Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі № 6

“Pipes. Створення та робота з pipes”

з дисципліни: «Реактивне програмування»

Студент: Зусько Владислав Юрійович

Група:  ІП-02

Дата захисту роботи:

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою:

Київ, 2023

**Зміст**

## Pipes: їхнє призначення та використання

У фреймворку Angular "pipes" (трубки) використовуються для обробки та трансформації даних, що відображаються в шаблонах компонентів. Вони є важливою частиною інтерфейсу користувача, оскільки дозволяють форматувати та відображати дані у зручному для користувача способі. Додамо більше деталей щодо призначення і використання трубок в Angular. Призначення трубок (Pipes) в Angular:

- Трубки використовуються для форматування різних типів даних, таких як рядки, числа, дати тощо. Наприклад, ви можете використовувати трубку date для відображення дати у більш зручному форматі.
- Трубки дозволяють вам відфільтрувати набір даних, видаливши елементи, які не відповідають певному умові. Так, трубка filter допоможе вам вибрати лише ті елементи масиву, які відповідають заданому критерію.
- Angular також надає трубку для сортування масивів, що дозволяє впорядковувати дані у специфічному порядку.
- Ви можете використовувати трубки для роботи з рядками, наприклад, обрізати, перетворити регістр, тощо.
- Angular також має можливість використовувати трубку i18n для багатомовного інтерфейсу, яка допомагає перекладати текст на інші мови.

Для використання трубок в шаблоні Angular ви можете вставити їх після виразу, який ви хочете трансформувати. Трубки можна легко комбінувати. Наприклад, ви можете використовувати трубку date разом з uppercase для відображення дати у верхньому регістрі.

Ви також можете створювати власні трубки, які відповідатимуть вашим потребам. Для цього вам потрібно створити клас імплементуючий інтерфейс PipeTransform та визначити метод transform.

Таким чином, pipes в Angular дозволяють зручно та ефективно трансформувати та відображати дані в шаблонах, роблячи їх більш читабельними та користувацькі-дружніми.

## Ланцюжки pipes

Ланцюжки pipes (pipe chains) в Angular - це механізм, який дозволяє вам використовувати один або кілька фільтрів (pipes) для обробки та форматування даних перед їх відображенням в шаблоні компоненту. Ланцюжок pipes складається з одного або декількох pipes, які виконуються послідовно один за одним. Вони допомагають зробити код більш читабельним і підтримуваним, оскільки ви можете розділити обробку даних на окремі кроки.

Ланцюжок pipes складається з послідовності pipes, які вказуються через вертикальну риску |.

Дані проходять через pipes у вказаному порядку. Спочатку застосовується перший pipe, результат передається наступному, і так далі. Це дозволяє вам створювати послідовні операції для обробки даних.

Багато pipes приймають параметри, які допомагають налаштовувати їх поведінку. Ці параметри можна вказати у шаблоні після імені pipe в круглих дужках.

Angular має декілька вбудованих pipes для звичайних операцій, таких як форматування дати, рядків, сортування тощо. Ви також можете створювати власні користувацькі pipes, щоб відобразити більше специфічних обробок даних.

Pipes у Angular працюють ліниво, що означає, що вони обробляють дані лише при необхідності, наприклад, якщо дані змінилися або компонент був перерендерений.

Ви можете комбінувати різні pipes у складніше обробку даних, що дозволяє створювати багатофункціональні обробки для ваших даних.

Ланцюжки pipes корисні для форматування та обробки даних у шаблонах Angular, щоб вони відповідали конкретним вимогам вашого

додатку. Вони полегшують читання та підтримку коду, дозволяють зменшити логіку обробки даних у компонентах та підвищують повторне використання коду.

## Створення своїх pipes

Створення власних pipes (каналів) в Angular дозволяє вам створювати власні фільтри для обробки та трансформації даних, які виводяться в шаблонах вашого додатку. Нижче наведено докладну інформацію щодо створення своїх pipes в Angular.

Для створення свого власного Pipe вам потрібно створити новий файл TypeScript у вашому проекті Angular, де ви реалізуєте кастомний Pipe. Наприклад, створіть файл my-custom.pipe.ts.

У вашому новому файлі my-custom.pipe.ts вам потрібно імпортувати необхідні ресурси Angular: import { Pipe, PipeTransform } from '@angular/core'; Після цього визначте ваш Pipe, який реалізує інтерфейс PipeTransform. Наприклад:

```
@Pipe({
  name: 'myCustomPipe'
})
export class MyCustomPipe implements PipeTransform {
  transform(value: any, ...args: any[]): any {
    // Логіка трансформації даних
    return transformedValue;
  }
}
```

У фразі @Pipe name - це ім'я, яке ви використовуватимете в вашому шаблоні для виклику вашого каналу.

Після створення свого Pipe ви можете використовувати його в шаблоні вашого компонента. Наприклад:

```
<div>{{ someData | myCustomPipe }}</div>
```

Тут someData - це дані, які ви бажаєте трансформувати за допомогою вашого Pipe, і myCustomPipe - ім'я вашого Pipe, вказане в декораторі @Pipe.

Ви можете передавати параметри до свого Pipe, якщо це необхідно. Визначте їх у методі transform і використовуйте їх в логіці трансформації.

Щоб Angular могла розпізнати ваш Pipe, ви повинні зареєструвати його в вашому модулі. Додайте ваш Pipe до розділу declarations модуля, наприклад:

```
declarations: [
  MyCustomPipe
]
```

Після цього Angular буде знати про ваш Pipe і дозволить вам використовувати його в шаблонах.

Це загальний огляд процесу створення власних pipes в Angular. Вам слід реалізувати логіку трансформації даних в методі transform вашого Pipe відповідно до вашого завдання.

## Передача параметрів у Pipes

У Angular, ви можете передавати параметри у власні Pipes, які використовуються для трансформації даних у шаблоні. Це робиться, додавши додаткові аргументи у шаблонний виклик Pipe. Ось деякі ключові моменти, пов'язані із передачею параметрів у Pipes:

У вашому шаблоні ви можете передати параметри у виклик вашого Pipe, використовуючи двокрапку та ім'я параметру, наприклад:

```
{{ someData | myCustomPipe:param1:param2 }}
```

У цьому прикладі param1 і param2 - це параметри, які ви передаєте у ваш Pipe.

У вашому Pipe, ви можете отримати передані параметри у методі transform за допомогою аргументів. Наприклад, якщо ви передали два параметри, ваш метод transform може виглядати так:

```
transform(value: any, param1: any, param2: any): any {
  // Ваша логіка трансформації з використанням параметрів param1 і param2
  return transformedValue;
}
```

В методі transform ви можете використовувати параметри param1 і param2 для зміни трансформації даних.

Параметри можуть бути будь-якого типу даних, таких як рядки, числа, об'єкти, масиви тощо. Ви можете приймати стільки параметрів, скільки потрібно для вашого Pipe.

Ви можете називати параметри у шаблоні та вашому Pipe як завгодно. Важливо, щоб кількість параметрів у виклику Pipe відповідала кількості параметрів у методі transform вашого Pipe.

Це загальний підхід до передачі параметрів у Pipes в Angular. Ви можете використовувати це для створення більш гнучких та налаштованих Pipes для трансформації даних у вашому додатку.

## Pure та Impure pipes

В Angular, існують два типи Pipes: Pure (чисті) та Impure (не чисті). Ось що означають ці терміни:

Pure Pipes:

- Чисті Pipes - це Pipes, які виконуються в тій же області пам'яті безперервності. Це означає, що якщо вхідні дані для Pipe не змінилися, то результат трансформації також не змінився.
- Чисті Pipes не виконують обчислення, якщо дані залишаються незмінними. Це робить їх ефективними в умовах оптимізації продуктивності додатків.
- Для позначення каналу як чистого, ви можете додати { pure: true } в декоратор @Pipe.

Impure Pipes:

- Не чисті Pipes - це Pipes, які можуть виконувати обчислення при кожному зміні циклу обнаруження змін, навіть якщо вхідні дані не змінилися.
- Це може призвести до зайвого обчислення і погіршити продуктивність додатку, особливо якщо у вас є не чисті Pipes, які викликаються багато разів.
- Не чисті Pipes можуть бути корисними в певних ситуаціях, коли вам потрібно виконати обчислення при кожному циклі обнаруження змін навіть для незмінних даних.
- Для позначення каналу як не чистого, ви можете просто залишити відсутнім { pure: true } в декораторі @Pipe.

Вибір між чистими та не чистими Pipes залежить від ваших потреб та продуктивності додатку. Зазвичай рекомендується використовувати чисті

Pipes, оскільки вони ефективні та зручні для оптимізації. Але не чисті Pipes можуть бути корисними у випадках, коли потрібно виконати обчислення при кожному циклі обнаруження змін, навіть для незмінних даних.

## Async Pipes

AsyncPipe - це вбудований Pipe в Angular, який використовується для обробки асинхронних операцій та відображення їх результатів у шаблоні. Ось декілька ключових відомостей про AsyncPipe:

AsyncPipe дозволяє автоматично підписатися на асинхронний Observable або Promise та автоматично оновлювати вміст в шаблоні, коли значення змінюються. Ви можете використовувати його для відображення результатів асинхронних запитів до сервера, зчитування даних з потоку подій тощо.

AsyncPipe автоматично відписується від асинхронного потоку, коли компонент, який використовує цей Pipe, видаляється або перестає бути активним. Це допомагає уникнути проблем з витоками пам'яті і зробити код більш безпечним.

AsyncPipe також автоматично обробляє помилки, які можуть виникнути при асинхронному запиті, та виводить їх в шаблоні. Ви можете використовувати спеціальний інтерфейс для обробки помилок.

Коли асинхронний результат готовий, AsyncPipe автоматично оновлює відповідний фрагмент шаблону, дозволяючи вам відображати оновлені дані без необхідності вручного оновлення.

Для використання AsyncPipe у шаблоні ви просто додаєте його до виразу, який містить асинхронне значення. Наприклад: {{ myObservableValue | async }}.

Використання AsyncPipe дозволяє спростити роботу з асинхронними операціями та робить код більш читабельним і безпечним.

## Хостинг

1. https://zuskoip02laba6-1.web.app/
2. https://zuskoip02laba6-4.web.app/

## Література


1) https://angular.io/
2) https://github.com/angular/angular
3) https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_getting_started