**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

# «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

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: Головня Олександр Ростиславович Група: ІП-11 Дата захисту роботи: Викладач: доц. Полупан Юлія Вікторівна Захищено з оцінкою:

Київ, 2024

Зміст

[«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО» 1](#_Toc178277137)

[Частина 1: Створення Angular-додатків “HelloApp” і «Shopping list» 3](#_Toc178277138)

[Опис основних структурних блоків Angular-додатку «HelloApp»: модулі, компоненти, шаблони. 3](#_Toc178277139)

[Компоненти та шаблони додатку «HelloApp»: 3](#_Toc178277140)

[Модулі додатку «HelloApp»: 4](#_Toc178277141)

[Створення головної сторінки 5](#_Toc178277142)

[Отримані результати: 5](#_Toc178277143)

[Опис основних структурних блоків Angular-додатку «Shopping list»: модулі, компоненти, шаблони. 6](#_Toc178277144)

[Компоненти та шаблони додатку «Shopping list»: 6](#_Toc178277145)

[Модулі додатку «Shopping list»: 8](#_Toc178277146)

[Створення головної сторінки 8](#_Toc178277147)

[Отримані результати: 9](#_Toc178277148)

[tsconfig.json 9](#_Toc178277149)

[angular.json 10](#_Toc178277150)

[Deploy на firebase: 13](#_Toc178277151)

[Частина 2: Прив'язка даних. 14](#_Toc178277152)

[Інтерполяція. Приклад використання. 14](#_Toc178277153)

[Прив'язка властивостей елементів HTML. Приклад використання. 14](#_Toc178277154)

[Прив'язка до атрибуту. Приклад використання. 15](#_Toc178277155)

[Прив’язка до події. Приклад використання. 16](#_Toc178277156)

[Двостороння прив’язка. Приклад використання. 17](#_Toc178277157)

[Прив’язка до класів CSS. Приклад використання. 19](#_Toc178277158)

[Прив’язка стилів. Приклад використання. 19](#_Toc178277159)

[Висновок: 20](#_Toc178277160)

[Список використаних джерел: 21](#_Toc178277161)

# Частина 1: Створення Angular-додатків “HelloApp” і «Shopping list»

**Мета: навчитися встановлювати необхідне ПО для створення Angular- додатку. Навчитися створювати шаблон в компоненті Angular.**

**Завдання:**

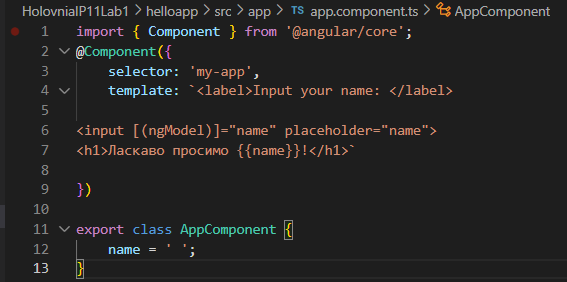
1. **створити при допомозі текстового редактора простий Angular-додаток “HelloApp”;**
2. **при допомозі текстового редактора створити простий додаток «Shopping list»;**
3. **зробити звіт по роботі;**
4. **розгорнути Angular-додаток «Shopping list» на платформі FireBase.**

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

Після створення package.json та встановлення усіх необхідних пакетів перейдемо до створення самого додатку. Для цього створимо в папці проекту підпапку, яку назвемо src - вона міститиме всі вихідні файли. І далі в папці src створимо підкаталог app.

## Компоненти та шаблони додатку «HelloApp»:

Даний додаток складатиметься лише з одного компоненту - AppComponent. Створимо у папці src/app новий файл, який назвемо app.component.ts і в якому визначимо наступний код компонента:

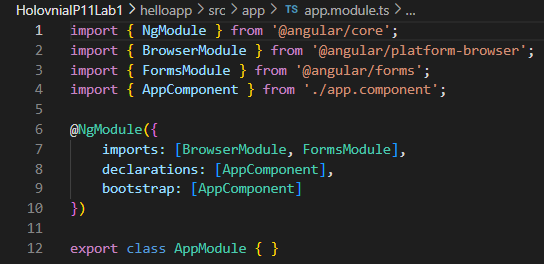


У цьому файлі ми імпортуємо функціональність модуля angular/core, аби мати змогу використовувати функції декоратора @Component. Далі власне йде функція-декоратор @Component, яка асоціює метадані із класом компонента AppComponent. У цій функції, по-перше, визначається параметр selector - селектор css для HTML-елемента, який міститиме даний компонент. У нашому випадку це 'my-app'. По-друге, тут визначається параметр template або шаблон, який вказує на те, яка буде внутрішня структура компоненту, які дані міститиме і як відображатиметься. Тут ми створили одне поле для введення та елемент заголовку, а також задали двосторонню прив'язку за допомогою виразів [(ngModel)]="name" і {{name}} до певної моделі name.

Наприкінці файлу створюється та експортується клас компонента AppComponent, у якому і зберігається змінна name. Її початковим значенням є порожній рядок.

## Модулі додатку «HelloApp»:

Кожен додаток Angular складається з модулів, що дозволяє легко підвантажувати та задіювати лише те, що безпосередньо необхідно. Даний додаток має лише один кореневий модуль, який і буде точкою входу додатку. Тому створимо у папці src/app новий файл, який назвемо app.module.ts з таким вмістом:

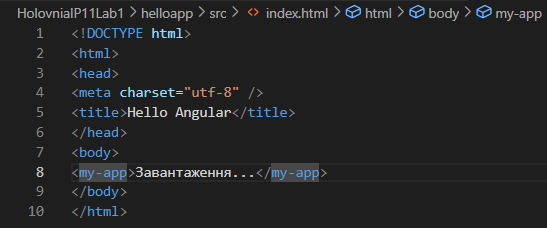


Як бачимо, процес створення модуля чимось нагадує процес створення компонента: сперш імпортуємо низку необхідних нам модулів, у тому числі функцію-декоратор NgModule та створений раніше клас AppComponent. Для роботи з браузером також потрібний модуль BrowserModule. Оскільки наш компонент використовує елемент input або елемент форми, також підключаємо модуль FormsModule. І далі імпортується створений раніше компонент.

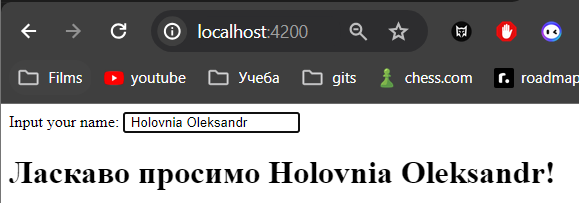
Також не забудемо створити головну сторінку з елементом my-app, у яку і відвантажуватиметься наш компонент.

## Створення головної сторінки

Далі визначимо в папці src головну сторінку index.html програми:

Після створення кількох додатковий файлів, таких як main.ts та polyfills.ts, а також angular.json та tsconfig.json, можемо запускати додаток за допомогою скриптів, визначених у package.json. Отримаємо сторінку, яка динамічно оновлюватиме вміст заголовку залежно від введених у текстове поле значень.

## Отримані результати:



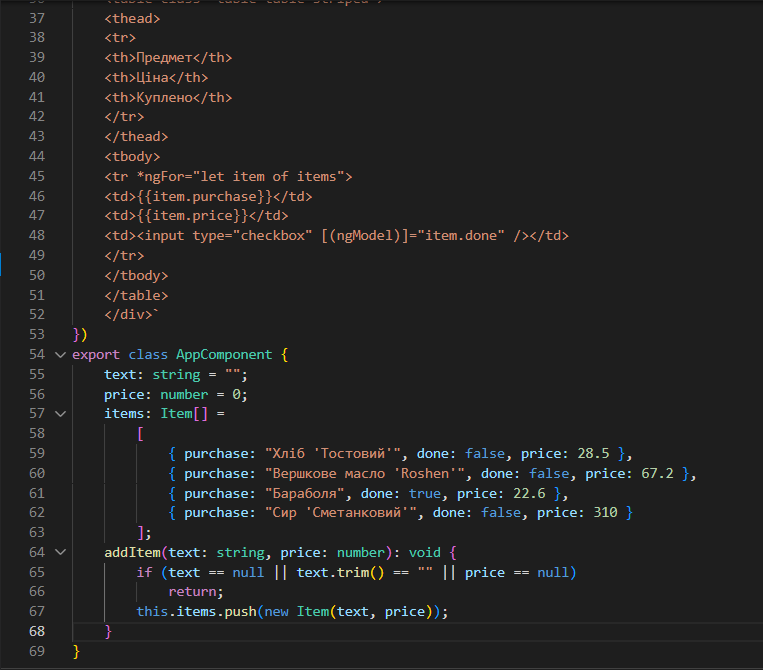
# Опис основних структурних блоків Angular-додатку «Shopping list»: модулі, компоненти, шаблони.

Як і для попереднього додатку, почнемо зі створення package.json та встановлення усіх необхідних пакетів, а потім перейдемо до створення самого додатку. Для цього створимо в папці проекту підпапку, яку назвемо src, а в ній - підкаталог app.

## Компоненти та шаблони додатку «Shopping list»:

Даний додаток теж складатиметься лише з одного компоненту - AppComponent. Створимо у папці src/app новий файл, який назвемо app.component.ts і в якому визначимо наступний код компонента:





У цьому файлі ми імпортуємо функціональність модуля angular/core, аби мати змогу використовувати функції декоратора @Component. Також створимо допоміжний клас Item. Цей клас містить три поля: purchase (назва покупки), done (чи зроблена покупка) та price (її ціна).

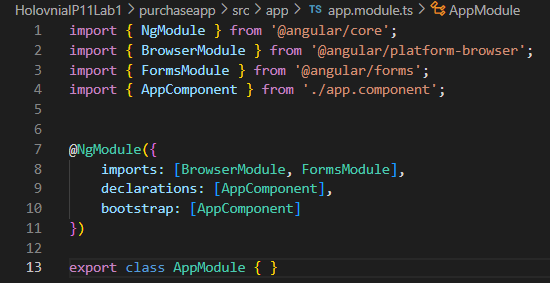
У самому класі компонента визначається початковий Shopping list, який виводитиметься на сторінку, а також визначено метод додавання до цього списку. Далі власне йде функція-декоратор @Component, яка асоціює метадані із класом компонента AppComponent. У цій функції, по-перше, визначається параметр selector - селектор css для HTML-елемента, який міститиме даний компонент. У нашому випадку це 'my-app'. По-друге, для виведення покупок в нашому компоненті визначено великий шаблон. У ньому для виведення даних з масиву items в таблицю передбачена директива:

\*ngFor="let item of items"

Крім того, зверху таблиці розташована форма для введення нового об'єкта Item. До натискання кнопки прив'язаний метод addItem() компонента. Наприкінці файлу створюється та експортується клас компонента AppComponent, у якому і зберігається всі дані.

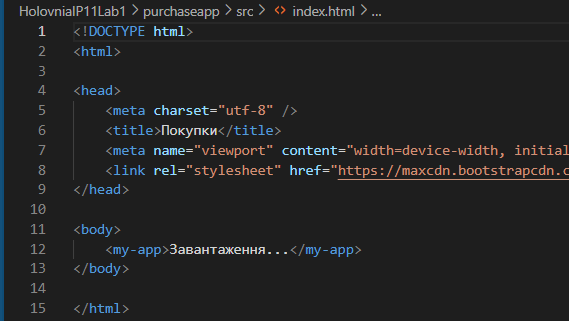
## Модулі додатку «Shopping list»:

Даний додаток має лише один кореневий модуль, код якого буде аналогічним коду такого ж модулю з попереднього додатку:



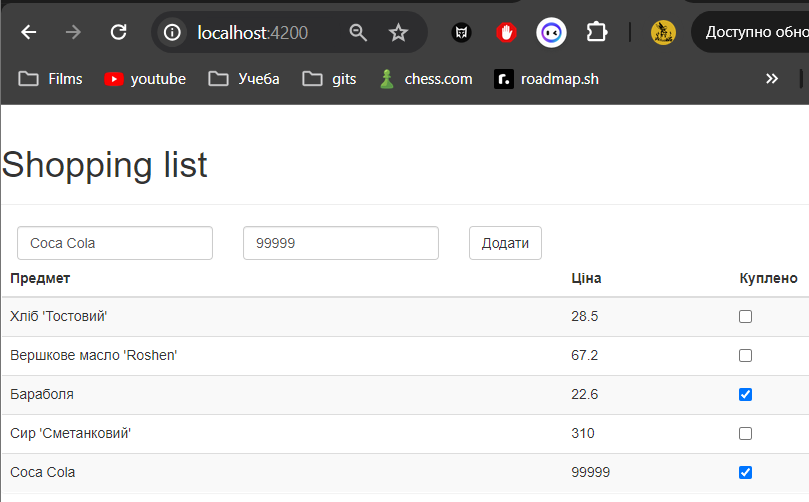
## Створення головної сторінки

Далі визначимо в папці src головну сторінку index.html програми:



Після створення кількох додатковий файлів, таких як main.ts та polyfills.ts, а також angular.json та tsconfig.json, можемо запускати додаток за допомогою скриптів, визначених у package.json. Отримаємо сторінку, яка динамічно формуватиме список покупок залежно від введених у текстове поле позицій та їх ціни.

## Отримані результати:



# tsconfig.json

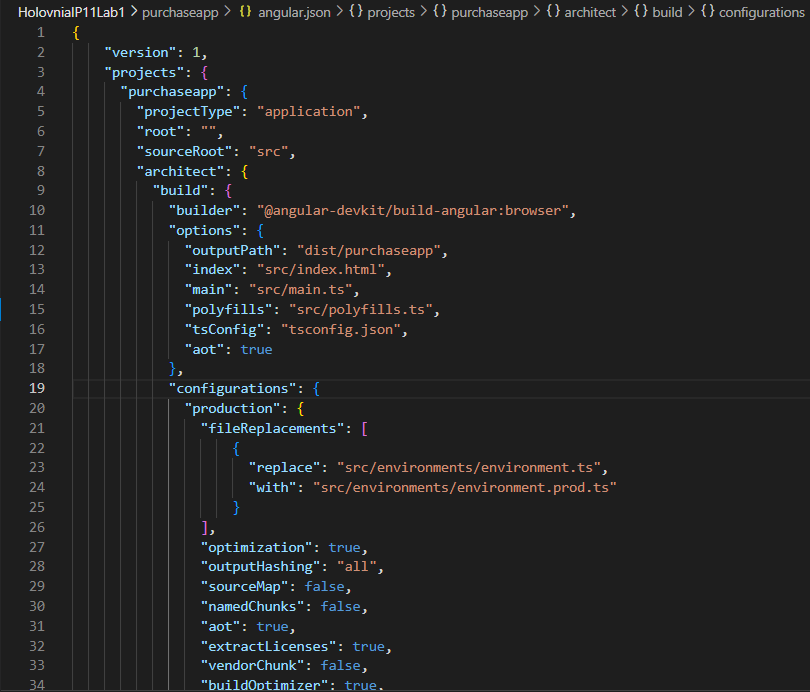
Оскільки для визначення коду програми використовується мова TypeScript, тому також створимо новий файл tsconfig.json:

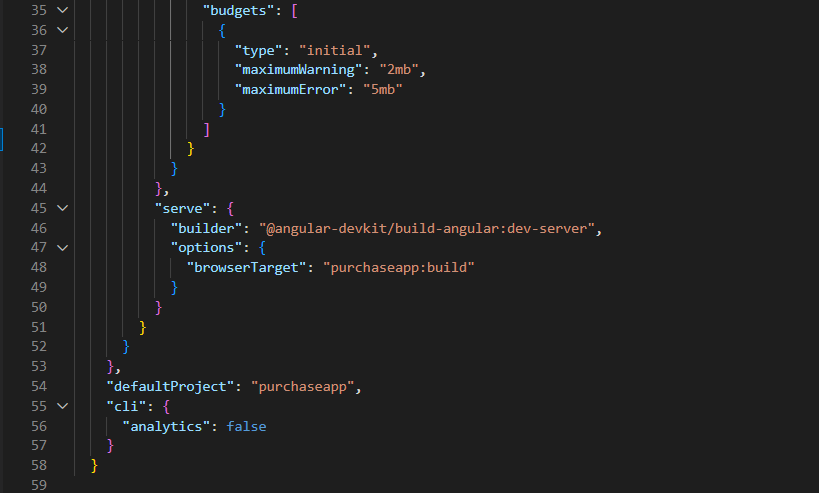
Цей файл визначає параметри компілятора TypeScript. Опція compilerOptions встановлює параметри компіляції. А опція "files" визначає файли, що компілюються. У нашому випадку це файл програми - main.ts, який підтягує решту файлів програми, і файл поліфілів polyfills.ts.



# angular.json

Для компіляції програми використовувати Angular CLI, тому ми повинні описати поведінку CLI за допомогою файлу angular.json. Отже, додамо до кореневої папки проекту новий файл angular.json і визначимо в ньому такий вміст:





Параметр version визначає версію конфігурації проекту.

Далі йде секція projects, яка визначає налаштування для кожного проекту. У нашому випадку у нас лише один проект, який називається за назвою каталогу – helloapp або purchaseapp.

Проект визначає такі опції:

* projectType: тип проекту. Значення "application" вказує, що проект представлятиме додаток, який можна буде запускати у браузері;
* root: вказує на папку файлів проекту відносно робочого середовища. Порожнє значення відповідає кореневій папці проекту, тому що в даному випадку робоче середовище та каталог проекту збігаються;
* sourceRoot: визначає кореневу папку файлів із вихідним кодом. У нашому випадку це папка src, де власне визначено всі файли програми;
* architect: задає параметри для побудови проекту. У файлі package.json визначені команди build та serve, і для кожної з цих команд у секції architect задані свої налаштування.

Для кожної команди задається параметр builder, який визначає інструмент для побудови проекту. Так, для команди "build" встановлено значення "@angular-devkit/build- angular:browser" - даний білдер для побудови використовує збирач пакетів webpack. А для команди "serve" встановлено значення "@angular-devkit/build-angular:dev-server" - даний білдер запускає веб-сервер і розгортає на ньому скомпільовану програму.

Параметр options визначає параметри побудови файлів. Для команди "build" тут визначено такі опції:

* + outputPath: шлях, за яким буде публікуватися скомпільований додаток;
  + index: шлях до головної сторінки програми;
  + main: шлях до головного файлу програми, де власне запускається програма Angular;
  + polyfills: шлях до файлу поліфілів;
  + tsConfig: шлях до файлу конфігурації TypeScript;
  + aot: вказує, чи використовуватиметься компіляція AOT (Ahead-Of-Time) (попередня компіляція перед виконанням). У цьому випадку значення true означає, що вона використовується.

Для команди "serve" вказана лише одна опція - browserTarget, яка містить посилання на конфігурацію для команди build - "helloapp:build". Тобто, по суті, ця команда використовує ту ж конфігурацію, що і команда build.

Остання опція defaultProject вказує на проект за замовчуванням. У цьому випадку це наш єдиний проект.

# Deploy на firebase:

Тепер за допомогою інструментів сервісу firebase опублікуємо наш за стосунок. Для цього виконаємо у консолі команди firebase login, firebase init та firebase deploy. По завершенню цих команд ми отримаємо посилання, за яким буде доступний наш додаток. Наприклад, мій Shopping List можна побачити за посиланням: <https://holovnialab1shoplist.web.app/>

# Частина 2: Прив'язка даних.

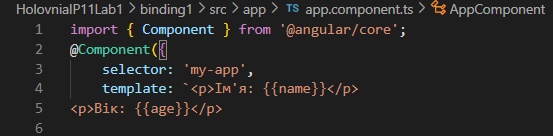
**Тема: Навчитися працювати з прив'язкою даних.**

**Завдання: Створити два Angular-додатки під назвою Binding1 та Binding2, як показано в частині 1.**

1. **Для Angular-додатку Binding1 виконати вправи 1-5;**
2. **Для Angular-додатку Binding2 виконати вправи 6-7;**
3. **Зробити звіт по роботі (по Angular-додатках Binding1 та Binding2);**
4. **Angular-додаток Binding1 розвернути на платформі FireBase.**

## Інтерполяція. Приклад використання.

У подвійних фігурних дужках вказується вираз, до якого йде прив'язка: {{вираз}}.





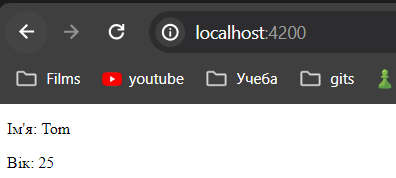
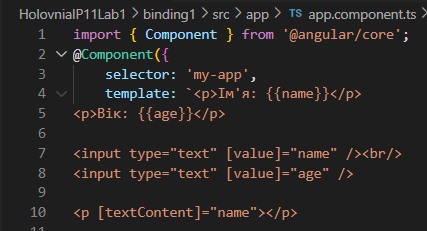


Рисунок 1.1. Приклад інтерполяції.

## Прив'язка властивостей елементів HTML. Приклад використання.

Одним із видів прив’язки є прив’язка властивостей елементів HTML, для цього потрібно у тегу HTML вказати властивість у квадратних дужках та присвоїти змінну до якої потрібно прив’язати. Треба пам’ятати, що прив’язка йде до властивості елементу, а не атрибуту.



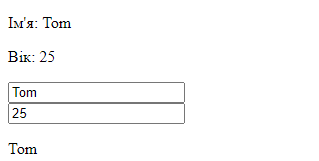
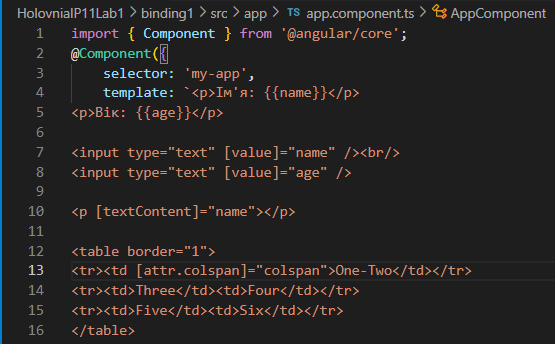
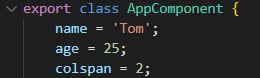


Рисунок 1.2. Приклад прив'язки до властивостей елементів HTML.

## Прив'язка до атрибуту. Приклад використання.

Іноді виникає необхідність виконати прив'язку не до властивості, а саме до атрибуту HTML-елемента. Це робиться за допомогою наступного виразу: [attr.назваАтрибуту]=”значення”.





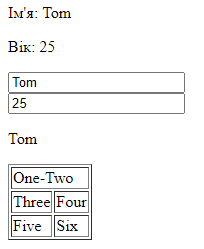
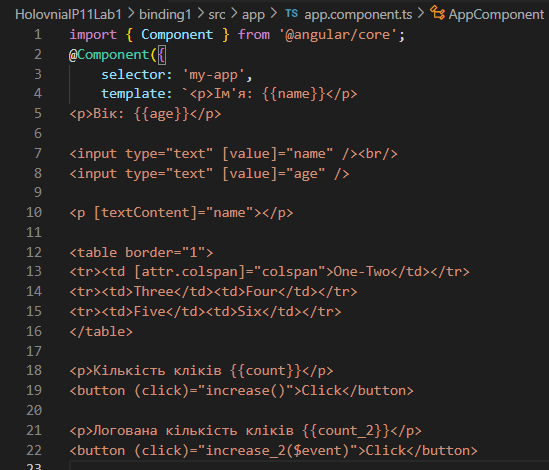
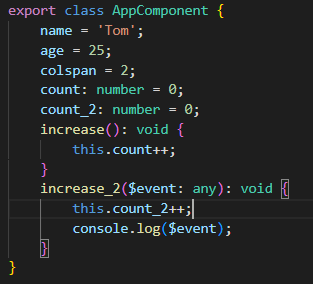


Рисунок 1.3. Приклад прив'язка до атрибуту.

## Прив’язка до події. Приклад використання.

Також можна прив’язатись до події. Коли подія буде відбуватись, буде викликатись обробник, який був вказаний при прив’язці. У прикладі прив’язка виконується до події кліку по кнопці: коли відбувається клік, викликається метод increase з компоненту і збільшується лічильник. Також ми можемо передати об’єкт події в обробник за допомогою виразу: [nameOfEvent]=”handler($event)”. У прикладі до функції increase\_2 передається об’єкт події кліку, який виводиться в консоль.





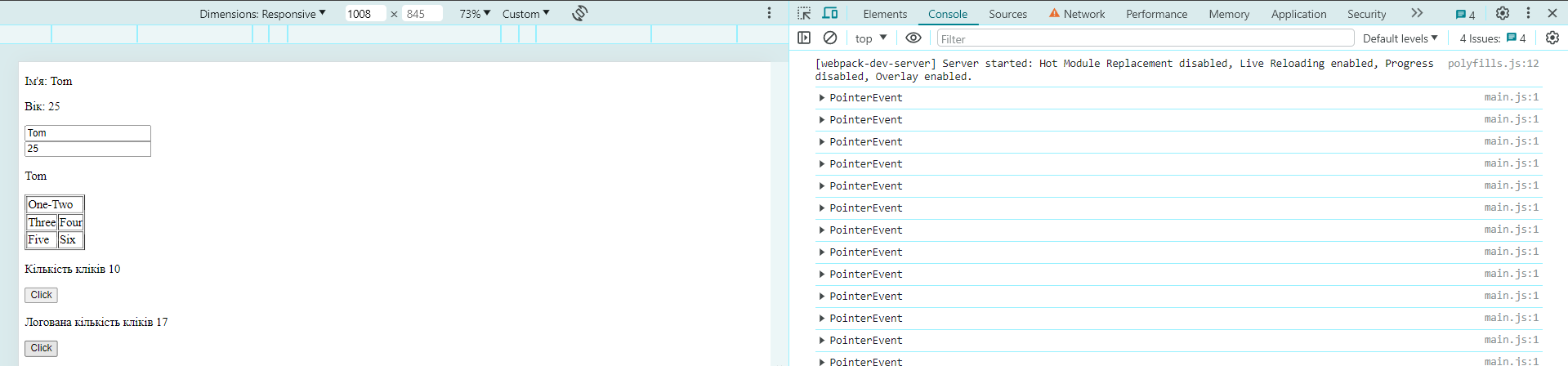
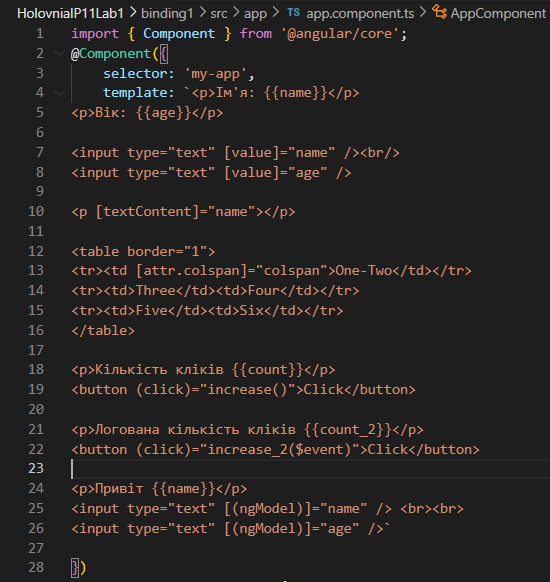


Рисунок 1.4. Приклад прив’язки до події.

## Двостороння прив’язка. Приклад використання.

Двостороння прив'язка дозволяє динамічно змінювати значення на одному кінці прив'язки при змінах на іншому кінці. Як правило, двостороння прив'язка застосовується під час роботи з елементами введення, наприклад, елементами типу input.



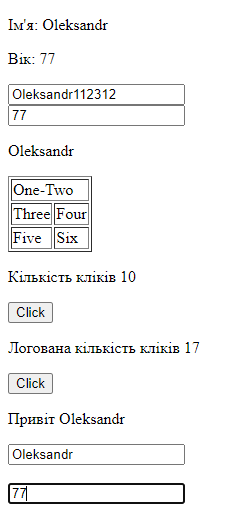
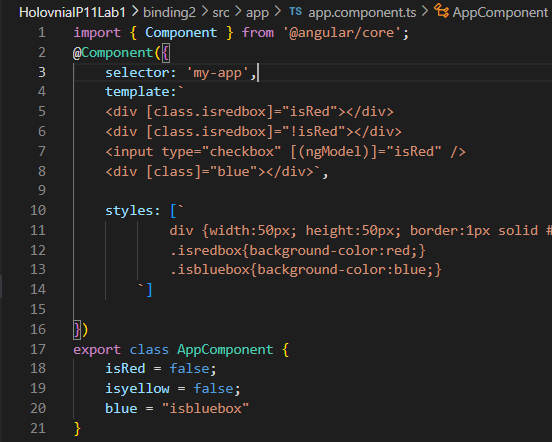


Рисунок 1.5. Приклад двосторонньої прив’язки.

## Прив’язка до класів CSS. Приклад використання.

Можемо прив’язувати елемент до класу CSS за допомогою виразу [class.nameOfClass]=”true/false”.



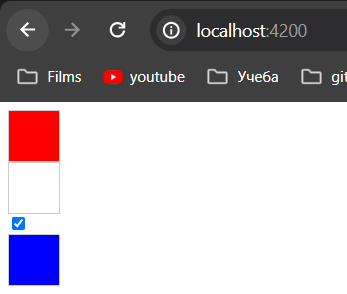
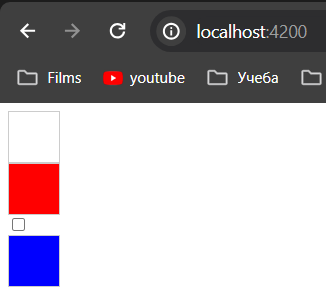
 

Рисунок 1.6. Приклад прив’язки до CSS.

## Прив’язка стилів. Приклад використання.

Можна також прив’язуватись напряму до стилів елементу, це виглядатиме як вбудовані стилі. Це можна зробити за допомогою виразу [style.nameOfStyle]=”value”.

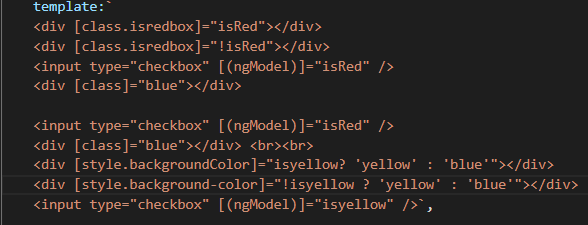




Рисунок 1.8. Приклад прив’язки стилів.

Посилання на додаток: <https://holovnialab1binding-7a6d9.web.app/>

# Висновок:

Під час виконання комп’ютерного практикуму я познайомився з основами Angular, вивчивтакі типи прив’язки, як інтерполяція, прив’язка властивостей HTML, прив’язка до атрибуту, прив’язка до події, прив’язка до класів CSS та прив’язка до стилів. Мною було створено 4 додатки, два з яких було розгорнуто на платформі firebase.

# Список використаних джерел:

1. Data Binding in Angular: https://www.javatpoint.com/data-binding-in-angular-8