Лабораторна робота № 5

Робота із компонентами.

Додамо до проекту власну компоненту. Це зручніше роботи, використовуючи утиліту командного рядку Angular CLI, однак перший раз зробимо це вручну. Для цього в директорії src/app створимо папку my, а у ній – файл my.component.ts (рис. 1).

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

@Component({
    selector: 'app-my',
    template: `<div>
        <p>This is my component</p>
    </div>`,
    styles: [`div {
        border: dashed 1px black;
    }
    p {
        margin: 15px;
    }`]
})
export class MyComponent {

}
```

Рис. 1. Створення папки та TypeScript файлу компоненти my.

У файлі створюємо експортний клас MyComponent (експортний для можливості його використання за межами файлу). Далі за допомогою import підключаємо декоратор Component та визначаємо з його допомогою selector – ім'я html тегу нашого компонента, template – шаблон для його відображення, та styles – css для стилізації його елементів. Однак цього поки що недостатньо, навівши мишею та назву класу MyComponent можемо побачити повідомлення, що даний компонент не підключено у жодному із модулів (рис. 2).

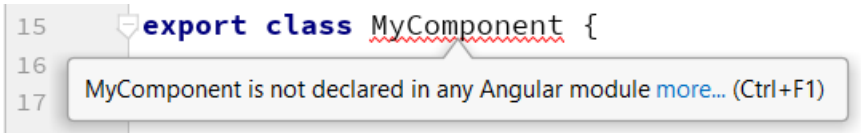

Рис. 2. Помилка у my.component.ts.

Виправимо це. Відкриємо модуль app.module.ts та додамо туди рядок імпорту MyComponent та додамо MyComponent до масиву declarations (рис. 3).

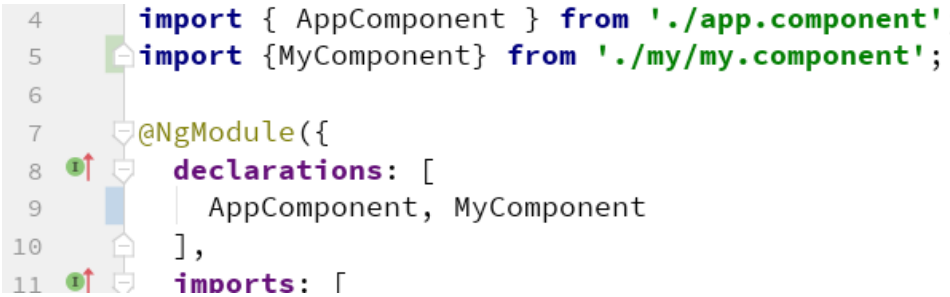

Рис. 3. Додавання MyComponent до declarations у app.module.ts.

Тепер виведемо наш компонент на сторінку. Для цього відкриємо файл app.component.html та додамо до нього рядок <app-my></app-my> (рис. 4).

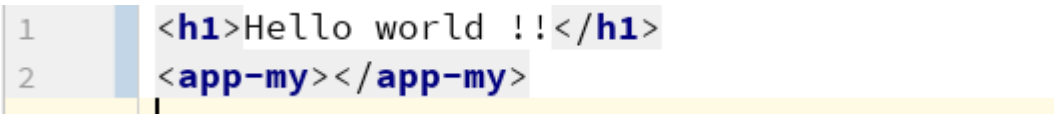

Рис. 4. Змінений app.component.html.

Переглянемо результат додавання нового компонента у браузері (рис. 5).

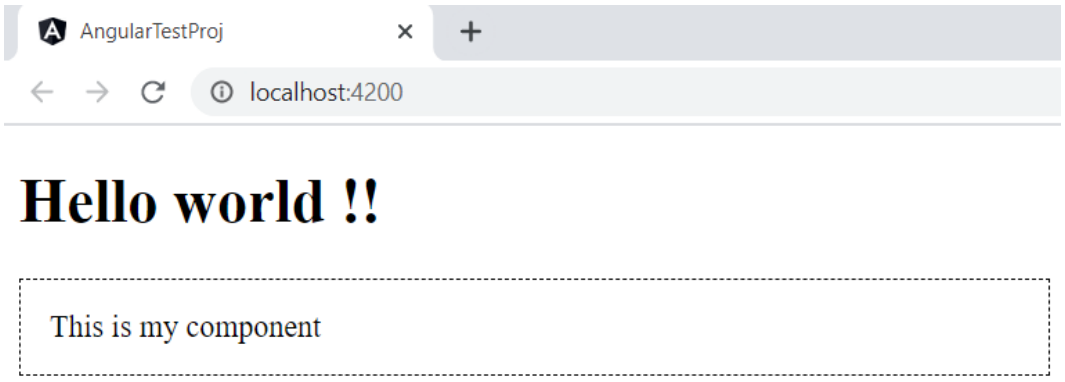

Рис. 5. Зміни на сторінці застосунку.

Повернімося до файлу my.component.ts. Зараз у ньому представлений і клас MyComponent, і html-шаблон компонента і його таблиця стилів css. Даний варіант

підійде лише для дуже простих компонент, html та css код яких є дуже малим. Але навіть у цьому випадку від виглядає досить непривабливо, оскільки поєднує код 3-х різних мов. Розділимо html, css та TypeScript у різні файли. Для цього у директорії scr/app/my створимо файли my.component.html та my.component.scss наступного вмісту (рис. 6-7).

```
<div>
    <h3>This is my component</h3>
    <p>version 0.0.2</p>
</div>
```

Рис. 6. Файл my.component.html.

```
div {
  border: dashed 1px black;
  padding: 15px;
  h3 {
    color: blue;
    font-weight: bold;
  }
  p {
    margin: 0px;
  }
}
```

Рис. 7. Файл my.component.cscc.

Також виконаємо зміни у my.component.ts – замість наведення html та css коду виконаємо підключення створених my.component.html та my.component.scss (рис. 8).

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

@Component({
    selector: 'app-my',
    templateUrl: 'my.component.html',
    styleUrls: ['my.component.scss']
})
export class MyComponent {

}
```

Рис. 8. Зміни у my.component.ts.

Переключимось у браузер та переглянемо результат виконаних змін (рис. 9).

Рис. 9. Перегляд виконаних змін.

Наступний компонент group створимо, використавши засоби командного рядку Angular CLI. Для цього виконаємо команду ng generate component group --skipTests, або її коротку форму ng g c group --skipTests (рис. 10).

```
C:\data\personal\michael\MOHYLA\Education\SPA_PWA\code\angular-test-proj>ng g c group --skipTests
CREATE src/app/group/group.component.html (20 bytes)
CREATE src/app/group/group.component.ts (266 bytes)
CREATE src/app/group/group.component.scss (0 bytes)
UPDATE src/app/app.module.ts (448 bytes)
```

Рис. 10. Створення компонента засобами Angular CLI.

У цей раз Angular CLI автоматично створює файли для html, css та typescript, а також виконує підключення нового компоненту у app.module.ts. Відкриємо group.component.html та змінимо його (рис. 11).

```
1  <div class="group">
2      <h3>Група 301</h3>
3      <p class="faculty">факультет комп'ютерних наук</p>
4      <p class="specialty">спеціальність: комп'ютерні науки</p>
5  </div>
```

Рис. 11. Зміни group.component.html.

Також додаємо стилізацію для нового компонента (рис. 12).

```scss
div.group {
  padding: 20px;
  background: linen;
  border-radius: 5px;
  h3 {
    margin-top: 0px;
  }

  p.faculty {
    color: blue;
  }
  p.specialty {
    font-size: 0.8em;
    color: brown;
  }
}
```

Рис. 12. Файл group.component.scss.

Додаємо виведення компонента в app.component.html (рис. 13) та переглядаємо результат (рис. 14).

```html
<h1>Hello world !!</h1>
<app-group></app-group>
```

Рис. 13. Додавання app-group до app.component.html.

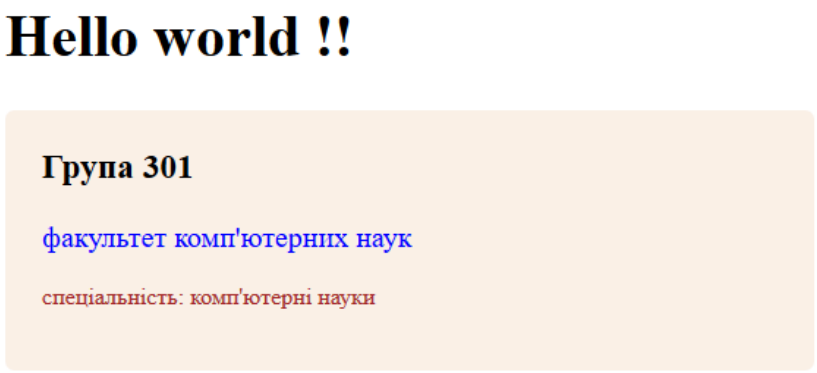

Рис. 14. Перегляд результату.

Додамо нашому компоненту трохи динаміки. Реалізуємо у класі GroupComponent 3 поля – номер групи, назву факультету та спеціальності (рис. 15).

```
8   export class GroupComponent implements OnInit {
9
10    number = '308';
11    faculty = `факультет комп'ютерних наук`;
12    specialty = `інженерія програмного забезпечення`;
13
14    constructor() { }
15
```

Рис. 15. Додавання змінних до класу GroupComponent.

Тепер, використовуючи оператор інтерполяції {{ }}, виконаємо вставку в шаблон змінних класу typescript (рис. 16).

```
1   <div class="group">
2       <h3>Група {{ number }}</h3>
3       <p class="faculty">{{ faculty }}</p>
4       <p class="specialty">спеціальність: {{ specialty }}</p>
5   </div>
```

Рис. 16. Зміни у шаблоні group.component.html.

Після змін, сторінка у браузері матиме наступний вигляд (рис. 17).

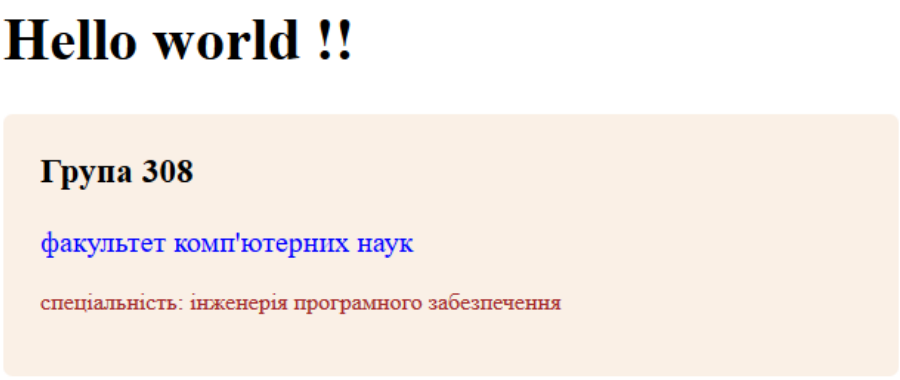

Рис. 17. Сторінка застосунку після змін.

Спробуємо вивести до шаблону значення інших типів. Додамо до класу значення типу число, масив та об’єкт (рис. 18) та виведемо їх на шаблон (рис. 19).

```
    number = '308';
    faculty = `факультет комп'ютерних наук`;
    specialty = `інженерія програмного забезпечення`;

    studentsQuantity = 22;
    students = [
        'Іванов Василь',
        'Смірнова Марія',
        'Максимов Іван'
    ];
    starosta = {
        name: 'Іванов Василь',
        age: 20,
        address: 'Mykolaiv, Ukraine'
    }
```

Рис. 18. Додавання нових полів до класу GroupComponent.

```
<div class="group">
    <h3>Група {{ number }}</h3>
    <p class="faculty">{{ faculty }}</p>
    <p class="specialty">спеціальність: {{ specialty }}</p>
    <p>Кількість студентів: {{studentsQuantity}}</p>
    <p>Склад групи: {{students}}</p>
    <p>Староста: {{starosta}}</p>
</div>
```

Рис. 19. Додавання нових полів у шаблон group.component.html.

Переглядаємо результат (рис. 20).

Група 308

факультет комп'ютерних наук

спеціальність: інженерія програмного забезпечення

Кількість студентів: 22

Склад групи: Іванов Василь,Смірнова Марія,Максимов Іван

Староста: [object Object]

Рис. 20. Результат змін.

Як бачимо, з числом ніяких проблем, масив виводиться аналогічно до array.join(), а от із об’єктом не все так добре. Для виведення об’єкту у json форматі можемо застосувати додатокі можливості при виводі даних, так звані pipe-и (рис. 21). Також використаємо тег pre для більш привабливого вигляду виведеного json-у.

```
6       <p>Склад групи: {{students}}</p>
7       <p>Староста:</p><pre>{{starosta | json}}</pre>
8   </div>
```

Рис. 21. Виведення об’єкту у вигляді json-у.

Тепер поле «староста» виглядає наступним чином (рис. 22).

Староста:


```
{
  "name": "Іванов Василь",
  "age": 20,
  "address": "Mykolaiv, Ukraine"
}
```


Рис. 22. Вигляд поля у json-форматі.

Додамо трохи динаміки – приховаємо назву факультету та спеціальність через 2 секунди після завантаження сторінки. Для цього у метод ngOnInit розмістимо наступний код (рис. 23).

```
28  ngOnInit() {
29      setTimeout( handler: () => {
30          this.faculty = '';
31          this.specialty = '';
32      }, timeout: 2000);
```

Рис. 23. Зміна значень полів класу.

Переглянемо результат (рис. 24).

Група 308

факультет комп'ютерних наук

спеціальність: інженерія програмного забезпечення

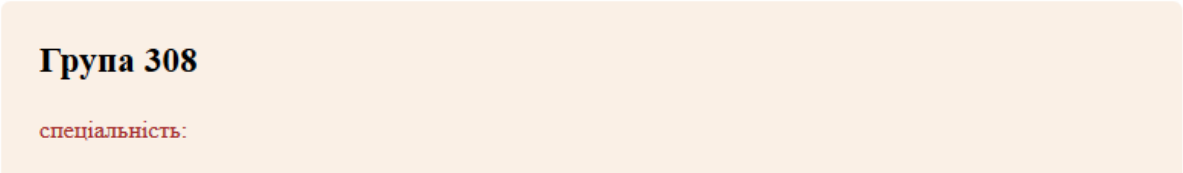

Рис. 24. Результат зміни значень полів.

Для передачі змін із класу в компонент та у зворотньому напрямку в Angular є спеціальний механізм, що має назву двостороннє зв'язування. Атрибут src тегу img буде взято у квадратні дужки, що означає передачу значення із typescript коду класу GroupComponent до html шаблону. Отже, реалізуємо картинку, що буде змінюватися кожні декілька секунд (рис. 25-27) та переглянемо отриманий результат.

```
images = [
  'https://image.mel.fm/i/I/IoT9VfRJLP/590.png',
  'https://kartinki-vernisazh.ru/_ph/478/2/550191551.jpg?1577384607',
  'https://i.pinimg.com/474x/7d/14/ba/7d14bac4b8ba2ff32151088ed0c020d6.jpg'
];
curImageIndex = 0;
curImage: string;

constructor() { }

ngOnInit() {
    this.curImage = this.images[this.curImageIndex];
    setInterval( callback: () => {
        this.curImageIndex++;

        if (this.curImageIndex >= this.images.length) {
            this.curImageIndex = 0;
        }
        this.curImage = this.images[this.curImageIndex];
    }, ms: 2000);
```

Рис. 25. Зміни у group.component.ts.

```
<div class="group">
    <img [src]="curImage">
    <h3>Група {{ number }}</h3>
```

Рис. 26. Зміни у group.component.html.

```
img {
  width: 300px;
  margin-bottom: 20px;
}
```

Рис. 27. Ширина зображення у group.component.scss.