**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 2

Робота з компонентами. Взаємодія

між компонентами. Прив’язка до подій дочірнього компоненту.

Життєвий цикл компоненту.

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**

[**ЗМІСТ** 2](#_Toc147436549)

[**Загальний хід роботи:** 3](#_Toc147436551)

[**1)** **Стилі та шаблони компонента.** 3](#_Toc147436552)

[**2)** **Селектор: host. Призначення та використання.** 4](#_Toc147436553)

[**3)** **Підключення зовнішніх файлів стилів та шаблонів.** 5](#_Toc147436554)

[**4)** **ng-content: призначення та використання.** 6](#_Toc147436555)

[**5)** **Взаємодія між компонентами. Передача даних у дочірній компонент.** 7](#_Toc147436556)

[**6)** **Прив'язка до сетера.** 8](#_Toc147436557)

[**7)** **Прив'язка до подій дочірнього компонента.** 9](#_Toc147436558)

[**8)** **Двостороння прив'язка.** 9](#_Toc147436559)

[**9)** **Життєвий цикл компонента** 10](#_Toc147436560)

[**Висновки** 12](#_Toc147436561)

[**Список літератури** 13](#_Toc147436562)

# **Загальний хід роботи:**

1. Згідно з інструкціями ініціалізовано перший додаток
2. Наступним кроком було опрацьовано завдання до лабораторної роботи для Components1
3. Далі дії було повторено для Components2- Components5
4. Додатки Components1 та Components5 було викладено на безкоштовний хостинг Firebase

# **Стилі та шаблони компонента.**

У фреймворку Angular стилі та шаблони компонента грають важливу роль у створенні динамічних та інтерактивних веб-додатків. Стилі в Angular зазвичай використовуються для визначення зовнішнього вигляду компонента, дозволяючи створювати привабливий та сучасний інтерфейс. Шаблони ж визначають структуру та вміст компонента, який відображатиметься на сторінці. Вони побудовані з використанням HTML та директив Angular, що надають можливість вставляти змінні, обробляти події та взаємодіяти з даними компонента. За допомогою компонентів Angular створюють масштабовані та модульні додатки, де стилі та шаблони є важливими складовими для досягнення зручного та ефективного веб-розробки.

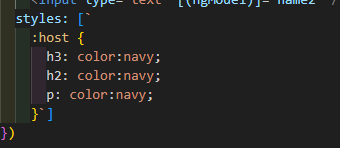
Під час виконання лаборатоної роботи було використано файл Даний приклад демонструє структуру та взаємодію стилів та шаблонів в Angular за допомогою двох основних файлів: app.module.ts та app.component.ts. У цьому файлі описано компонент AppComponent. Він має селектор 'my-app', що означає, що цей компонент може бути використаний у шаблонах за допомогою <my-app></my-app>. У шаблоні (template) компонента визначено HTML-розмітку, яка включає поле вводу та виведення імені. Директива [(ngModel)] використовується для двостороннього зв'язку даних, що дозволяє автоматично оновлювати значення name у компоненті, коли змінюється вміст поля вводу. Компонент також має властивість name, яка ініціалізується пустим рядком.

Цей приклад демонструє простий введення інтерактивного елементу веб-сторінки за допомогою Angular компонента та двостороннього зв'язку даних. Стилі можуть бути додані до цього компонента шляхом використання CSS-класів або inline-стилів в шаблоні для визначення зовнішнього вигляду елементів, що додаються до сторінки.

# **Селектор: host. Призначення та використання.**

Селектор :host - це особливий селектор в CSS, який використовується в контексті стилів компонентів Angular. Він дозволяє визначати стилі самого кореневого елемента компонента, тобто елемента, який містить весь вміст компонента на сторінці.

Основне призначення селектора :host в Angular полягає в тому, щоб надавати стилі та зовнішній вигляд самого компонента, а не його вкладених дочірніх елементів. Це корисно, оскільки в компоненті може бути багато елементів, і ви хочете застосовувати стилі тільки до кореневого контейнера, не впливаючи на стилі дочірніх елементів, які можуть бути використані в інших місцях додатку.



У цьому прикладі стилі визначені для кореневого елемента компонента. Вони будуть застосовані до цього елемента на сторінці, коли компонент буде відображений.

Цей механізм дозволяє забезпечити ізоляцію стилів компонентів від інших частин додатку та полегшити підтримку та обслуговування компонентів, особливо коли ви розробляєте багатокомпонентний додаток.

# **Підключення зовнішніх файлів стилів та шаблонів.**

Підключення зовнішніх файлів стилів та шаблонів у Angular - це важлива частина розробки веб-додатків на цьому фреймворку. Це дозволяє відокремити логіку додатку від вигляду та оформлення, що полегшує розробку та підтримку коду.

1. Шаблони (Templates):
   * У Angular для визначення структури сторінок та компонентів використовуються шаблони. Шаблони представляють собою розмітку HTML, де можна використовувати спеціальні директиви та вкладені змінні для відображення динамічного контенту.
   * Шаблони можна визначити прямо в коді компонента за допомогою властивості template або витягнути їх у окремий файл і вказати шлях до нього за допомогою templateUrl у декораторі @Component. Використання окремого файлу дозволяє підтримувати окремі файли для шаблонів та забезпечує більшу чистоту та читабельність коду.
2. Стилі (Styles):
   * Стилі в Angular визначають зовнішній вигляд компонентів та елементів на сторінці. Вони можуть бути задані безпосередньо в коді компонента, використовуючи властивість styles, або витягнуті в окремий файл CSS та вказані за допомогою styleUrls у декораторі @Component.
   * Використання окремого файлу CSS дозволяє застосовувати глобальні стилі до всього додатку, а також полегшує перевикористання стилів у різних компонентах.
3. Модулі (Modules):
   * У Angular компоненти та їх відповідні модулі організовані в окремих модулях. Модулі використовуються для групування функціональності та декларування залежностей, таких як компоненти, сервіси, роути, а також файли стилів та шаблонів.
   * Зовнішні файли стилів та шаблонів можуть бути підключені до модулів за допомогою директив styleUrls та templateUrl у деклараціях компонентів.
4. **Завдання ізоляції:**
   * Angular забезпечує ізоляцію стилів та шаблонів компонентів, щоб уникнути конфліктів та забезпечити безпечну роботу з бібліотеками і фреймворками сторонніх розробників.

Підключення зовнішніх файлів стилів та шаблонів допомагає підвищити читабельність коду, спрощує роботу над додатком у команді розробників і робить розробку більш модульною. Крім того, це дозволяє швидко змінювати вигляд і структуру додатку, не змінюючи логіку, що лежить в основі.

# **ng-content: призначення та використання.**

ng-content - це директива в Angular, яка використовується для передачі та вставки контенту внутрішнього компонента зовнішньому компоненту. Вона дозволяє створювати більш загальні та переиспользуемые компоненти, які можуть приймати та відображати різний контент у внутрішньому вмісті. Ось призначення та використання ng-content:

1. Призначення ng-content:
   * ng-content використовується для створення динамічних та реюзабельних компонентів, які можуть відображати контент, який передається їм ззовні.
   * Вона дозволяє вбудовувати зовнішній контент внутрішнього компонента та розміщувати його в певних місцях шаблону.
2. Використання ng-content:
   * Для використання ng-content ви повинні включити його в шаблон внутрішнього компонента за допомогою тегу <ng-content></ng-content>. Внутрішній контент, переданий ззовні, буде вставлено в місце, де ви розмістили цю директиву у шаблоні внутрішнього компонента.
   * Внутрішній компонент може мати декілька місць для вставки контенту за допомогою багаторазового використання ng-content.

У коді лабораторної роботи використовується ng-content для передачі та вставки контенту внутрішнього компонента ChildComponent зовнішнім компонентом AppComponent. Давайте розглянемо призначення та використання ng-content на цьому прикладі: У ChildComponent ми маємо <ng-content></ng-content>, що є місцем для вставки зовнішнього контенту, переданого через <child-comp> у компоненті AppComponent. У AppComponent ви використовуєте <child-comp>, і ви передаєте <h2>Ласкаво просимо {{name}}!</h2> як зовнішній контент. Цей зовнішній контент вставляється внутрішній ChildComponent через ng-content.

При запуску додатка, ви побачите, що текст <h2>Ласкаво просимо {{name}}!</h2> з компонента AppComponent буде вставлений в <ng-content></ng-content> у ChildComponent, і виведено як "Привіт Петро" (де Петро - значення з компонента AppComponent). Таким чином, ви можете передавати та вставляти зовнішній контент внутрішнім компонентам, що дозволяє створювати більш змістовні та переиспользуемі компоненти.

# **Взаємодія між компонентами. Передача даних у дочірній компонент.**

Взаємодія між компонентами в Angular - це важливий аспект розробки, який дозволяє передавати дані від одного компонента до іншого. Одним із способів взаємодії є передача даних у дочірній компонент. Давайте розглянемо приклад передачі даних від батьківського компонента до дочірнього:

Спочатку, у батьківському компоненті (наприклад, AppComponent), нам потрібно передати дані. Це можна зробити за допомогою властивостей та ввести їх у шаблон батьківського компонента.

У ChildComponent ми використовуємо декоратор @Input() для вказівки, що властивість childData приймає дані з батьківського компонента. Тепер дані, передані від батьківського компонента, можуть бути відображені в шаблоні дочірнього компонента.

При запуску додатку, ми побачемо, що значення name з батьківського компонента AppComponent передається та відображається в дочірньому компоненті ChildComponent. Цей механізм дозволяє створювати компоненти, які спільно взаємодіють та передають дані один одному в Angular додатку.

# **Прив'язка до сетера.**

Прив'язка до сетера (setter) - це спосіб встановлення значення властивості об'єкта в мові програмування TypeScript та Angular, який спрацьовує при спробі присвоїти значення цій властивості. Сетери дозволяють виконувати певний код або логіку, коли значення властивості змінюється.

У нашому випадку використовується компонент ChildComponent, який має дві властивості userName і userAge, і використовується сетер для властивості userAge для обмеження віку користувача в межах від 0 до 100.

1. Є дві властивості: userName та \_userAge. userName - це властивість, яку можна передати зовнішнім компонентом через вхідний параметр (Input). \_userAge - приватна властивість для зберігання віку користувача.
2. Є сетер userAge, який встановлює значення \_userAge. У цьому сетері виконується перевірка віку, і якщо вік менший за 0, то \_userAge встановлюється на 0, якщо вік більший за 100, то \_userAge встановлюється на 100. В іншому випадку, значення \_userAge встановлюється на переданий вік.
3. Гетер userAge використовується для отримання значення \_userAge.

За допомогою цього сетера можна забезпечити обмеження діапазону віку користувача, що передається в компонент ChildComponent. Наприклад, якщо вхідний параметр userAge зовнішнього компонента встановлюється на значення -5, то у дочірньому компоненті \_userAge буде автоматично встановлено на 0 через обробку сетером.

# **Прив'язка до подій дочірнього компонента.**

У нашому прикладі ви встановлюєте прив'язку до подій дочірнього компонента ChildComponent і обробляєте ці події в батьківському компоненті AppComponent.

У батьківському компоненті AppComponent ви використовуєте подію (onChanged) на компоненті child-comp. При зміні цієї події виконується метод onChanged, який збільшує або зменшує значення clicks в залежності від переданого значення.

У дочірньому компоненті ChildComponent ви використовуєте @Output() для створення емітера onChanged, який відправляє дані при кліку на кнопках "+" або "-".

Кнопки в темплейті мають (click)="change(true)" та (click)="change(false)", що вказує на виклик методу change з параметром true або false в залежності від кнопки, на яку було натиснуто.

Отже, коли ми натискаємо кнопку "+" або "-", подія (onChanged) відправляється з дочірнього компонента до батьківського, і метод onChanged батьківського компонента AppComponent виконується з відповідним значенням. В результаті, кількість кліків clicks змінюється відповідно.

# **Двостороння прив'язка.**

У нашому прикладі використовується двостороння прив'язка для обрання та оновлення значення userName. У дочірньому компоненті ChildComponent ви використовуєте [ngModel]="userName" для відображення поточного значення userName в полі вводу.

(ngModelChange)="onNameChange($event)" - це подія, яка спрацьовує, коли значення поля вводу змінюється. В метод onNameChange передається нове значення.

У батьківському компоненті AppComponent ви використовуєте [(userName)]="name" для зв'язку name з властивістю userName дочірнього компонента ChildComponent. Ця двостороння прив'язка означає, що зміни, внесені у дочірньому компоненті, автоматично оновлюють значення name у батьківському компоненті і навпаки.

Коли ми вводимо або змінюємо значення у полі вводу дочірнього компонента, зміни відображаються в обраному імені (name) у батьківському компоненті AppComponent, і навпаки. Ця двостороння прив'язка дозволяє легко синхронізувати дані між батьківським та дочірнім компонентами без необхідності вручну слідкувати за змінами та оновлювати дані.

# **Життєвий цикл компонента**

Життєвий цикл компонента в Angular дозволяє вам контролювати та виконувати код на різних етапах життєвого циклу компонента. Наш дочірній компонент ChildComponent реалізує кілька методів життєвого циклу та логує їхнє виконання.

У нашому дочірньому компоненті ChildComponent використовуються наступні методи життєвого циклу:

* ngOnInit: Цей метод виконується після ініціалізації компонента.
* ngOnChanges: Цей метод викликається, коли вхідні дані компонента змінюються.
* ngDoCheck: Цей метод викликається при кожній зміні в компоненті або в його дочірніх компонентах.
* ngAfterViewInit: Цей метод викликається після ініціалізації представлення компонента та після відображення першого представлення.
* ngAfterViewChecked: Цей метод викликається після кожного поновлення представлення компонента.
* ngAfterContentInit: Цей метод викликається після ініціалізації вмісту дочірнього компонента.
* ngAfterContentChecked: Цей метод викликається після кожного поновлення вмісту дочірнього компонента.

У нас є логування для кожного з цих методів, і номер ітерації (this.count) використовується для слідкування за порядком виклику цих методів. Це допоможе вам розуміти, як працює життєвий цикл компонента та в якому порядку виконуються різні методи під час роботи нашого додатку.

# **Висновки**

В даному завданні був розглянутий звіт по роботі, який містив докладний опис різних аспектів розробки Angular-додатків, зокрема стилів та шаблонів компонента, селектора "host", підключення зовнішніх файлів стилів та шаблонів, використання ng-content, взаємодію між компонентами, передачу даних у дочірній компонент, прив'язку до сетера, прив'язку до подій дочірнього компонента, двосторонню прив'язку та життєвий цикл компоненту.

Крім того, було вказано про розгортання двох Angular-додатків Components1 та Components5 на платформі Firebase.

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**

* + 1. Introduction to the Angular docs [Електронний ресурс] – Режим доступу до ресурсу: <https://angular.io/docs>.
    2. Angular. Angular tutorials [Електронний ресурс] / Angular – Режим доступу до ресурсу: <https://angular.io/tutorial>.
    3. The Angular Book [Електронний ресурс] – Режим доступу до ресурсу: <https://angular-book.dev/>.
    4. Daniel Schmitz. Angular 14 from Scratch / Daniel Schmitz., 2022.
    5. Denys Vuika. Developing with Angular / Denys Vuika., 2018.