**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ**

**СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Звіт по лабораторній роботі № 1

Створення Angular-додатків. Прив'язка даних в Angular

з дисципліни: «Реактивне програмування»

Студентка: Степанюк Марта Олегівна

Група: ІО-04

Дата захисту роботи: 22.09.2023

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою:

Київ, 2023

**Зміст**

[a) Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони. 3](#_Toc145610398)

[b) Опис основних структурних блоків Angular-додатку «Shopping list»: модулі, компоненти, шаблони. 7](#_Toc145610399)

[c) Опис файлу package.json. Призначення, основні параметри. 9](#_Toc145610400)

[d) Опис файлу tsconfig.json. Призначення, основні параметри. 11](#_Toc145610401)

[e) Опис файлу angular.json. Призначення, основні параметри. 13](#_Toc145610402)

[1) Інтерполяція в Angular: огляд приклади використання. 15](#_Toc145610403)

[2) Прив'язка властивостей елементів HTML: огляд приклади використання. 15](#_Toc145610404)

[3) Прив'язка до атрибуту: огляд приклади використання. 16](#_Toc145610405)

[4) Прив'язка до події: огляд приклади використання. 16](#_Toc145610406)

[5) Двостороння прив'язка: огляд приклади використання. 17](#_Toc145610407)

[6) Прив'язка до класів CSS: огляд приклади використання. 18](#_Toc145610408)

[7) Прив'язка стилів: огляд, приклади використання. 18](#_Toc145610409)

**Основна частина**

Мої проекти можна переглянути за наступними посиланнями:

Angular-додаток «Shopping list»:

*https://stepaniuki004laba1-1.web.app*

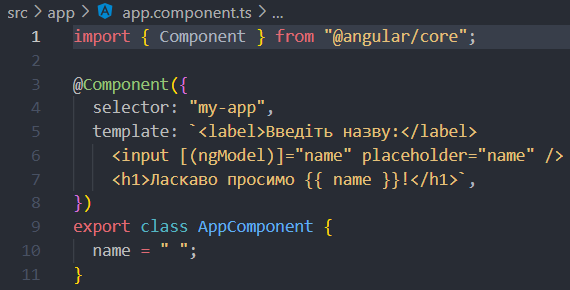
*https://stepaniuki004laba1-1.firebaseapp.com*

Angular-додаток Binding1:

*https://stepaniuki004laba1-2.web.app*

*https://stepaniuki004laba1-2.firebaseapp.com*

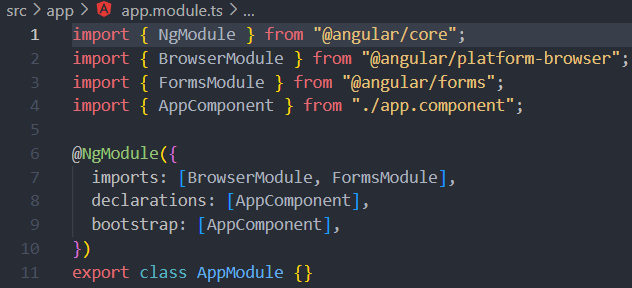
1. Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони.

*app.component.ts*:

Цей компонент є частиною Angular-додатка і називається *AppComponent*. Основна мета цього компонента - відображати користувацький інтерфейс для введення ім'я користувача та виводити повідомлення з цим ім'ям.

Основні характеристики цього компонента:

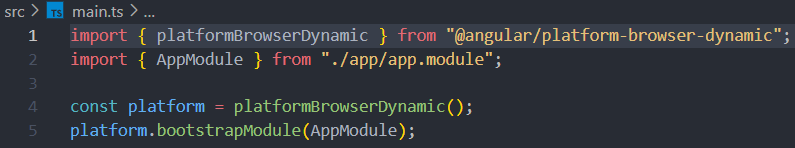
1. *Селектор: selector: "my-app"* - цей компонент може бути включений у HTML-сторінку за допомогою тегу *<my-app>*.
2. *Шаблон*: У шаблоні компонента використовуються HTML-теги і вирази Angular для створення користувацького інтерфейсу. Шаблон включає такі елементи:
   * *<label>Введіть назву:</label>* - це елемент *<label>*, який відображає текст "Введіть назву:".
   * *<input [(ngModel)]="name" placeholder="name" />* представляє собою текстове поле для введення імені користувача, яке автоматично синхронізується з властивістю *name* компонента через механізм *ngModel*, тобто будь-які зміни в полі вводу автоматично відображаються у значенні властивості *name* компонента.
   * *<h1>Ласкаво просимо {{ name }}!</h1>* - це заголовок *<h1>*, який відображає привітання з ім'ям користувача, яке також береться з властивості *name* компонента.
3. *Клас AppComponent*: Цей компонент є класом, який визначає поведінку компонента. У класі є одна властивість:
   * *name*: Властивість *name* встановлює початкове значення на " " (пробіл).

*app.module.ts*

Даний модуль є частиною Angular-додатка і називається *AppModule*. Він використовує декоратор *@NgModule*, який вказує Angular, як налаштовувати і складати додаток.

Основні характеристики цього модуля:

1. *Імпорти (imports)*: У розділі *imports* вказуються інші модулі, які будуть використовуватися в даному модулі. У даному випадку, модуль *AppModule* імпортує два модулі:
   * *BrowserModule*: Цей модуль надає фундаментальну функціональність для роботи з браузером, таку як робота з DOM-деревом.
   * *FormsModule*: Цей модуль надає підтримку для форм та дозволяє використовувати директиву *ngModel*, яка використовується для двостороннього зв'язку даних.
2. *Декларації (declarations)*: У розділі *declarations* вказуються компоненти, директиви, які належать до цього модуля. У цьому випадку є лише один компонент - *AppComponent*, який був імпортований з файлу "./app.component".
3. *Стартовий компонент (bootstrap)*: У розділі *bootstrap* вказується компонент, який буде кореневим для додатка. У цьому випадку, кореневим компонентом є *AppComponent*, тобто саме цей компонент буде відображатися при завантаженні додатка.

*main.ts:*

Наведений код є стандартним кодом для запуску Angular додатку в браузері за допомогою Angular-платформи.

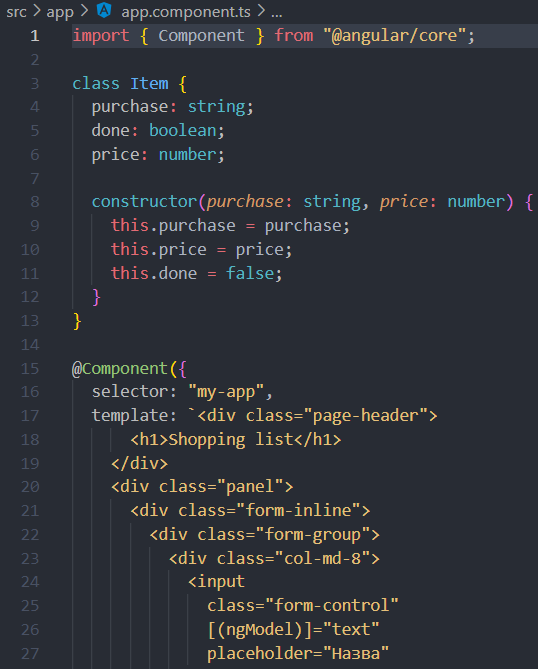
Основні характеристики цього коду:

1. *import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";:* Цей рядок імпортує функцію *platformBrowserDynamic* з модуля *@angular/platform-browser-dynamic*. Ця функція відповідає за запуск Angular додатку у браузері та динамічне компілювання та завантаження коду додатку.
2. *import { AppModule } from "./app/app.module";:* Цей рядок імпортує головний модуль додатку *AppModule* з файлу *"./app/app.module"*.
3. *const platform = platformBrowserDynamic();:* Тут створюється інстанс Angular-платформи, викликаючи функцію *platformBrowserDynamic()*. Це необхідно для підготовки середовища для запуску Angular додатку у браузері.
4. *platform.bootstrapModule(AppModule);:* Завершальний рядок коду, де запускається Angular додаток. Викликається метод *bootstrapModule*, який приймає головний модуль додатку *AppModule*. Цей метод розпочинає процес завантаження і компіляції коду додатку та запуску його в браузері.

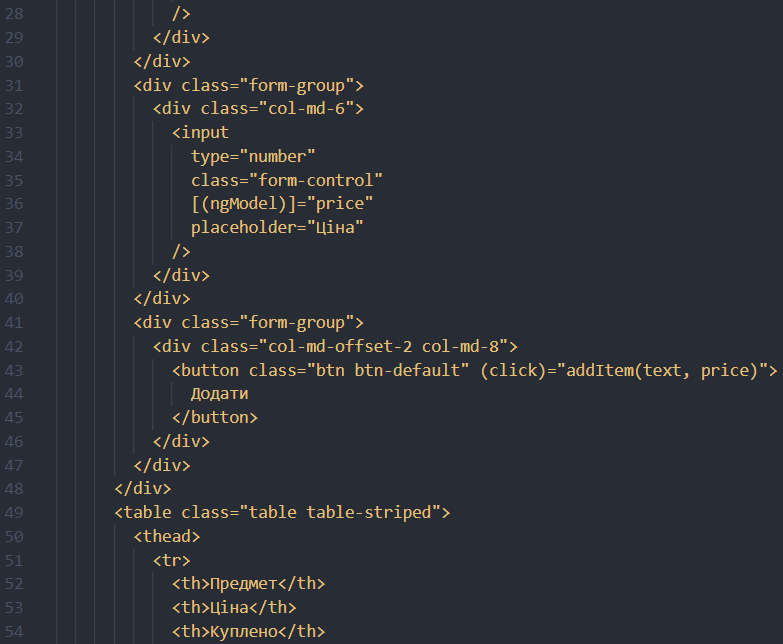
*polyfills.ts:*

Цей рядок коду імпортує зону (Zone) з бібліотеки "zone.js". Вона використовується в Angular-додатках для відстеження та керування асинхронними операціями, такими як події, запити до сервера та інші асинхронні взаємодії.

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

*app.component.ts*:

Цей компонент є частиною Angular-додатка і називається *AppComponent*. Він представляє собою простий список покупок і форму для додавання нових елементів у цей список.



Основні характеристики цього компонента:

1. *Клас Item*: Це клас, який представляє елемент списку покупок. Він має три властивості: *purchase* (назва покупки), *done* (прапорець, що вказує, чи куплено товар), і *price* (ціна товару). Конструктор класу приймає *purchase* і *price*, а *done* ініціалізується значенням *false* за замовчуванням.
2. *Декоратор @Component*: У цьому декораторі вказані метадані для компонента. Це включає в себе *selector* і *template*, який містить розмітку HTML для відображення користувацького інтерфейсу.
3. *Властивості класу AppComponent*:
   * *text* (string): Властивість для збереження назви товару, який користувач вводить в текстовому полі.
   * *price* (number): Властивість для збереження ціни товару, яку користувач вводить в числовому полі.
   * *items* (масив об'єктів Item): Властивість для збереження списку покупок. Початковий список містить декілька прикладових покупок.
4. *Метод addItem*: Цей метод викликається при натисканні на кнопку "Додати". Він додає новий елемент до списку покупок на основі значень *text* і *price*, які користувач ввів у відповідні поля. Перевірки на пусті значення виконуються перед додаванням нового елементу.
5. *Шаблон (template)*: Шаблон містить HTML-розмітку для відображення користувацького інтерфейсу. Він містить форму для введення назви і ціни товару, таблицю для відображення списку покупок і можливість позначити товари як куплені за допомогою прапорця. Для відображення списку покупок використовується директива *\*ngFor*, яка ітерується через елементи масиву *items* і відображає їх у таблиці.
6. Опис файлу package.json. Призначення, основні параметри.



Цей код представляє собою файл *package.json*, який містить інформацію про проект Angular, його залежності і команди для управління проектом. Основні параметри та їх призначення:

1. *"name"*: Ім'я проекту. У даному випадку, проект називається "helloapp".
2. *"version"*: Версія проекту. Версія встановлена на "1.0.0".
3. *"description"*: Короткий опис проекту. У даному випадку, описується як "First Angular 16 Project".
4. *"author"*: Автор проекту.
5. *"scripts"*: Об'єкт, який містить команди для виконання певних операцій у проекті. Основні команди:
   * *"ng"*: За допомогою цієї команди можна використовувати Angular CLI.
   * *"start"*: Запускає сервер розробки Angular за допомогою команди "ng serve".
   * *"build"*: Запускає процес збірки проекту за допомогою команди "ng build".
6. *"dependencies"*: Об'єкт, який містить залежності проекту. Це пакети, які потрібні для виконання проекту. Основні залежності:
   * *"@angular/common", "@angular/compiler", "@angular/core", "@angular/forms", "@angular/platform-browser", "@angular/platform-browser-dynamic", "@angular/router"*: Пакети Angular для розробки веб-додатків.
   * *"rxjs"*: Бібліотека для роботи з асинхронним програмуванням та реактивним програмуванням в Angular.
   * *"zone.js"*: Загальний пакет для керування контекстом виконання.
7. *"devDependencies"*: Об'єкт, який містить залежності для розробки та збірки проекту. Основні залежності:
   * *"@angular-devkit/build-angular"*: Засіб для розробки Angular, який допомагає збирати і виконувати різноманітні завдання для Angular-проекту.
   * *"@angular/cli"*: Командний рядок Angular CLI, який надає різні команди для розробки та збірки Angular-додатків.
   * *"@angular/compiler-cli"*: Компілятор TypeScript для Angular, який використовується під час збірки проекту.
   * *"typescript"*: Мова TypeScript, яка використовується для написання коду в Angular-проекті.
8. Опис файлу tsconfig.json. Призначення, основні параметри.

Цей код представляє собою конфігураційний файл для TypeScript з іменем *tsconfig.json*. Файл *tsconfig.json* використовується для налаштування параметрів компіляції TypeScript-коду у JavaScript. Основні параметри та їх призначення:

1. *"compileOnSave"*: *false*: Цей параметр вказує, чи повинен компілюватися код автоматично при збереженні файлу. У даному випадку, автоматична компіляція вимкнена (false).
2. *"compilerOptions"*: Об'єкт, який містить параметри компілятора TypeScript.
   * *"baseUrl": "./"*: Вказує кореневий каталог для визначення шляхів до модулів. У цьому випадку, використовується поточний каталог.
   * *"sourceMap": true*: Дозволяє генерувати файли карт джерела (source maps), що допомагають відлагоджувати TypeScript-код під час виконання у середовищі JavaScript.
   * *"declaration": false*: Вказує, чи створювати файли оголошень (declaration files) для TypeScript. У цьому випадку, оголошення не генеруються.
   * *"downlevelIteration": true*: Дозволяє використовувати ітерацію "downlevel" для компіляції async/await в більш старі версії JavaScript.
   * *"experimentalDecorators": true*: Дозволяє використовувати декоратори, які є експериментальною функцією в TypeScript.
   * *"module": "esnext"*: Вказує, який формат модуля використовувати під час компіляції TypeScript. У цьому випадку, використовується "esnext".
   * *"moduleResolution": "node"*: Вказує, який алгоритм розв'язання модулів використовувати. У цьому випадку, використовується "node", що означає використання Node.js-стилю для розв'язання модулів.
   * *"target": "es2022"*: Вказує, для якого стандарту ECMAScript компілювати код TypeScript. У цьому випадку, код компілюється для ECMAScript 2022.
   * *"typeRoots": ["node\_modules/@types"]*: Масив шляхів до каталогів, де знаходяться оголошення типів для властивостей, які не мають оголошення типів у власних пакетах.
   * *"lib": ["es2022", "dom"]*: Масив бібліотек, які мають бути доступні для TypeScript. У цьому випадку, доступні бібліотеки для ECMAScript 2022 та DOM.
3. *"files"*: Масив шляхів до файлів, які повинні бути включені в процес компіляції. У цьому випадку, включаються файли *src/main.ts* і *src/polyfills.ts*.
4. *"include"*: Масив шаблонів шляхів до файлів, які також повинні бути включені у компіляцію. У цьому випадку, включаються всі файли *.d.ts* (оголошення типів) у каталозі *src* і його підкаталогах.
5. Опис файлу angular.json. Призначення, основні параметри.



Цей код представляє собою файл конфігурації проекту для Angular, який використовується Angular CLI. Цей файл має назву *angular.json* і визначає налаштування проекту, його структуру та опції збірки. Основні параметри та їх призначення:

1. *"version"*: Версія конфігурації проекту. У даному випадку, версія вказана як 1.
2. *"projects"*: Об'єкт, який містить налаштування для конкретних проектів. У цьому випадку, є лише один проект з назвою "helloapp".
   * *"projectType"*: Тип проекту. У цьому випадку, тип вказаний як "application", що вказує на те, що це додаток Angular.
   * *"root"*: Кореневий каталог проекту. У даному випадку не вказаний, що означає кореневий каталог проекту.
   * *"sourceRoot"*: Шлях до кореневого каталогу з вихідними файлами проекту.
   * *"architect"*: Об'єкт, який містить налаштування для різних архітектурних завдань (наприклад, збірки і запуску) проекту.
     + *"build"*: Налаштування для завдання збірки проекту. Це завдання використовує *"@angular-devkit/build-angular:browser"* як будівельника, який компілює код і створює браузерну версію додатка. Опції включають *outputPath* (шлях для виведення результатів збірки), *index* (шлях до головного файлу HTML), *main* (головний файл TypeScript коду), *polyfills* (файл для поліфілів), *tsConfig* (файл налаштувань TypeScript) та *aot* (вказує, чи використовуватиметься компіляція AOT).
     + *"serve"*: Налаштування для завдання запуску сервера розробки. Використовує *"@angular-devkit/build-angular:dev-server"* як будівельника та посилається на завдання *"helloapp:build"* для налаштування.
3. *"defaultProject"*: Назва проекту, який вважається проектом за замовчуванням при виконанні команд Angular CLI.
4. *"cli"*: Об'єкт, який містить додаткові налаштування для Angular CLI.
   * *"analytics"*: Ідентифікатор аналітики, який може використовуватися для збору статистичних даних про використання Angular CLI.
5. Інтерполяція в Angular: огляд приклади використання.

Інтерполяція в Angular - це механізм вставки даних з компонента (класу) у шаблон компонента (HTML-розмітка). Вона дозволяє відображати значення змінних, функцій та виразів у Angular додатку.

***Додаток Binding1***

Вид прив'язки полягає у використанні фігурних дужок, в які передається значення з компонента.

*Відповідна частина app.component.ts:*

При запуску програми вирази типу {{name}} автоматично замінюватимуться відповідними значеннями, визначеними в компоненті. Якщо в процесі роботи програми властивості name і age в компоненті змінять своє значення, то також зміниться значення в розмітці html, яка прив'язана до цих властивостей.

1. Прив'язка властивостей елементів HTML: огляд приклади використання.

Наступні приклади демонструють різні способи прив'язки властивостей елементів HTML в Angular, що дозволяє динамічно маніпулювати вмістом.

***Додаток Binding1***

Використовується *[value]* для вставки значень *name* та *age* у властивості *value* двох текстових полів вводу.

*Відповідна частина app.component.ts:*

Змінна *name* вставляється у текстовий вміст елементу *<p>* за допомогою *[textContent]*.

*Відповідна частина app.component.ts:*

1. Прив'язка до атрибуту: огляд приклади використання.

У наступному прикладі ми бачимо використання прив'язки до атрибута. Зазвичай подібна прив'язка застосовується до атрибутів елементів aria, svg та table. Наприклад, атрибут colspan, який поєднує стовпці таблиці, не має відповідної властивості. І в цьому випадку ми можемо застосовувати прив'язку до атрибутів.

***Додаток Binding1***

Властивість *colspan* елементу *<td>* прив'язується до змінної *colspan* компонента за допомогою *[attr.colspan]*.

*Відповідна частина app.component.ts:*

1. Прив'язка до події: огляд приклади використання.

Наступний приклад показує використання прив'язки до подій для реагування на події користувача та виклику методів компонента при виникненні цих подій.

***Додаток Binding1***

Використовується *(click)* для прив'язки до події кліку на кнопці "Click", і ця подія викликає метод *increase()* компонента.

*Відповідна частина app.component.ts:*

Також використовується *(click)*, але в цьому випадку подія передається у метод *increase\_2($event)*, де виводиться подія у консоль.

*Відповідна частина app.component.ts:*

1. Двостороння прив'язка: огляд приклади використання.

Наступні приклади ілюструють двосторонню прив'язку, яка є потужним інструментом у Angular для спрощення взаємодії між шаблоном та компонентом, дозволяючи автоматично оновлювати дані в обох напрямках: з компонента до шаблону та назад.

***Додаток Binding1***

Використовується *[(ngModel)]* для двостороннього зв'язку зі змінною *name*. Це дозволяє як виводити значення змінної у поле вводу (при завантаженні сторінки), так і автоматично оновлювати значення змінної під час введення користувачем.

*Відповідна частина app.component.ts:*

***Додаток Binding2***

Використовується *[(ngModel)]* для двостороннього зв'язку стану чекбоксів зі змінними *isRed* та *isyellow*. Це дозволяє автоматично оновлювати значення цих змінних при зміні стану чекбоксів і, навпаки, відображати поточний стан чекбоксів на сторінці.

*Відповідна частина app.component.ts:*



1. Прив'язка до класів CSS: огляд приклади використання.

Наступні приклади ілюструють використання прив'язки до класів CSS в Angular, що дозволяє динамічно змінювати зовнішній вигляд елементів на сторінці.

***Додаток Binding2***

Використовується *[class.isredbox]* для додавання або видалення класу CSS "isredbox" до елементів *<div>* на основі значення змінної *isRed*. Якщо *isRed* дорівнює *true*, то додається клас "isredbox", якщо *false* - видаляється.

*Відповідна частина app.component.ts:*

Використовується *[class]* для динамічного встановлення класу CSS на основі значення змінної *blue*. У цьому випадку, якщо *blue* дорівнює "isbluebox", то цей клас буде застосовано до елементів *<div>*.

*Відповідна частина app.component.ts:*

1. Прив'язка стилів: огляд, приклади використання.

Наступний приклад ілюструє використання прив'язки до стилів (CSS) в Angular, що дозволяє динамічно змінювати стилі елементів на сторінці.

***Додаток Binding2***

Використовується *[style.backgroundColor]* для динамічної зміни фонового кольору елементів *<div>* на основі значення змінної *isyellow*. Якщо *isyellow* дорівнює *true*, то фон стає жовтим, в іншому випадку - синім.

*Відповідна частина app.component.ts:*

**Список використаних джерел**

1. Конспект лекцій.
2. Методичка з лабораторної роботи.
3. Офіційний сайт - https://angular.io/
4. TypeScript Documentation - https://www.typescriptlang.org/docs/