{{rec.type|typeColor}}">
<!-- Et en style : -->
<span class="type" [style.backgroundColor]="rec.type|typeColor">{{rec.type}}
</span>
```

Ici il nous rend simplement une couleur mais on aurait pu l'adapter pour une classe, modifier du texte ou même un objet.