Par NIDHAL JELASSI
nidhal.jelassi@fsegt.utm.tn

PROGRAMMATION WEB AVANCÉE

ANGULAR

Chapitre 4 : Les directives
Supplément : Les Pipes

OBJECTIFS

- Comprendre la définition et l'intérêt des directives.
- Voir quelques directives d'attributs offertes par Angular et savoir les utiliser
- Créer votre propre directive d'attributs
- Voir quelques directives structurelles offertes par Angular et savoir les utiliser
- Définir les pipes et l'intérêt de les utiliser
- Vue globale des pipes prédéfinies
- Créer un pipe personnalisé

QU'EST CE QU'UNE DIRECTIVE ?

- Jusqu'ici, nous avons déjà utilisé des directives… sans vraiment nous en rendre compte.
- Une directive est une classe comme le composant sauf qu'elle n'a pas de template.
- C'est une classe qui permet d'attacher un comportement spécifique aux éléments du DOM. Elle est décorée avec l'annotation **@Directive**.
- Apparait dans un élément comme un tag (comme le font les attributs).
- La command pour créer une directive est : **ng g d nomDirective**

QU'EST CE QU'UNE DIRECTIVE ?

- Il est possible d'avoir plusieurs directives appliquées à un même élément.
- Une directive possède un sélecteur CSS qui indique au framework où l'activer dans notre template.
- Lorsque Angular trouve une directive dans notre template HTML, il instancie la classe de la directive correspondante et donne à cette instance le contrôle sur cet élément du DOM.

TYPES DE DIRECTIVES

Angular distingue trois types de directives :

- Les composants qui sont des directives avec des templates. D'ailleurs la classe **Component** *hérite* de la classe **Directive** dans Angular.
- Les directives structurelles qui changent l'apparence du DOM en ajoutant, manipulant et supprimant des éléments. Les directives **ngFor** et **ngIf** déjà vus font partie de cette catégorie.
- Les directives d'attributs (attribute directives) qui permettent de changer l'apparence ou le comportement d'un élément.

1

Les directives structurelles

LES DIRECTIVES STRUCTURELLES

- Comme déjà dit, les directives structurelles d'Angular sont les directives qui manipulent les éléments du DOM, reconnaissables au fait qu'elles commencent toutes par un asterisque (*), comme par exemple ***ngIf** et ***ngFor**.

- Elles sont appliquées sur **l'élément HOST** (l'élément sur lequel la directive est appliquée).

- Alors que l'*interpolation* et le *property binding* permettent de modifier l'affichage et le contenu, ils ne permettent pas de modifier la structure du DOM en ajoutant ou en retirant des éléments par exemple

LES DIRECTIVES STRUCTURELLES

- Pour remédier à cette limitation, Angular fournit des directives structurelles qui permettent de modifier la structure du DOM.

- Les directives les plus connues sont :
  - *ngIf
  - *ngFor
  - [ngSwitch], *ngSwitchCase and *ngSwitchDefault

ngFor

- La directive structurelle **ngFor** permet de boucler sur un array et d'injecter les éléments dans le DOM.

```
<ol>
    <li *ngFor="let personne of listePersonne"> {{personne.nom}} {{personne.age}}</li>
</ol>
```

Bienvenue dans l'application de gestion de CV

1. Nidhal 35
2. Yasmine 8
3. Senda 3

ngFor

- **NgFor** permet de répéter un élément plusieurs fois dans le DOM.
- Prend en paramètre les entités à reproduire.
- Fournit certaines valeurs :
  - *index*: position de l'élément.
  - *odd*: true si l'élément est à une position impaire.
  - *even*: true si l'élément est à une position paire.
  - *first*: true si l'élément est à la première position.
  - *last*: true si l'élément est à la dernière position.

ngFor

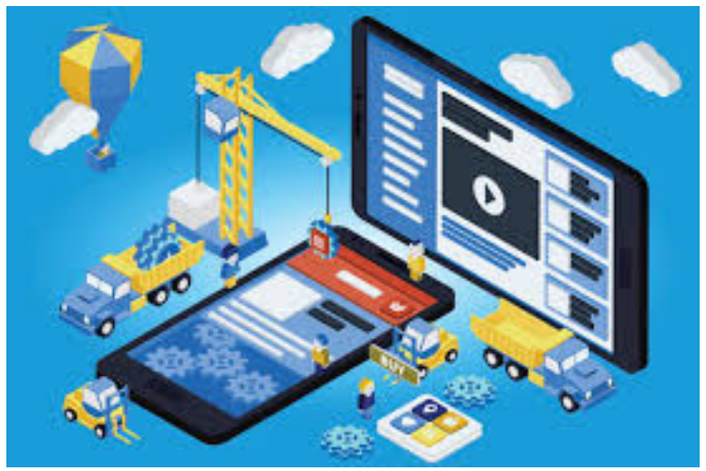
- **Exemple :**

```
<ol>
    <li *ngFor="let personne of listePersonne; let isFirst = first; let isLast = last;
    let isEven = even; let isOdd = odd"> {{personne.nom}} {{personne.age}} -
First : {{isFirst}} - Last : {{isLast}} - Pair : {{isEven}} - Impair : {{isOdd}}</li>
</ol>
```

Bienvenue dans l'application de gestion de CV

1. Nidhal 35 - First : true - Last : false - Pair : true - Impair : false
2. Yasmine 8 - First : false - Last : false - Pair : false - Impair : true
3. Senda 3 - First : false - Last : true - Pair : true - Impair : false

ngFor : Application



- Créez un composant contenant un tableau d’objets personnes.
- Chaque personne est caractérisée par son nom, prénom, âge et métier.
- Utiliser *ngFor pour afficher la liste de ces personnes.

ngIf

- **ngIF** est associé à une expression booléenne. Si cette expression est fausse (false) alors l'élément et son contenu sont retirés du DOM, ou jamais ajoutés).
  - Si le booléen est true alors l'élément host est visible.
  - Si le booléen est false alors l'élément host est caché.

```
<p *ngIf="pers">
    La personne sélectionnée existe !!!
</p>
```

ngIf : Application

- Créez un composant contenant un bouton et un paragraphe.
- Le bouton s'appellera « Afficher / Cacher ».
- Un paragraphe contenant une phrase.
- Au click, si le paragraphe est caché on l'affiche, s'il est affiché, on le cache.

ngSwitch

- La directive **ng-switch** est une structure conditionnelle de type switch qui s'utilise directement dans le template. Elle permet d'afficher un élément donné selon la valeur de l'expression qu'elle évalue.

```
export class CvComponent implements OnInit {

  person : string;
```

```
<ul [ngSwitch]="person">
    <li *ngSwitchCase="'Nidhal'">Hello Nidhal</li>
    <li *ngSwitchCase="'Yasmine'">Hello Yasmine</li>
    <li *ngSwitchCase="'Senda'">Hello Senda</li>
    <li *ngSwitchDefault>Aucune personne n'est ajouté</li>
  </ul>
```

2

Les directives d’attributs

NgStyle

- Cette directive permet de modifier l'apparence de l'élément cible.
- Lorsque l'on utilise cette directive, il est nécessaire de la placer entre crochets comme ceci : [ngStyle] . Ce n'est pas forcément le cas de tous les attributs directives.
- Elle prend en paramètre un attribut représentant un objet décrivant le style à appliquer.
- Elle utilise le property Binding.

NgStyle : Application

Créez un mini-simulateur de Microsoft Word pour gérer un paragraphe en utilisant ngStyle.

- Préparer un input de type *texte*, un input de type *number*, et un *selectbox*.
- Faite en sorte que lorsqu'on écrive une couleur dans le texte input, ca devienne la couleur du paragraphe. Et que lorsque on change le nombre dans le number input, la taille de l'écriture change également.
- Finalement ajouter une liste et mettez y un ensemble de police. Lorsque le user sélectionne une police dans la liste, la police dans le paragraphe change.

NgClass

- Cette directive permet de modifier l'attribut class. Permet de choisir facilement parmi une liste de plusieurs classes possibles qui vous permettent de spécifier une liste d'objets incluant des conditions. Angular est capable d'utiliser la classe correcte en fonction de la véracité des conditions.
- Votre objet doit contenir des paires clé / valeur. La clé est un nom de classe qui sera appliqué lorsque la valeur (conditionnelle) est évaluée à true.
- Elle prend en paramètre
  - Une chaine (string)
  - Un tableau (dans ce cas il faut ajouter les [ ] donc [ngClass])
  - Un objet (dans ce cas il faut ajouter les [ ] donc [ngClass])
- Elle utilise le property Binding.

NgClass : Exemple

```css
.style-1 {
    color : green;
    background-color: rgba(100, 100, 224, 0.596);
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, U
}

.style-2 {
    color : blue;
    background-color : rgb(217, 255, 127);
    font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}

.style-3 {
    color : black;
    background-color: greenyellow;
    font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida S
}
```

CSS

```html
<h1 [ngClass]="{
        'style-1' : true,
        'style-2' : false,
        'style-3' : false
    }">
    Bienvenue dans l'application de gestion de CV
</h1>
```

HTML

APPLICATION

- Ajouter un div et un bouton qui à chaque click permet de changer de style. On aura 3 styles différents déclarés dans le fichier CSS du composant.

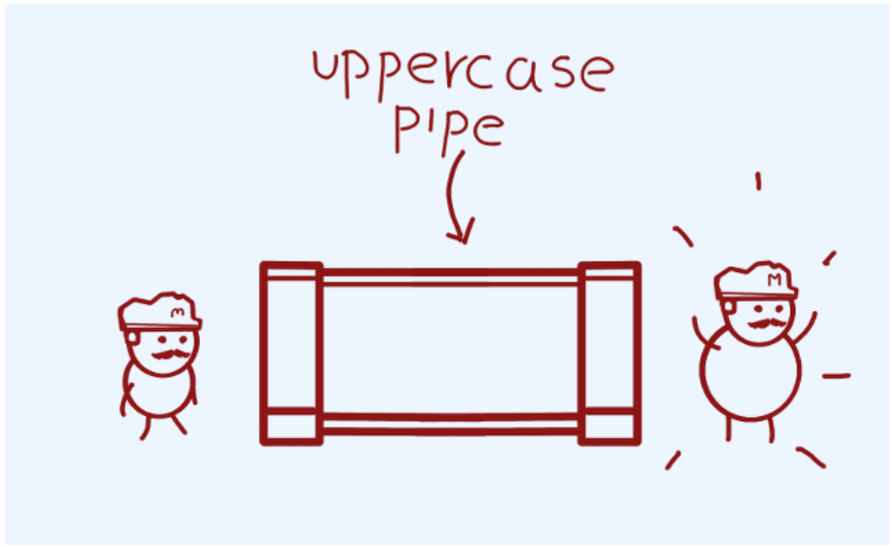
CREER SA PROPRE DIRECTIVE

- Afin de créer sa propre « attribut directive » il faut utiliser un HostBinding sur la propriété que vous voulez binder.
- Exemple : @HostBinding('style.backgroundColor') bg:string="green";
- Si on veut associer un **événement** à notre directive on utilise un HostListener qu'on associe à un **événement** déclenchant une méthode.
- Exemple : @HostListener('mouseenter') mouseenter(){
  this.bg = "blue";
  }
- Afin d'utiliser le **HostBinding** et le **HostListener** il faut les importer du core d'angular

LES PIPES

PRÉSENTATION D'UN PIPE

- Un pipe est une fonctionnalité qui permet de formater et de transformer vos données avant de les afficher dans vos Templates.
- Exemple : l'affichage d'une date selon un certain format, Mettre une chaîne de caractères en majuscules, etc.
- Il existe des pipes offerts par Angular et prêt à l'emploi.
- Vous pouvez créer vos propres pipes personnalisés.



SYNTAXE

- Afin d'utiliser un pipe vous utilisez la syntaxe suivante :

{{ **variable** | **nomDuPipe** }}

- *Exemple* : {{ **maDate** | **date** }}

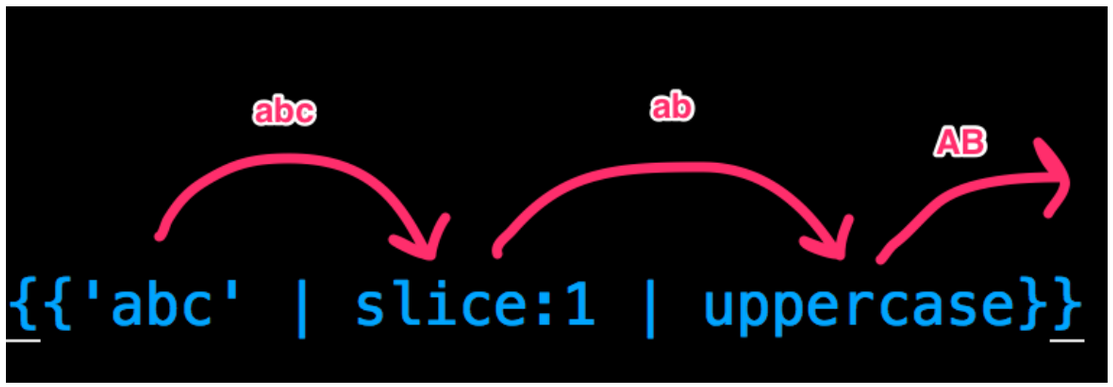
- Afin d'utiliser plusieurs pipes combinés vous utilisez la syntaxe suivante :

{{ **variable** | **nomDuPipe1** | **nomDuPipe2** | **nomDuPipe3** }}

- *Exemple* : {{ **maDate** | **date** | **uppercase** }}

LES PIPES DISPONIBLES PAR DÉFAUT (BUILT-IN PIPES)

- La documentation d'angular vous offre la liste des pipes prêt à l'emploi (https://angular.io/api?type=pipe).
  - uppercase
  - lowercase
  - titlecase
  - currency
  - date
  - json
  - percent
  - etc..



PARAMÉTRER UN PIPE

- Afin de paramétrer les pipes ajouter ‘**:**’ après le pipe suivi de votre paramètre.

{{ **maDate** | **date:"MM/dd/yy"** }}

- Si vous avez plusieurs paramètres c’est une suite de ‘**:**’

{{ **nom** | **slice:1:4** }}

PIPE PERSONNALISÉ

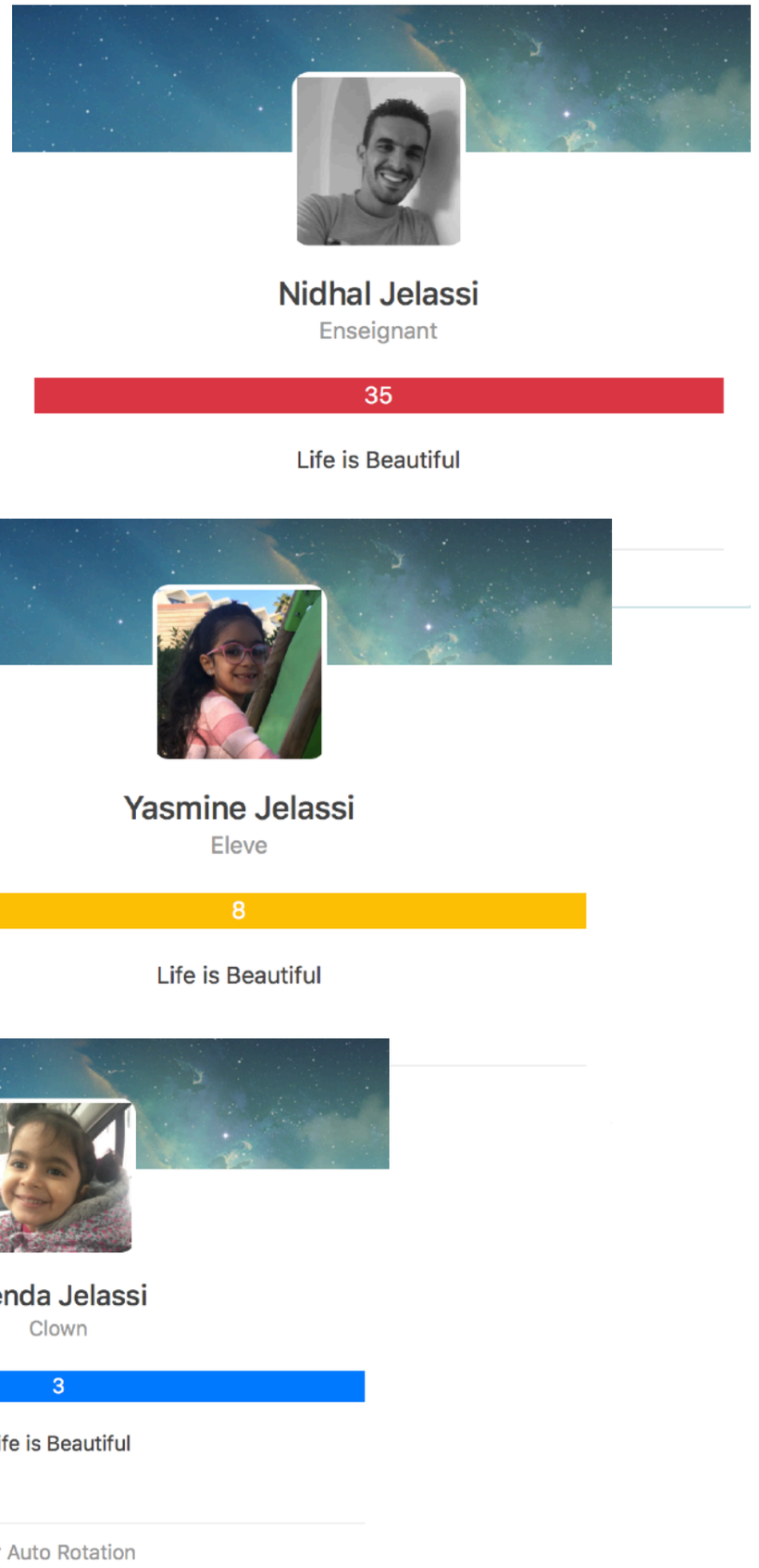
- Un pipe personnalisé est une classe décoré avec le décorateur **@Pipe**.
- Elle implémente l'interface **PipeTransform.**
- Elle doit implémenter la méthode **transform** qui prend en paramètre la valeur cible ainsi qu'un ensemble d'options.
- La méthode **transform** doit retourner la valeur transformée.
- Le pipe doit être déclaré au niveau de votre module de la même manière qu'une directive ou un composant.
- Pout créer un pipe avec le cli : **ng g p nomPipe**

EXEMPLE

```
<p class ="{{selectedPersonne.age | age}}">{{selectedPersonne.age}} </p>
```

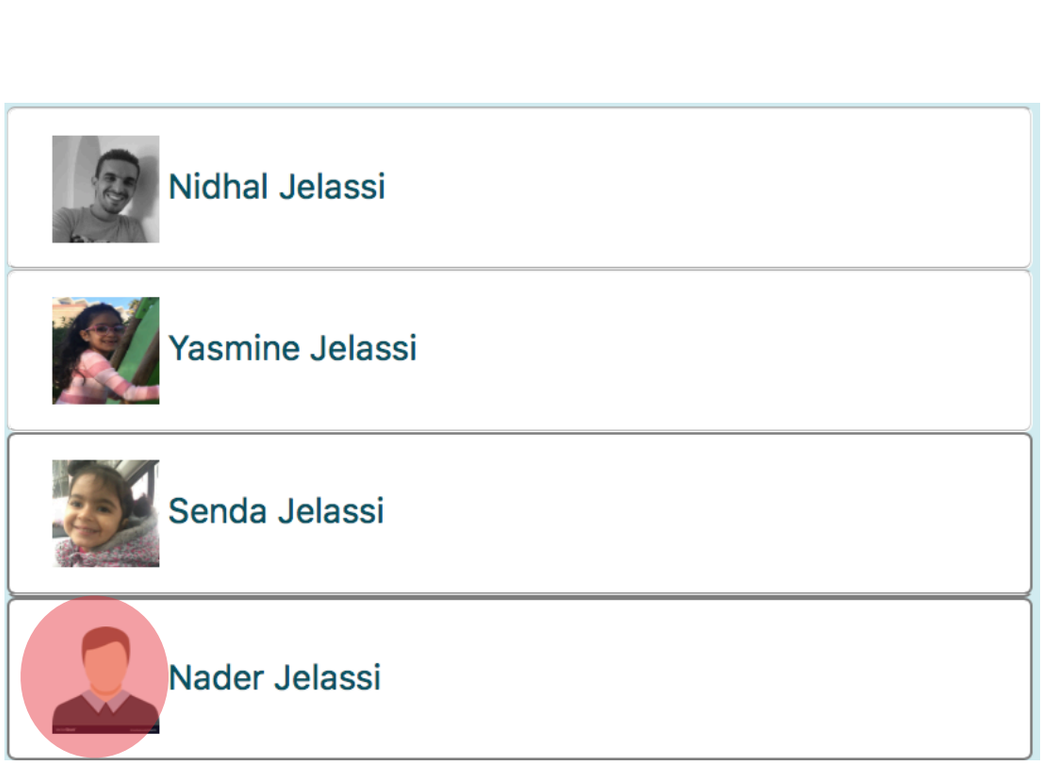
+

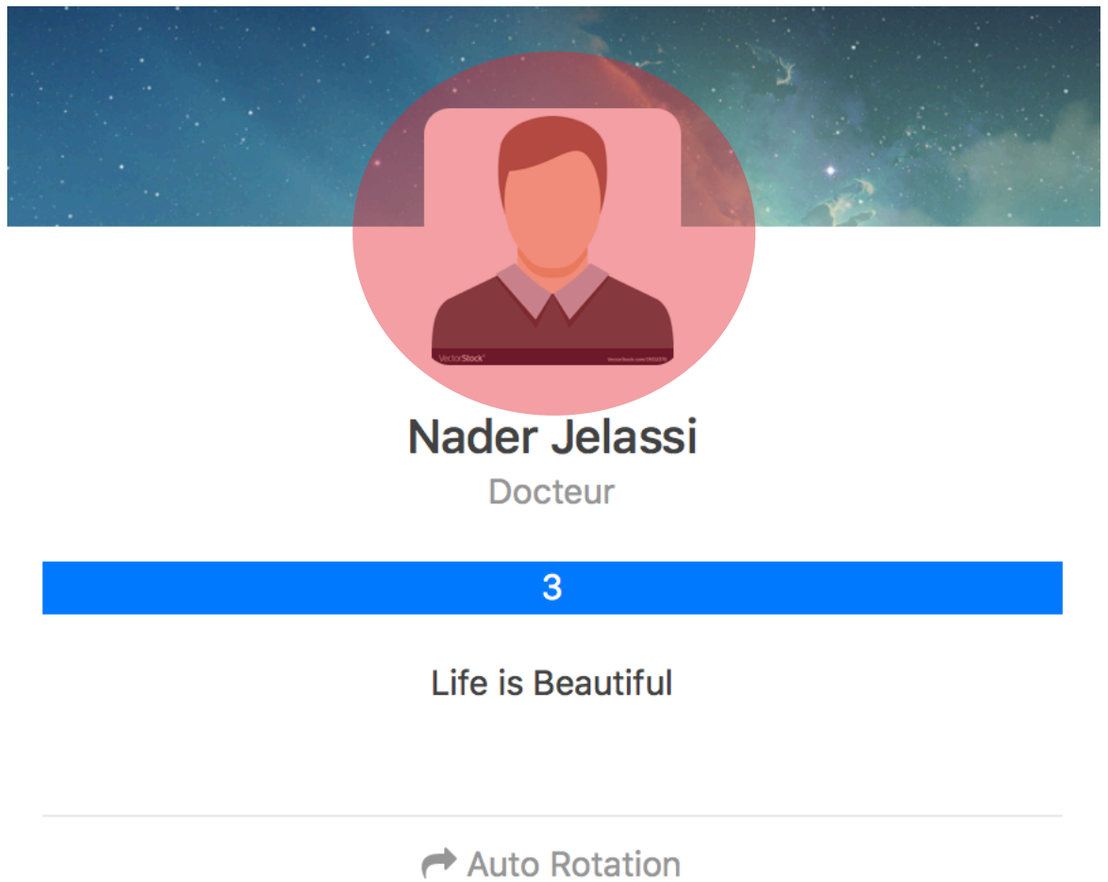
```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'age'
})
export class AgePipe implements PipeTransform {

  transform(age: number): string {
    let color : string;
    if(age>0 && age<4)
      color = 'bg-primary text-white';
    else if(age>=4 && age <10)
      color = 'bg-warning text-white';
    else
      color = 'bg-danger text-white';
    return 'profession ' + color;

  }
}
```



RETOUR AU PROJET CV

- Créer un pipe appelé defaultImage qui retourne le nom d'une image par défaut que vous stockerai dans vos assets au cas ou la valeur fournie au pipe est une chaine vide.





PIPES PURES ET IMPURES

- Par défaut, les Pipes sont purs. C'est à dire que leur méthode **transform** est une fonction pure, la valeur retournée ne dépend que des paramètres reçus et les appels n'ont aucun effet de bord. Autrement dit, le Pipe est "stateless".

- Les Pipes impurs sont évalués à chaque Change Detection même quand les paramètres ne changent pas. Ils sont déclarés en passant la valeur false au paramètre **pure** du décorateur **Pipe**.

```
@Pipe({
  name: 'filter',
  pure: false
})
```

PIPES PURES ET IMPURES

Nativement, seuls les Pipes suivant sont impurs :

- **async** : Il est impur car les valeurs retournées par ce Pipe peuvent changer à n'importe quel moment étant donné qu'elles proviennent d'une source asynchrone (Observable ou Promise).

- **json** : Etant donné qu'il est principalement utilisé pour le "debug", ce Pipe est impur car il est préférable que la valeur retournée soit mise à jour même si l'immutabilité n'est pas respectée.

- **slice** : Bien que ce Pipe fonctionnerait parfaitement en étant pur, il est probablement impur pour faciliter l'adoption d'Angular par la communauté car malheureusement l'immutabilité des Arrays est rarement respectée par les développeurs Angular.