

Angular : pipe

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`



Plan

- 1 Introduction
- 2 Créer un pipe
- 3 Personnaliser un pipe
- 4 Pipes globaux

Angular

Pipe

- a le rôle de formater et modifier l'affichage d'une donnée dans le `.component.html`
- est une classe décorée par le décorateur `@pipe`
- doit implémenter la méthode `transform(value: any, args?: any)` de l'interface `PipeTransform`
- peut prendre un ou plusieurs paramètres

Angular

Exemple

```
{{ "bonjour" | uppercase }}  
<!-- BONJOUR -->
```

© Achref EL MOUELHI ©

Angular

Exemple

```
{{ "bonjour" | uppercase }}  
<!-- BONJOUR -->
```

Certains pipes peuvent prendre des paramètres (déclarer `maDate = Date.now()` **; dans** `component.ts`)

```
{{ maDate | date:'d MMM y' }}  
<!-- affiche 19 oct 2018 -->
```

Angular

Exemple

```
{{ "bonjour" | uppercase }}  
<!-- BONJOUR -->
```

Certains pipes peuvent prendre des paramètres (déclarer `maDate = Date.now()` **; dans** `component.ts`)

```
{{ maDate | date:'d MMM y' }}  
<!-- affiche 19 oct 2018 -->
```

Il est possible d'enchaîner les pipes

```
{{ maDate | date:'d MMM y' | uppercase }}  
<!-- affiche 19 OCT 2018 -->
```

Angular

Considérons l'objet suivant défini dans un `.component.ts`

```
stagiaire: Stagiaire = new Stagiaire(100, 'Wick', 'John')  
;
```

© Achref EL MOUËL

Angular

Considérons l'objet suivant défini dans un `.component.ts`

```
stagiaire: Stagiaire = new Stagiaire(100, 'Wick', 'John')  
;
```

Avec `*ngFor`, on ne peut itérer sur des objets

```
<ul>  
  <li *ngFor="let elt of stagiaire">  
    </li>  
</ul>
```

Angular

Depuis Angular 6, un pipe `keyvalue` a été introduit pour pouvoir itérer sur les objets

```
<ul>
  <li *ngFor="let elt of stagiaire | keyvalue">
    {{ elt.key }} : {{ elt.value }}
  </li>
</ul>
```

Angular

Remarque

Angular propose dans `CommonModule` 13 pipes.

Angular

Pipes prédéfinis pour objets

- json
- keyvalue

© Achref EL MOUETRI

Angular

Pipes prédéfinis pour objets

- `json`
- `keyvalue`

Pipes prédéfinis pour chaînes de caractères

- `titlecase`
- `uppercase`
- `lowercase`
- `slice` (aussi pour les tableaux)

Angular

Pipes qui s'adaptent avec la valeur `locale` (i18n)

- `currency`
- `percent`
- `number`
- `date`
- ...

Angular

Les pipes

- pipes prédéfinis
- pipes personnalisés

Angular

Pour créer un pipe

```
ng generate pipe pipe-name
```

© Achref EL MOUELHI ©

Angular

Pour créer un pipe

```
ng generate pipe pipe-name
```

Ou le raccourci

```
ng g p pipe-name
```

Angular

Pour créer un pipe

```
ng generate pipe pipe-name
```

Ou le raccourci

```
ng g p pipe-name
```

Pour créer un pipe sans le fichier de test

```
ng g p pipe-name --skip-tests=true
```

Angular

Exemple : pour créer un nouveau pipe (`get-char`) qui retourne la première lettre d'une chaîne de caractères

```
ng g p get-char --skip-tests=true
```

© Achref EL MOU

Angular

Exemple : pour créer un nouveau pipe (`get-char`) qui retourne la première lettre d'une chaîne de caractères

```
ng g p get-char --skip-tests=true
```

Pour une meilleure structuration, plaçons le pipe dans un répertoire `pipes`

```
ng g p pipes/get-char --skip-tests=true
```

Angular

Constats

- Un fichier créé : `get-char.pipe.ts`
- Une mise à jour effectuée : déclaration du pipe dans `app.module.ts`

Angular

Contenu du `get-char.pipe.ts`

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

@Pipe({
  name: 'getChar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

Angular

Contenu du `get-char.pipe.ts`

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

@Pipe({
  name: 'getChar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

Explication

`value` est la valeur sur laquelle le pipe va s'appliquer.

Angular

Modifions `get-char.pipe.ts` **pour retourner la première lettre**

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

@Pipe({
  name: 'getChar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: string, ...args: unknown[]): string {
    return value[0];
  }
}
```

Angular

Testons ce nouveau pipe dans `app.component.html`

```
{{ "bonjour" | getChar }}
```

```
<!-- affiche b -->
```

```
{{ "wick" | getChar }}
```

```
<!-- affiche w -->
```

```
{{ "john" | getChar }}
```

```
<!-- affiche j -->
```

Angular

Modifions `get-char.pipe.ts` **pour retourner un caractère à une position donnée** (`pos`)

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

@Pipe({
  name: 'getChar'
})
export class GetCharPipe implements PipeTransform {

  transform(value: string, ...args: number[]): string {
    if (args[0] && args[0] < value.length) {
      return value[args[0]];
    }
    return value[0];
  }
}
```

Angular

Pour tester

```
{{ "bonjour" | getChar:2 }}  
<!-- affiche n -->  
  
{{ "wick" | getChar:6 }}  
<!-- affiche w -->  
  
{{ "john" | getChar }}  
<!-- affiche j -->
```

Angular

Remarque

Le pipe `getChar` est utilisable seulement dans le module où il a été déclaré.

© Achref EL MOUELHI ©

Angular

Remarque

Le pipe `getChar` est utilisable seulement dans le module où il a été déclaré.

Question

Comment l'utiliser dans plusieurs modules ?

Angular

Remarque

Le pipe `getChar` est utilisable seulement dans le module où il a été déclaré.

Question

Comment l'utiliser dans plusieurs modules ?

Solution

- Créer un nouveau module `shared`.
- Dans `shared` on déclare et exporte tous les éléments à partager entre plusieurs modules.
- Importer le module `shared` dans les modules où on a besoin d'utiliser le pipe.

Angular

Créons le module `shared` dans le répertoire `modules`

```
ng g m modules/shared
```

© Achref EL MOUELHI ©

Angular

Créons le module `shared` dans le répertoire `modules`

```
ng g m modules/shared
```

Contenu généré

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  declarations: [
  ],
  imports: [
    CommonModule,
  ],
})
export class SharedModule { }
```

Angular

Déclarons le pipe `getChar` dans `shared.module.ts`

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { GetCharPipe } from 'src/app/pipes/get-char.pipe';

@NgModule({
  declarations: [
    GetCharPipe
  ],
  imports: [
    CommonModule,
  ],
})
export class SharedModule { }
```

Angular

Exportons le pipe `getChar` dans `shared.module.ts`

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { GetCharPipe } from 'src/app/pipes/get-char.pipe';

@NgModule({
  declarations: [
    GetCharPipe
  ],
  imports: [
    CommonModule,
  ],
  exports: [
    GetCharPipe
  ]
})
export class SharedModule { }
```

Angular

N'oublions pas de supprimer la déclaration de `getChar` dans `app.module.ts`

© Achref EL MOUETRI

Angular

N'oublions pas de supprimer la déclaration de `getChar` dans `app.module.ts`

Remarque

Pour utiliser le pipe `getChar` dans un module `x`, il suffit d'importer `SharedModule` dans la section `imports` de `x`.

Angular

Exercice 1

Créer un pipe `even-value` qui permet de retourner, dans tableau, les valeurs paires d'un tableau.

© Achref EL MOUELHI

Angular

Exercice 1

Créer un pipe `even-value` qui permet de retourner, dans tableau, les valeurs paires d'un tableau.

Exemple

```
<ul>
  <li *ngFor="let elt of tab | evenValue">
    {{ elt }}
  </li>
</ul>
<!-- affiche 2 et 8
```

Angular

Exercice 2

Créer un pipe `str-to-array` qui permet de transformer une chaîne de caractère en tableau de caractères. Le pipe peut accepter 0, 1 ou deux paramètres.

- si aucun paramètre n'a été fourni, alors le pipe retourne un tableau contenant tous les caractères de la chaîne.
- si un seul paramètre a été passé, alors le pipe retourne les caractères de la sous-chaîne qui commence de la position indiquée par ce paramètre sous forme d'un tableau de caractère.
- si deux paramètres sont présents, alors le pipe retourne les caractères de la sous-chaîne située, entre les deux paramètres, sous forme d'un tableau de caractère.

Vous pouvez utiliser `Array.from(str)` ou `str.split("")` pour convertir la chaîne de caractères `str` en tableau de caractères.

Angular

Remarque

Un pipe peut être autonome en lui ajoutant l'option `standalone: true`.