ДОДАТОК

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі No 1

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

з дисципліни: «Реактивне програмування»

Студент: Трофимов Данило Олегович

Група: ІП-02

Дата захисту роботи:

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою:

Київ, 2023

ЗМІСТ

Загальний хід роботи:

1. Згідно з інструкціями було встановлено npm пакет @angular/cli.
2. Було скопійовано та видозмінено авторство файлу package.json.
3. Наступним кроком було створено файли проєкту та поміщено зміст файлів, наведений у інструкції до виконання лабораторної роботи, відповідно до завдвння.
4. Наступним кроком було конфігуровано файли tsconfig.json та Angular.json.

Частина 1: Створення Angular-додатків “HelloApp” і «Shopping list»
Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони.

1. **Компонент app.component.ts**:
    - Цей файл містить опис компоненту Angular. Програма визначає компонент AppComponent.
    - Компонент має шакблон, який містить поле name, яке використовується для зберігання імені користувача.
    - Визначений в компоненті шаблон HTML, який включає в себе текстове поле для введення імені та відображення імені користувача.
2. **Модуль app.module.ts**:
    - Файл app.module.ts визначає основний модуль додатку AppModule.
    - В цьому модулі визначені необхідні імпорти, такі як BrowserModule і FormsModule, які встановлюють основні функціональність Angular.

        - BrowserModule - це один із стандартних модулів Angular, який призначений для використання у браузері. Він надає функціональність, необхідну для коректної роботи додатків Angular у браузері. BrowserModule важливий для того, щоб додаток правильно працював у веб-середовищі. Він ініціалізує необхідні ресурси для роботи з DOM, обробки подій і т. д.

- FormsModule - це модуль Angular, який надає підтримку для роботи з формами у додатку. Він дозволяє використовувати двостороннє зв'язування даних ([(ngModel)]) для автоматичної синхронізації значень форми з властивостями компонента. З FormsModule можна легко створювати і обробляти HTML-форми в додатку Angular. Цей модуль допомагає у збереженні та використанні даних, введених користувачем у формі, без необхідності явної роботи з DOM.

  - Також визначено декларації, які включають в себе компонент AppComponent, і вказано, що додаток повинен бути завантажений з компоненту AppComponent.

3. **Індексний файл index.html**:
   - Файл index.html є головним HTML-файлом додатку Angular.
   - У розділі <body>, використовується елемент <my-app>, який є селектором компонента AppComponent. Це місце, де Angular-додаток буде відображений.
4. **Головний TypyScript-файл main.ts**:
   - Файл main.ts є точкою входу для додатку Angular.
   - Він використовує функцію platformBrowserDynamic() для платформи браузера та gпочинає виконання основного модулю додатку AppModule, який був описаний у app.module.ts.
5. **Файл polyfills.ts:**
   - Файл polyfills.ts може містити поліфіли, які роблять одаток сумісним із старими браузерами або додають підтримку для певних функцій. У нас він імпортує zone.js, яка використовується для керування зонами в Angular і обробки подій.

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

1. **Компонент app.component.ts**:
    - Він містить клас Item, який представляє структуру покупок з полями purchase (назва товару), done (покупка зроблена чи ні) та price (ціна товару).
    - Компонент AppComponent визначає шаблон для відображення списку покупок і форми для додавання нових покупок.
    - У шаблоні використовуються директиви *ngFor для відображення списку покупок та [(ngModel)] для двостороннього зв'язування даних з полями вводу форми.
    - Масив items містить початкові дані про покупки.
    - Метод addItem(text: string, price: number) викликається при додаванні нової покупки і додає її до масиву items.
2. **Модуль app.module.ts**:
    - Він імпортує необхідні модулі, такі як BrowserModule і FormsModule, які дозволяють використовувати функціональність Angular у браузері та для роботи з формами.
    - Визначено компоненти, які доступні в цьому модулі (у нашому випадку це лише AppComponent).
    - Компонент AppComponent вказаний як компонент, який слід автоматично відображати при завантаженні додатку.
3. **Індексний файл index.html**:
    - Він імпортує необхідні модулі, такі як BrowserModule і FormsModule, які дозволяють використовувати функціональність Angular у браузері та для роботи з формами.
    - Визначено компоненти, які доступні в цьому модулі (у вашому випадку це лише AppComponent).
    - Компонент AppComponent вказаний як компонент, який слід автоматично відображати при завантаженні додатку.
4. **Головний TypyScript-файл main.ts**:

- Він імпортує необхідні модулі, такі як BrowserModule і FormsModule, які дозволяють використовувати функціональність Angular у браузері та для роботи з формами.
- Визначено компоненти, які доступні в цьому модулі (у вашому випадку це лише AppComponent).
- Компонент AppComponent вказаний як компонент, який слід автоматично відображати при завантаженні додатку.

5. **Файл polyfills.ts:**
   - Файл polyfills.ts може містити поліфіли, які роблять одаток сумісним із старими браузерами або додають підтримку для певних функцій. У нас він імпортує zone.js, яка використовується для керування зонами в Angular і обробки подій.

Опис файлу package.json. Призначення, основні параметри.

Файл package.json є частиною конфігурації вашого проекту Angular і містить інформацію про залежності, налаштування скриптів та інші параметри проекту. Ось розбір параметрів та їх призначення package.json для проєкту PurchaseApp:

1. "name": Цей параметр вказує ім'я проекту. У нашому випадку, ім'я проекту - "purchaseapp".
2. "version": Версія нашого проекту. У нас вказана версія "1.0.0".
3. "description": Короткий опис нашого проекту. У нас опис - "First Angular 16 Project".
4. "author": Ім'я автора проекту. У даному випадку, автор - "Danylo Trofymov".
5. "scripts": Цей об'єкт містить налаштування скриптів для виконання певних операцій у вашому проекті. У нас випадку є три скрипти:
   - "ng": Скрипт для запуску Angular CLI.
   - "start": Скрипт для запуску сервера розробки Angular за допомогою ng serve.
   - "build": Скрипт для збірки проекту за допомогою ng build.

6. "dependencies": Цей об'єкт містить перелік залежностей, які потрібні для роботи вашого проекту.
    - "@angular/common", "@angular/compiler", "@angular/core", "@angular/forms", "@angular/platform-browser", "@angular/platform-browser-dynamic", "@angular/router": Ці пакети представляють основний функціонал Angular, необхідний для створення веб-додатків.
    - "rxjs": Це бібліотека для роботи з асинхронним програмуванням та обробки потоків даних.
    - "zone.js": Ця бібліотека використовується Angular для керування зонами та обробки подій.
7. "devDependencies": Цей об'єкт містить перелік залежностей, які використовуються тільки під час розробки та збірки проекту.
    - "@angular-devkit/build-angular": Засіб для розробки Angular-додатків.
    - "@angular/cli": Angular CLI (Command Line Interface) для створення та управління Angular-проектами.
    - "@angular/compiler-cli": Компілятор TypeScript для Angular.
    - "typescript": TypeScript - мова програмування, яка використовується для розробки Angular-додатків.

Цей файл package.json дозволяє нам налаштовувати та керувати залежностями та скриптами Angular-додатку, а також за потреби мануально міняти версії пакетів, що вказані у залежностях.

Опис файлу tsconfig.json. Призначення, основні параметри.

Файл tsconfig.json у проєкті Angular визначає налаштування компілятора TypeScript для проєкту. Опис параметрів з цього файлу:

1. "compileOnSave": Цей параметр вказує, чи повинен компілятор TypeScript автоматично компілювати файли при збереженні змін. У нас він

встановлений в false, що означає, що компіляція не відбудеться автоматично під час збереження файлів.

2. "compilerOptions": Цей об'єкт містить різні параметри налаштування компілятора TypeScript:
    - "baseUrl": Вказує базовий шлях для визначення відносних шляхів. У нас встановлено "./", що означає поточний каталог.
    - "sourceMap": Вказує, чи створювати файли карт відображення (source maps), які допомагають відлагоджувати TypeScript-код. У нас включено створення карт відображення.
    - "declaration": Вказує, чи створювати файли оголошень (declaration files). У нашому випадку встановлено false, що означає, що файли оголошень не будуть створюватися.
    - "downlevelIteration": Встановлюється в true, щоб дозволити ітерацію за допомогою пониженого коду (downlevel code). Це може бути корисним для сумісності зі старими браузерами.
    - "experimentalDecorators": Вказує, чи дозволено експериментальні декоратори. У нащому випадку встановлено true, що дозволяє використовувати декоратори, такі як @Component, які використовуються в Angular.
    - "module": Вказує формат модуля, який використовується в TypeScript. У нас встановлено "esnext", що вказує на використання сучасного формату модуля.
    - "moduleResolution": Вказує, як розрішувати модулі. У нас встановлено "node", що означає використання Node.js-стилю розрішення модулів.
    - "target": Вказує, до якої версії ECMAScript компілювати код. У нашому випадку встановлено "es2022", що означає використання ECMAScript 2022.
    - "typeRoots": Масив шляхів до каталогів, де TypeScript буде шукати оголошення типів (type definitions). У нашому випадку встановлено шлях "node_modules/@types".

- "lib": Масив бібліотек, які включаються у проєкт. У нашому випадку включені "es2022" та "dom", що дозволяє використовувати функціональність ECMAScript 2022 та браузера.

3. "files": Цей масив містить перелік файлів, які будуть включені в компіляцію. У нашому випадку це src/main.ts і src/polyfills.ts.
4. "include": Масив шаблонів шляхів до файлів, які будуть включені у компіляцію. У нас - всі файли з розширенням .d.ts у папці src і її підпапках будуть включені.

Цей файл tsconfig.json дозволяє налаштувати та контролювати процес компіляції TypeScript-коду для проєкту Angular.

Опис файлу angular.json. Призначення, основні параметри.

Файл angular.json - це конфігураційний файл для вашого проєкту Angular, який містить налаштування для різних аспектів проєкту, таких як збірка, розгортання, тестування та інші операції. Ось опис основних параметрів з цього файлу для проєкту PurchaseApp:

1. "version": Версія конфігураційного файлу. У нашому випадку встановлено "1".
2. "projects": Цей об'єкт містить налаштування для проєктів в нашому репозиторії. У нас є один проєкт з іменем "purchaseapp".
    - "purchaseapp": Цей об'єкт представляє Angular-додаток і містить такі параметри:
        - "projectType": Тип проєкту. У нас це "application", що вказує, що це додаток.
        - "root": Кореневий каталог проєкту. У нашому випадку це порожній рядок, що означає, що кореневий каталог визначається автоматично.
        - "sourceRoot": Каталог, де знаходиться вихідний код проєкту. У нашому випадку це "src".

- "architect": Цей об'єкт містить конфігурацію для різних архітектурних операцій, таких як збірка, розгортання і службовий сервер розробки.
    - "build": Конфігурація для операції збірки проєкту. У нас використовується "@angular-devkit/build-angular:browser" як будівельник для компіляції веб-додатка.
        - "options": Цей об'єкт містить параметри для операції збірки, такі як:
            - "outputPath": Шлях, де буде розміщений скомпільований вихідний код. У нас це "dist/purchaseapp".
            - "index": Шлях до головного HTML-файлу вашого додатку. У нас - "src/index.html".
            - "main": Шлях до головного файлу TypeScript вашого додатку. У нас - "src/main.ts".
            - "polyfills": Шлях до файлу polyfills.ts, який імпортує поліфіли для роботи в різних браузерах. У нас - "src/polyfills.ts".
            - "tsConfig": Шлях до файлу tsconfig.json, який використовується для компіляції TypeScript. У нас - "tsconfig.json".
            - "aot": Флаг, який вказує, чи використовувати компіляцію Ahead-of-Time (AOT). У нас встановлено true, що означає використовувати AOT-компіляцію.
    - "serve": Конфігурація для операції розгортання в режимі розробки. У нашому випадку використовується "@angular-devkit/build-angular:dev-server" як будівельник для запуску локального сервера розробки.

- "options": Цей об'єкт містить параметри для операції розгортання в режимі розробки, такі як "browserTarget", який вказує на "purchaseapp:build" - це посилання на операцію збірки, яку слід використовувати під час розгортання.

3. "defaultProject": Ім'я проєкту, який вважається за проєкт за замовчуванням. У нас це "purchaseapp".

Цей файл angular.json дозволяє налаштовувати різні аспекти вашого Angular-проєкту, такі як налаштування збірки, розгортання, тестування і багато інших.

Посилання на додаток ShoppingApp: https://trofumovip02laba1-1.web.app/

Частина 2: Прив'язка даних.

Інтерполяція в Angular: огляд приклади використання.

Інтерполяція в Angular - це спосіб вставки значень властивостей компонента в шаблон HTML. Ви можете використовувати двокутні дужки {{}} для вставки значень властивостей компонента в шаблон.

Під час виконання лабораторної роботи було використано інтерполяцію для властивостей name і age, які знаходяться в класі AppComponent. Ці властивості вставляються в шаблон за допомогою інтерполяції. Коли компонент відображається, значення name відображатиметься в рядку "Ім'я: Tom", а значення age - в рядку "Вік: 25". Варто звернути увагу, що значення цих властивостей визначаються в самому компоненті. Це дозволяє динамічно змінювати дані в шаблоні, якщо властивості компонента змінюються під час виконання програми. Наприклад, якщо оновити значення name або age в методі компонента, і вони автоматично відобразяться в шаблоні.

Прив'язка властивостей елементів HTML: огляд приклади використання.

Прив'язка властивостей елементів HTML в Angular дозволяє вам зв'язувати значення властивостей компонента з властивостями або атрибутами елементів HTML. Ви можете використовувати квадратні дужки [property]="expression" для встановлення зв'язку між властивістю компонента і властивістю або атрибутом елемента. У нашому прикладі ми використовуємо прив'язку властивостей для елементів <input>, де значення властивості name компонента і age компонента зв'язуються з властивостями value цих елементів. Якщо запустити цей компонент, можна побачити, що значення полів вводу (<input>) будуть автоматично заповнені значеннями name і age, які визначені в компоненті. Якщо змінити значення name або age в компоненті, значення полів вводу також оновляться автоматично, завдяки прив'язці властивостей.

Прив'язка до атрибуту: огляд приклади використання.

Прив'язка до атрибуту в Angular дозволяє змінювати значення атрибутів HTML елементів на основі властивостей нашого компонента. Можна використовувати [attr.attributeName]="expression" для встановлення зв'язку між властивістю компонента і атрибутом елемента HTML. У нашому прикладі, ми використовуємо прив'язку до атрибуту [attr.colspan] для зміни значення атрибуту colspan в елементі <td>. У компоненті ми повинні мати властивість colspan, яка визначає значення атрибуту colspan для першого рядка вашої таблиці. Це дозволяє нам динамічно змінювати значення атрибутів HTML на основі даних в нашому компоненті. Якщо змінити значення colspan в компоненті, воно також оновиться в атрибуті colspan елемента <td>.

Прив'язка до події: огляд приклади використання.

Прив'язка до події в Angular дозволяє виконувати дії у відповідь на події, які виникають в елементах DOM, такі як клік миші, натискання клавіші тощо. Ми можемо використовувати (event)="handler()" для встановлення

зв'язку між подією та методом компонента. У нащому прикладі, и використовуємо прив'язку до події (click) для кнопки <button>, щоб відслідковувати подію кліку і виконувати метод increase() при кожному кліку на кнопку. У компоненті ми маємо властивість count, яка ініціалізується значенням 0. При кожному кліку на кнопку, метод increase() виконується, і значення count збільшується на 1. Зміни значення count автоматично відображаються в шаблоні, завдяки прив'язці до властивості count. Це дозволяє нам створювати реактивний інтерфейс, де події спричиняють виконання коду в нашому компоненті, і результат відображається на сторінці.

Двостороння прив'язка: огляд приклади використання.

Двостороння прив'язка в Angular дозволяє нам об'єднувати значення властивостей компонента і значення елементів форми таким чином, що зміни в одному автоматично відображаються в іншому. Для досягнення цього ми використовували директиву ngModel, яка надає можливість двосторонньої прив'язки між властивістю компонента і елементом форми. У нашому прикладі ми використовуємо двосторонню прив'язку для елементів <input> і властивості name компонента. В нашому прикладі, коли ми вводимо текст у першому або другому полі вводу, зміни автоматично відображаються в властивості name нашого компонента. Це означає, що якщо ми введемо "John" у полі вводу, то властивість name буде автоматично оновлена і в шаблоні відобразиться "Ім'я: John". Також змінеться друге поле вводу, яке прийме значення “John” Зміни властивості name також впливають на значення полів вводу. Тобто, якщо оновити name в коді компонента, ці зміни автоматично відобразяться у полях вводу. Двостороння прив'язка корисна при роботі з формами і дозволяє підтримувати спільні дані між компонентом і елементами форми без додаткового коду для оновлення і відображення.

Прив'язка до класів CSS: огляд приклади використання.

Прив'язка до класів CSS в Angular дозволяє нам змінювати класи елементів HTML на основі властивостей нашого компонента. Ви можете використовувати [class.className]="expression" для встановлення зв'язку між властивістю компонента і класами елемента HTML. У нашому прикладі, ми використовуємо прив'язку до класів CSS для двох елементів і використовуєте <input> для зміни стану змінної isRed.
У цьому прикладі, коли значення isRed дорівнює true, клас isredbox додається до елементів , і вони отримують червоний фон. Коли значення isRed дорівнює false, клас isredbox видаляється, і елементи отримують звичайний фон. Крім того, ми використовуємо двосторонню прив'язку (ngModel) для <input type="checkbox>, щоб змінювати значення isRed на основі вибору чекбоксу. Ця техніка дозволяє нам динамічно змінювати стилі елементів на основі стану вашого компонента і дозволяє створювати реактивні інтерфейси.

Прив'язка стилів: огляд, приклади використання.

Прив'язка стилів в Angular дозволяє нам динамічно змінювати стилі елементів на основі властивостей нашого компонента або інших умов. Ми можемо використовувати [style.property]="expression" для встановлення зв'язку між властивістю компонента і стилями елемента HTML. У нашому прикладі, ви використовуєте прив'язку стилів для кількох елементів і <input>. У нашому прикладі ми використовуємо прив'язку до класів CSS ([class.className]) та прив'язку до стилів ([style.property] або [style.property-name]) для зміни стилів елементів на основі значень змінних компонента (isRed і isyellow) або класу (blue). Відповідні зміни стилів відобразяться на сторінці при зміні значень цих змінних або класу.

Посилання на додаток Binding 1: https://trofymovip02laba1-2.web.app

Висновки

У ході виконнаня лабораторної роботи було виконано наступне:

1. Було створено Angular-додаток «HelloApp», де було ознайомлено зі структурою Angular-проєкту, а також з файлами package.json, tsconfig.json та angular.json. Сам додаток має базовий інтерфейс та виводить вітальне повідомлення на екрані, використовуючі односторонню прив’язку. А значення імені зберігається завдяки двосторонній прив’язці
2. Було створено Angular-додаток «ShoppingList», завдяки якому було ознайомлено з загальними можливостями Angular-додатку. Додаток має функціонал списку покупок з можливістю поставити позначки про покупки. Цей додаток використовує двостороннє зв’язування, прив’язку до події, а також директрива ngFor для ітерації по масиву та виокремлення елементів, завдяки чому будується таблиця.
3. Було створено Angular-додаток «Binding1», у якому було ознайомлено з механізмами односторонньої прив’язки, двосторонньої прив’язки, прив’язки до властивостей HTML, та прив’язки до атрибуту
4. Було створено Angular-додаток «Binding2», у якому було ознайомлено з механізмами прив’язки до стилів та прив’язки до класів CSS

Загалом під виконання лабораторної роботи було набуто значного розуміння Angular, фреймворку для розробки веб-додатків. Було опрацьовано основні механізми Angular, які допомагають створювати реактивні додатки.

Список літератури

5. Introduction to the Angular docs [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs.
6. Angular. Angular tutorials [Електронний ресурс] / Angular – Режим доступу до ресурсу: https://angular.io/tutorial.
7. The Angular Book [Електронний ресурс] – Режим доступу до ресурсу: https://angular-book.dev/.
8. Daniel Schmitz. Angular 14 from Scratch / Daniel Schmitz., 2022.
9. Denys Vuika. Developing with Angular / Denys Vuika., 2018.