9. Pipes

Introduction

Les `Pipes` dans le framework Angular permettent d'automatiser la formatage de données. Pour cela il est nécessaire d'utiliser une convention de délaration dans la template HTML, afin de définir le `Pipe` qui sera lancé automatiquement lorsqu'une donnée sera à formater.

Pipes Angular

Présentation

Certains Pipes existe nativement dans Angular, et permettent d'éffectuer des formatages simples. Il est nécessaire d'importer le module `CommonModule` afin que ces pipes soit reconnus dans les templates HTML.

Dans cet exemple, la propriété `connexionDate` va être formatée par le pipe `date` avec en paramètre le type de formatage `fullDate` .

```
<p>Date de connexion {{ user.connexionDate | date:'fullDate' }}</p>
```

`fullDate` peut être remplacé par de nombreuse expressions.

Les pipes peuvent prendre plusieurs paramètres en entrées, ils doivent être séparés par un `:` . Par exemple, nous définissons que le format de la date doit être au format français.

```
{{ user.connexionDate | date:'fullDate':'':'fr-FR' }}
```

Chaîner des Pipes

Il est possible de chaîner les Pipes afin d'appliquer plusieurs formatages de données à la suite.

Dans cet exemple, la propriété `connexionDate` va être formatée par le pipe `date` avec en paramètre le formatage `fullDate` , puis sera affichée en majuscule avec le Pipe `uppercase` .

```
{{ user.connexionDate | date:'fullDate' | uppercase}}
```

Liste des Pipes disponibles

- `async` - AsyncPipe : Associe dans la template la résolution d'un Observable ou d'une Promesse,
- `currency` - CurrencyPipe : Mise en forme d'une valeur monétaire,
- `date` - DatePipe : Mise en forme d'une date,
- `decimal` - DecimalPipe : Mise en forme d'une valeur décimale,
- `i18n-plural` - I18nPluralPipe : Mise en forme d'un texte en fonction du nombre de données,
- `json` - JsonPipe : Mise en forme d'un objet sous forme d'une sérialisation JSON,
- `lowercase` - LowerCasePipe : Mise en forme d'un texte en minuscule,
- `uppercase` - UpperCasePipe : Mise en forme d'un texte en majuscule,
- `percent` - PercentPipe : Mise en forme d'un nombre en pourcentage,
- `slice` - SlicePipe : Slice un tableau ou une chaîne de caractère avec un index début et fin paramétré,
- `titlecase` : TitleCasePipe : Mise en forme d'un texte avec chaque première lettre des mots en majuscule.

Documentation officielle

Pipes personnalisées

Présentation

Il est possible de créer des pipes personnalisées afin de mettre en oeuvre un formatage spécifique. Pour cela on utilisera le décorateur `@Pipe` , et l'on implémentera la méthode `transform` qui prend en paramètre la donnée à formater, ainsi que des paramètres optionnels, fonction du besoin.

```
@Pipe({
  name: "monpipe",
})
class MonPipe {
  transform(
    value: string,
    param1: string = null
```

```
    param1: string = null,
    params2: string = null
  ): string {
    return "valeur-format";
  }
}
```

Il suffit ensuite, dans la template HTML, d'utiliser l'annotation `monpipe` pour exploiter le `Pipe` customisé.

```
<p>{{valeurAFormater | monpipe:'params1':'params2'}}</p>
```

NB : le nom du `selector` pour un pipe n'authorise pas l'utilisation du caractère `-` .

Exemple

Dans cet exemple, nous allons définir une URL par défaut sur une image si la source en entrée est null ou vide. Il faut tout d'abord définir la classe avec le décorateur `@Pipe` , permettant d'afficher une image par défaut dans le cas où celle en entrée est vide ou nulle.

Le premier paramètre de la méthode `transform` du `Pipe` correspond à l'url de l'image, le 2ème correspond à l'url de remplacement en cas de source nulle.

```
import { Pipe } from "@angular/core";

@Pipe({
  name: "defaultimage",
})
export class ImageDefaultPipe {
  transform(src: string, defaultUrl: string): string {
    let imageSrc: string = "";
    if (src && src != "") {
      imageSrc = src;
    } else {
```

```
      imageSrc = defaultUrl;
    }

    return imageSrc;
  }
}
```

Dans le module, il est nécessaire de déclarer le `Pipe` dans les `declarations` .

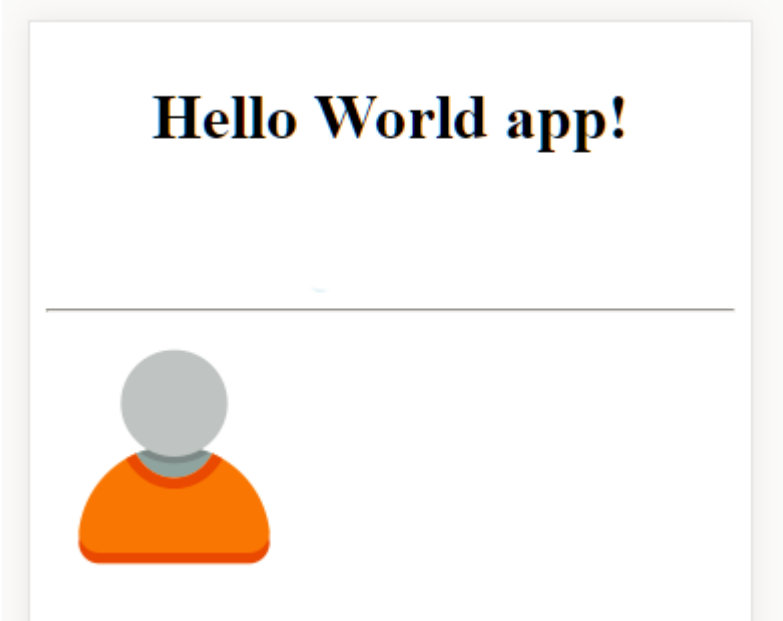
```
@NgModule({
  bootstrap: [],
  declarations: [UnComponent, ImageDefaultPipe],
  exports: [UnComponent],
  imports: [CommonModule, routing],
  providers: [],
})
export class ExemplePipeModule {}
```

Une fois le `pipe` déclaré dans le module, celui-ci devient exploitable dans une template HTML. Il suffit d'exploiter le nom du pipe `defaultimage` et de définir l'image par défaut si la source `user.thumbnail` est vide.

```
<img [src]="user.thumbnail | defaultimage:'assets/images/user-default.png'" />
```

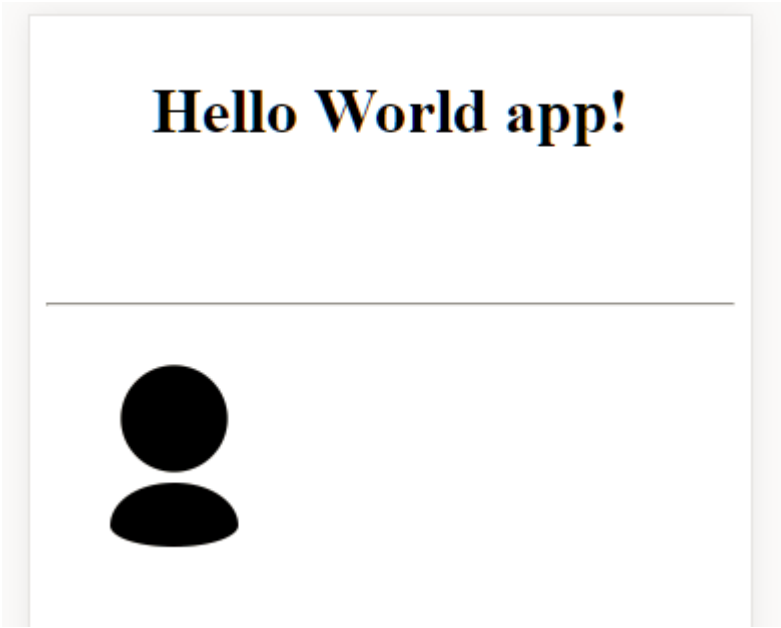
Dans le cas où l'utilisateur à un thumbnail défini, l'affichage est le suivant.

```
public user: any = {
  thumbnail: "assets/images/user-a.png"
};
```



Dans le cas où l'utilisateur à un thumbnail vide, l'affichage est le suivant.

```
public user: any = {
  thumbnail: ""
};
```



Excercice

Présentation

Application Météo.

L'objectif de cet excercice est d'afficher la date et la température de la météo au travers d'un pipe.

Etapes de création d'utilisation et de création d'un pipe

1. Exploiter le DatePipe Angular pour formater la date de la météo. Ce pipe est disponible au travers de `CommonModule` , vous ne devriez pas avoir besoin de faire d'import.
2. Créer un pipe pour formater les degrés affichés. Affichant devant "+" si température positive ou égale à 0, - si température négative.

3. Ajouter l'unité Celcius, et rajouter l'unité Farhenheit. Ex : `+13.58 ° C (56 ° F)`

Bravo, vous utiliser les pipes de Angular, et vous avez créé un Pipe personnalisé !

Correction disponible ici