Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі № 1

“Створення Angular-додатків “HelloApp” і «Shopping list». Прив'язка даних в Angular.”

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

Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони

Ключовим модулем додатку є кореневий. У ньому приєднуються додатково базові модулі, а саме браузерний (BrowserModule) та модуль форм (FormsModule), який використовується для двусторонніх прив'язок. Також, в межах модуля містяться декларації компонентів (AppComponent).

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import AppComponent from './app.component';

@NgModule({
  imports: [BrowserModule, FormsModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export default class AppModule {}
```

Додаток складається з 1 кореневого компоненту. Компонент описує поля, які використані в шаблоні (name). Також, він специфікує селектор (app) для цього компоненту та посилання на файл шаблону (app.component.html).

```
import { Component } from '@angular/core';

@Component({
  selector: 'app',
  templateUrl: './app.component.html',
})
export default class AppComponent {
  protected name = ' ';
}
```

Також, застосунок містить 1 шаблон для кореневого компоненту. У ньому відбувається прив’язка поля імені до поля вводу і виведення вказаного значення.

```
<label>Введіть назву:</label>
<input [(ngModel)]="name" placeholder="name" />
<h1>Ласкаво просимо {{ name }}!</h1>
```

Опис основних структурних блоків Angular-додатку «Shopping list»: модулі, компоненти, шаблони

Кореневий модуль містить приєднання базових частин додатку. Зокрема браузерного та модуля форм. Додатково там міститься приєднання базового компоненту.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';

import AppComponent from './app.component';

@NgModule({
  imports: [BrowserModule, FormsModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export default class AppModule {}
```

Кореневий компонент в програмі описує поля, які користувач може заповнити та реалізує метод для додавання нової одиниці у список. Сам клас одиниці списку винесено окремо. Також, компонент описує селектор та посилання на шаблон.

```
import { Component } from '@angular/core';

import Item from '../domain/Item';

@Component({
  selector: 'app',
  templateUrl: './app.component.html',
})
export default class App {
  protected text = '';

  protected price = 0;

  protected items: Item[] = [
    { purchase: 'Хліб', done: false, price: 15.9 },
```

```
    { purchase: 'Вершкове масло', done: false, price: 60 },
    { purchase: 'Картопля', done: true, price: 22.6 },
    { purchase: 'Сир', done: false, price: 310 },
  ];

  public addItem(text: string, price: number) {
    if (text === null || text.trim() === '' || price === null || price
<= 0) {
      return;
    }

    this.items.push(new Item(text, price));
  }
}
```

Шаблон в свою чергу показує форму створення одиниці та виводить у вигляді таблиці список елементів. Активно використано структурні директиви та двосторонню прив’язку.

Опис файлу package.json. Призначення, основні параметри.

Файл package.json - це важливий файл у проектах на основі платформи Node.js, включаючи проекти Angular. Цей файл містить метадані про проект, його залежності, скрипти для виконання різних операцій та іншу інформацію. Основне призначення package.json - це забезпечення керування залежностями та конфігурацією проекту. Основні параметри package.json включають:

1. name: рядок, який визначає ім'я проекту. Ім'я повинно бути унікальним у межах репозиторію NPM.
2. version: рядок, що визначає версію проекту. Використовується для визначення, яку версію проекту ви використовуєте.
3. description: короткий опис проекту, який допомагає користувачам та розробникам зрозуміти призначення проекту.
4. main: шлях до основного файлу JavaScript, який виконується при запуску проекту за допомогою Node.js.
5. scripts: об'єкт, що містить набір команд для виконання певних операцій, таких як запуск сервера розробки, збірка проекту тощо. Найпоширеніші команди включають start (запуск додатку), build (збірка проекту), test (запуск тестів) та інші.
6. dependencies: об'єкт, що містить перелік залежностей, необхідних для роботи проекту у виробничому середовищі. Це можуть бути бібліотеки, фреймворки, модулі тощо.
7. devDependencies: об'єкт, що містить перелік залежностей, необхідних для розробки та збірки проекту, але необов'язкові для роботи виробничого додатку. Сюди зазвичай включаються розробницькі інструменти, тестові бібліотеки тощо.
8. keywords: масив ключових слів, що описують проект і допомагають знаходити його в репозиторії NPM.

9. author: інформація про автора або команду, що створила проект.
10. license: ліцензія, яка визначає умови використання проекту.
11. repository: інформація про репозиторій проекту, включаючи URL та тип (наприклад, Git).
12. engines: вказує версію Node.js, яка підтримується проектом.

Це лише основні параметри package.json, і файл може містити багато інших параметрів в залежності від потреб проекту. package.json грає важливу роль у керуванні залежностями, скриптами та налаштуваннями проекту, і його належне підтримування допомагає забезпечити стабільність та легкість розробки та впровадження вашого додатку.

Опис файлу tsconfig.json. Призначення, основні параметри.

Файл tsconfig.json (TypeScript Configuration) - це конфігураційний файл для TypeScript-проектів. TypeScript - це мова програмування, що розширює JavaScript, додаючи до нього сильну типізацію. Файл tsconfig.json визначає параметри компіляції TypeScript-коду в JavaScript та інші налаштування для проекту. Основне призначення tsconfig.json - це забезпечити контроль над компіляцією та типізацією в TypeScript-проекті. Основні параметри tsconfig.json включають:

1. compilerOptions: Цей об'єкт містить налаштування компілятора TypeScript. Деякі з найбільш важливих параметрів включають:
    a. target: Версія JavaScript, до якої буде компілюватися код (наприклад, "ES5" або "ES6").
    b. module: Формат модуля, який буде використовуватися (наприклад, "CommonJS" або "ESNext").
    c. outDir: Шлях до каталогу, куди буде збережений скомпільований JavaScript-код.
    d. rootDir: Початковий каталог, з якого буде компілюватися TypeScript-код.
    e. strict: Включає або виключає строгу типізацію та перевірку типів.
    f. noImplicitAny: Забороняє використовувати тип any без явного вказування типу.
    g. esModuleInterop: Дозволяє взаємодіяти з CommonJS-модулями в стилі ES6.
    h. sourceMap: Генерує файли карти джерела для полегшення налагодження.

2. include і exclude: Масиви шаблонів файлів, які повинні бути включені або виключені з компіляції. Наприклад, include може містити шаблони для файлів TypeScript, які потрібно компілювати.
3. files: Масив шаблонів файлів, які повинні бути включені у компіляцію, незалежно від include і exclude.
4. extends: Шлях до іншого файлу tsconfig.json, від якого поточна конфігурація унаслідує параметри.
5. compilerOptions.baseUrl і compilerOptions.paths: Налаштування для псевдонімів шляхів у вашому TypeScript-проекті (корисні для зменшення вкладеності шляхів).
6. references: Якщо ваш проект має декілька підпроектів (наприклад, пакети), то цей об'єкт дозволяє вам вказати залежності між ними.

tsconfig.json дозволяє налаштувати TypeScript для вашого проекту так, як потрібно, і забезпечити відповідність процесу компіляції вашим потребам. Для кожного проекту важливо належним чином налаштувати цей файл, оскільки це визначає правила для перетворення вашого TypeScript-коду в JavaScript, який можна виконувати в браузері або на сервері.

Опис файлу angular.json. Призначення, основні параметри.

Файл angular.json - це конфігураційний файл для проектів, розроблених за допомогою фреймворку Angular. Цей файл містить налаштування проекту, включаючи конфігурації збірки, налаштування відстеження змін файлів, завдання для виконання, опції розгортання та інші параметри, необхідні для розробки та побудови Angular-додатків. Основне призначення angular.json - це керування всіма аспектами розробки та збірки Angular-додатку. Основні параметри angular.json включають:

1. projects: Об'єкт, що містить конфігурації проектів. Зазвичай у вас є один основний проект з ім'ям "your-app-name", але ви також можете мати додаткові проекти (наприклад, бібліотеки).
    a. your-app-name: Конфігурація вашого основного Angular-додатку. Включає налаштування для збірки, відстеження файлів, завдання для виконання та інші параметри.
        i. root: Шлях до кореневої теки проекту.
        ii. sourceRoot: Шлях до кореневої теки джерела коду (де знаходяться файли Angular).
        iii. architect: Об'єкт, що містить налаштування завдань для розробки та збірки, такі як build (збірка проекту), serve (запуск розробницького сервера), test (запуск тестів) тощо.
2. newProjectRoot: Вказує шлях до теки, в якій буде створено новий проект за допомогою Angular CLI.
3. defaultProject: Ім'я проекту, яке буде використовуватися за замовчуванням при використанні Angular CLI.
4. schematics: Конфігурація схем (generators) для створення компонентів, модулів та інших частин додатку за допомогою Angular CLI

5. cli: Конфігурація Angular CLI, така як версія, експериментальні функції тощо.

Це лише загальний огляд основних параметрів angular.json. Файл angular.json дозволяє налаштовувати та керувати всіма аспектами розробки та збірки Angular-додатків і є ключовим для успішної роботи з Angular-проектами.

Інтерполяція в Angular: огляд, приклади використання.

Інтерполяція - це один з основних способів відображення даних в шаблонах Angular. Вона дозволяє вставляти значення з компонентів в шаблони, що дозволяє вам відображати динамічні дані на сторінці. У шаблонах Angular інтерполяцію виконується за допомогою пари фігурних дужок {{ }}. Інтерполяція в Angular дозволяє вставляти значення з компонентів у шаблони. Приклади використання:

1. У шаблоні можна використовувати інтерполяцію для вставки тексту з компоненту через його поле.
2. Якщо є об'єкт у компоненті, доступно використовувати інтерполяцію, щоб вставити його властивості в шаблон.
3. Інтерполяція також підтримує вирази. Можна виконувати обчислення та відображати їх результати.
4. Також, можна використовувати інтерполяцію для виклику методів компоненту та відображення результату.

Це лише декілька прикладів використання інтерполяції в Angular. Вона дозволяє створювати динамічні та інтерактивні сторінки, вставляючи дані з компонентів безпосередньо в HTML-шаблони.

Прив'язка властивостей елементів HTML: огляд приклади використання

Прив'язка властивостей елементів HTML в Angular - це механізм, який дозволяє зв'язувати значення з компоненту з властивостями HTML-елементів. Це дозволяє створювати динамічні та інтерактивні сторінки, де дані з компоненту автоматично оновлюються на сторінці і навпаки. Основний механізм для прив'язки властивостей - це використання атрибутів, таких як [property] у шаблонах Angular. Ось приклади використання прив'язки властивостей в Angular. У кожному з них властивість обраховується залежно від поля класу, проте там може бути використано будь-який вираз:

1. Прив'язка значення до атрибута:

<img [src]="imageUrl" alt="Зображення">

2. Прив'язка до властивості класу:

<div [class.active]="isActive">Активний елемент

3. Прив'язка до стилів елемента:

<p [style.color]="textStyles.color" [style.font-size.px]="textStyles.fontSize">Текст зі стилями</p>

Ці приклади демонструють різні способи прив'язки властивостей елементів HTML до даних та подій у компоненті Angular. Прив'язка властивостей є потужним інструментом для створення динамічних та інтерактивних інтерфейсів у додатку.

Прив'язка до атрибуту: огляд приклади використання.

В Angular, прив'язка до атрибуту через attr. використовується для динамічної зміни значень HTML-атрибутів на основі даних, які знаходяться в компоненті. Це особливо корисно, коли ви хочете змінити атрибути, які не мають своїх властивостей DOM у вигляді пропсів Angular. Прив'язка до атрибуту в Angular використовується з використанням [attr.attributeName] у шаблоні, де attributeName - це назва атрибуту HTML-елемента, який ви хочете змінити. Значення цього атрибуту буде встановлено на значення властивості компонента. Приклади використання:

1. Зміна значення атрибута "title" для елемента:

<button [attr.title]="tooltipText">Кнопка зі вспливаючою підказкою</button>

Цей приклад дозволяє встановити значення атрибута "title" для кнопки на основі значення tooltipText у компоненті.

2. Зміна значення атрибута "data-*":

<div [attr.data-custom]="customDataValue">Елемент з даними

Цей приклад дозволяє встановити значення атрибута "data-custom" для елемента на основі значення customDataValue у компоненті.

3. Зміна атрибута "aria-label" для покращення доступності:

<button [attr.aria-label]="accessibilityLabel">Кнопка з покращеною доступністю</button>

Цей приклад дозволяє покращити доступність кнопки, встановивши атрибут "aria-label" на основі значення accessibilityLabel у компоненті.

Прив'язка до атрибутів через attr. дозволяє змінювати атрибути HTML-елементів на сторінці на основі даних з компоненту і забезпечує більшу гнучкість у керуванні вашим інтерфейсом.

Прив'язка до події: огляд приклади використання.

Прив'язка до подій в Angular дозволяє відгукатися на дії користувача (наприклад, клік мишею, натискання на клавішу) і виконувати відповідні функції або методи компонента. Це дозволяє створювати інтерактивні та реактивні додатки. Прив'язка до подій виконується за допомогою директиви (event) в шаблоні Angular. Ось огляд та приклади використання прив'язки до подій в Angular:

1. Відгук на подію кліку на кнопку:

<button (click)="onButtonClick()">Натисни мене</button>

Цей приклад викликає метод onButtonClick() компонента, коли користувач клікає на кнопку.

2. Відгук на подію вводу:

<input type="text" (input)="onInputChange($event)">

Цей приклад викликає метод onInputChange() компонента при зміні значення в полі введення і передає об'єкт події $event, який містить інформацію про подію вводу.

3. Відгук на подію наведення:

<div (mouseenter)="onMouseEnter()">Наведіть курсор миші

Цей приклад викликає метод onMouseEnter() компонента, коли користувач наводить курсор миші на елемент.

Прив'язка до подій дозволяє створювати реактивний інтерфейс, який відгукується на дії користувача та виконує відповідні дії.

Двостороння прив'язка: огляд приклади використання

Двостороння прив'язка в Angular базується на директиві [(ngModel)], яка забезпечує зв'язок між властивістю об'єкта компонента і елементом форми в шаблоні. Це дозволяє автоматично оновлювати дані в обох напрямках: з компонента до шаблону і навпаки. Приклади використання двосторонньої прив'язки:

1. Простий варіант двосторонньої прив'язки:

У шаблоні можна створити поле вводу, яке автоматично оновлює значення властивості компонента та навпаки:

```
<input [(ngModel)]="message">
<p>{{ message }}</p>
```

2. Використання двосторонньої прив'язки для об'єктів:

Можна використовувати двосторонню прив'язку для об'єктів. Наприклад, у шаблоні можна зв'язати властивість name об'єкта з полем вводу:

```
<input [(ngModel)]="user.name">
<p>{{ user.name }}</p>
```

Прив'язка стилів: огляд, приклади використання

Прив'язка стилів в Angular використовується з використанням [style.styleName] у шаблоні, де styleName - це назва CSS-стилю, який треба змінити, і вираз, який вказує на значення цього стилю. Можна змінювати один або декілька стилів одночасно. Приклади використання прив'язки стилів:

1. Зміна кольору тексту на основі умови:

У шаблоні можна змінювати колір тексту на основі умови, використовуючи прив'язку стилів:

<p [style.color]="isActive ? 'green' : 'red'">Текст змінює колір</p>

Цей приклад дозволяє змінювати колір тексту в залежності від значення isActive у компоненті.

2. Зміна розміру шрифта на основі даних компонента:

Можна використовувати прив'язку стилів для зміни розміру шрифта тексту:

<p [style.font-size.px]="fontSize">Текст змінює розмір шрифта</p>

Цей приклад дозволяє динамічно змінювати розмір шрифта тексту, використовуючи значення fontSize у компоненті.

3. Зміна декількох стилів одночасно:

<div [ngStyle]="{ 'background-color': bgColor, 'border': borderStyle, 'padding.px': padding }">Зміна кольору тла, рамки і відступу

Цей приклад дозволяє змінювати кілька стилів одночасно, використовуючи об'єкт стилів у компоненті.

Прив'язка стилів дозволяє створювати інтерактивні та динамічні інтерфейси, змінюючи вигляд та стилі елементів на сторінці на основі даних у компоненті. Вона робить роботу зі стилями більш гнучкою і реактивною.

Література

1) https://angular.io/
2) https://stackoverflow.com/