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 5

Робота з директивами. Атрибутивні та структурні директиви.

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 ініціалізовано angular проєкти Directive1-6
2. Було опрацьовано завдання згідно з методичними вказівками
3. Було розгорнуто дотатки Directive1 та Directive2 на платформі Firebase

1) Директиви: призначення, приклади використання

Директиви в Angular є однією з ключових особливостей цього фреймворку, що дозволяють маніпулювати DOM елементами та додавати до них динамічну поведінку. Існує три основних типи директив в Angular:

1. Структурні директиви: Ці директиви змінюють структуру DOM, додаванням, видаленням чи заміною елементів. Наприклад, *ngFor для ітерації елементів масиву або *ngIf для умовного відображення елементів.
2. Атрибутивні директиви: Вони змінюють вигляд або поведінку DOM елементів. Наприклад, ngStyle дозволяє динамічно змінювати стилі елементів, а ngClass дозволяє додавати або видаляти CSS класи.

Приклади Використання

1. ngFor:

```
<ul>
    <li *ngFor="let item of items">{{ item }}</li>
</ul>
```

Цей код створює список li елементів для кожного елемента в масиві items.

2. ngIf:

```
<div *ngIf="isVisible">Цей блок відображається, якщо isVisible = true</div>
```

Цей блок відображається лише тоді, коли змінна isVisible має значення true.

3. ngStyle:

<div [ngStyle]="{'font-size': fontSize + 'px'}">Текст з динамічним розміром шрифту

Тут розмір шрифту елемента змінюється динамічно залежно від значення змінної fontSize.

4. ngClass:

   <div [ngClass]="{'active': isActive, 'disabled': !isActive}">Цей блок має класи 'active' або 'disabled'

Класи active або disabled будуть додані до цього елемента залежно від стану змінної isActive.

Використання цих директив значно спрощує роботу з динамічним контентом і структурою веб-сторінок у Angular, роблячи код більш читабельним та ефективним.

2) Огляд атрибутивних директивів

Атрибутивні директиви в Angular є потужним інструментом для зміни вигляду і поведінки DOM елементів. Вони дозволяють додавати або модифікувати атрибути, стилі, класи та інші властивості елементів без зміни їхньої структури. Ось декілька ключових прикладів атрибутивних директив в Angular:

Стандартні атрибутивні директиви

Використання ngClass

1. Присвоєння одного класу за умовою:

<div [ngClass]="{verdanaFont:isVerdana}">

Тут клас verdanaFont буде застосовано до div, якщо isVerdana має значення true.

2. Присвоєння класу іншому елементу за умовою:

<p [ngClass]="{segoePrintFont:isSegoe}">

Аналогічно, клас segoePrintFont буде застосовано до параграфа p, якщо isSegoe має значення true.

3. Застосування декількох класів одночасно:

<div [ngClass]="currentClasses">

Тут currentClasses - це об'єкт, що визначає кілька класів (verdanaFont і navyColor), які будуть застосовані, якщо відповідні властивості (isVerdana, isNavy) мають значення true.

4. Використання ngStyle

<div [ngStyle]="{'font-size':'13px', 'font-family':'Verdana', 'color':'green'}">

Тут ngStyle застосовує інлайнові стилі безпосередньо до div елемента. Це статичне присвоєння стилів, яке не залежить від логічних змінних.

5. Комбінація ngClass та ngStyle

<div [ngClass]="{invisible: visibility}">

Ця частина коду приховує div (через застосування класу invisible, який встановлює display:none), якщо visibility має значення true.

Створення власних атрибутивних директивів.

1. BoldDirective

Ця директива надає елементу жирний шрифт. Вона використовує ElementRef для доступу до DOM елемента та зміни стилю fontWeight.

2. HostmouseboldDirective

Ця директива змінює вагу шрифту елемента на жирний при наведенні курсора миші та повертає до звичайного, коли курсор покидає елемент. Вона використовує Renderer2 для безпечної роботи з DOM та HostListener для прослуховування подій.

3. ItalicDirective

Ця директива робить шрифт елемента курсивним. Подібно до попередньої, вона використовує Renderer2 для зміни стилів.

4. MouseboldDirective

Ця директива є альтернативним варіантом HostmouseboldDirective. Вона використовує декоратор HostListener для реагування на події миші.

5. MouseitalicDirective

Ця директива використовує HostBinding для прив'язки властивостей стилю до елемента. При наведенні курсору миші, шрифт стає курсивним, і повертається до звичайного, коли курсор покидає елемент.

@Input() декоратори використовуються для отримання значень з батьківських компонентів. Вони задають розміри шрифту (selectedSize і defaultSize).

@HostBinding декоратори використовуються для прив'язки властивостей до стилів хост-елементу (таких як fontSize, fontWeight, і cursor).

@HostListener декоратори використовуються для реагування на події DOM (такі як mouseenter і mouseleave), змінюючи стилі відповідно до подій.

3) Огляд структурних директив

Структурні директиви змінюють розмітку DOM шляхом додавання, видалення або заміни елементів. Ось детальний огляд ключових елементів у вашому прикладі:

Стандартні структурниі директиви

1. ngIf: Використовується для умовного відображення контенту. Наприклад, <p *ngIf="condition">Привіт світ!</p> показує параграф, якщо condition є істиною.
2. ngIf з else блоком: Ви використовуєте це для вибору між двома шаблонами. Якщо condition є істиною, відображається контент перед else, інакше - контент, визначений в <ng-template #unset>.

3. ngFor: Ця директива використовується для створення списку або повторення елементів. Наприклад, <li *ngFor="let item of items">{{item}} створює елемент списку для кожного елемента в масиві items.
4. ngSwitch: Це як умовний оператор, але для візуальних шаблонів. В залежності від значення count, відображається різний контент.
5. Використання індексу в ngFor: Це дозволяє вам отримати доступ до індексу поточного елемента в циклі.

Цей код є гарним прикладом використання структурних директив для управління відображенням контенту в Angular. Він демонструє основні та деякі складніші підходи до умовного рендерингу та списків.

Створення власних структурних директивів.

Створення власної структурної директиви в Angular, як на прикладі з директивою WhileDirective, є хорошим способом розширення функціоналу шаблонів. Директива WhileDirective працює подібно до стандартної директиви ngIf, але з власним синтаксисом. Ось кроки створення та використання цієї директиви:

Створення WhileDirective

Імпорт необхідних класів: Ми імпортували Directive, Input, TemplateRef, і ViewContainerRef з @angular/core. Це стандартні класи Angular, необхідні для створення директив.

Декларація директиви: Ми використовуємо декоратор @Directive, щоб вказати, що наш клас є директивою. Селектор [while] означає, що директива буде застосовуватися до елементів, які мають атрибут while.

Конструктор: У конструкторі ми ініціалізуємо TemplateRef і ViewContainerRef. TemplateRef отримує доступ до шаблону, а ViewContainerRef керує вмістом DOM-елементу, до якого застосовується директива.

@Input з сеттером: Ми визначаємо @Input властивість з сеттером, який викликається, коли властивість змінюється. Якщо умова істинна, createEmbeddedView викликається для відображення шаблону. Якщо умова хибна, clear видаляє вміст з DOM.

Використання WhileDirective

Щоб використовувати цю директиву у нашому Angular додатку:

Додавання директиви до модуля: Спочатку нам потрібно додати WhileDirective до відповідного Angular модуля у списку declarations.

Застосування у шаблонах: Після додавання директиви до модуля, ми можемо використовувати її в шаблонах, додавши атрибут *while="condition" до будь-яких елементів, де condition - це вираз, який повертає булеве значення.

4) Огляд всіх структурних блоків Angular-додатку Directives5. Детальний огляд директиви SumDirective

Angular-додаток складається з декількох ключових структурних блоків:

1. AppComponent (Компонент)

Імпорт і декоратор @Component: Визначає клас AppComponent як компонент Angular з декоратором @Component, що вказує на селектор (HTML-тег), шаблон і стилі.

Шаблон: HTML-код із зв'язуванням даних через ngModel для number1 і number2 і директивою *sum, яка обчислює суму.

Клас AppComponent: Містить властивості number1 і number2, які використовуються у шаблоні.

2. AppModule (Модуль)

Імпорт і декоратор @NgModule: Оприділяє клас AppModule як модуль Angular, який включає декларації компонентів та директив, імпорт інших модулів (наприклад, BrowserModule) і бутстрапінг головного компонента.

Декларації: Визначає компоненти (AppComponent) і директиви (SumDirective), які входять до цього модуля.

Імпорти: Включає інші модулі, такі як BrowserModule.

Бутстрапінг: Вказує, що AppComponent є стартовим компонентом.

3. SumDirective (Директива)

Імпорт і декоратор @Directive: Визначає клас SumDirective як директиву Angular з селектором [sum].

Конструктор: Використовує TemplateRef і ViewContainerRef для управління вмістом DOM.

Властивості та методи:

firstNumber і secondNumber - зберігають значення, передані через @Input.

updateView: Оновлює DOM, додаючи результат суми firstNumber і secondNumber.

Докладний огляд директиви SumDirective, її компонентів та функціональності:

1. Імпорт і Декоратор @Directive:

   @Directive({ selector: '[sum]' }): Це декларатор, який визначає SumDirective як директиву Angular. Селектор [sum] означає, що ця директива буде застосована до будь-якого HTML елемента у шаблоні компонента, який має атрибут sum.

2. Конструктор:

   constructor(private templateRef: TemplateRef<any>, private viewContainer: ViewContainerRef) {}: Конструктор ініціалізує директиву з двома ін'єкціями залежностей - TemplateRef та ViewContainerRef.

   TemplateRef<any>: Використовується для доступу до шаблону, у якому використовується ця директива.

ViewContainerRef: Використовується для маніпулювання вмістом DOM всередині контейнера шаблону.

3. Input властивості:

   @Input() set sumFrom(firstNumber: number) {}: Це setter для властивості sumFrom, який отримує перше число для суми. Коли ця властивість отримує значення, викликається метод updateView().

   @Input() set sumAnd(secondNumber: number) {}: Аналогічно, це setter для властивості sumAnd, який отримує друге число для суми та викликає updateView().

4. Локальні змінні:

   private firstNumber = 0;

   private secondNumber = 0;: Ці змінні зберігають значення чисел, які беруть участь у сумуванні.

5. Метод updateView():

   private updateView() {}: Цей метод відповідає за оновлення вмісту контейнера, в якому використовується директива.

   this.viewContainer.clear();: Видаляє увесь вміст із контейнера.

   this.viewContainer.createEmbeddedView(this.templateRef, { $implicit: this.firstNumber + this.secondNumber });: Створює новий вміст для контейнера, передаючи результат суми як контекст. $implicit означає, що це значення можна буде використовувати безпосередньо у шаблоні, де застосовано директиву.

Ця директива використовується для динамічного обчислення та відображення суми двох чисел у шаблоні компонента. Вона демонструє роботу з динамічним контентом, використовуючи Angular-специфічні концепти, такі як зв'язування даних, директиви та ін'єкція залежностей.

5) Огляд всіх структурних блоків Angular-додатку Directives6. Детальний огляд директиви OtherIfDirective

Angular-додаток включає в себе кілька структурних елементів, які працюють разом для створення функціонального інтерфейсу. Давайте розглянемо кожен з цих блоків:

1. AppComponent (Компонент)

   Декоратор @Component: Визначає AppComponent як компонент Angular з селектором app-root і шаблоном HTML.

   Шаблон:

   Містить кнопку, яка перемикає значення логічної змінної condition.

   Використовує директиву *appOtherIf для відображення або приховування елементів <p> в залежності від стану condition.

   Клас AppComponent:

   Містить властивість condition, яка визначає стан для управління відображенням контенту.

2. AppModule (Модуль)

   Декоратор @NgModule: Окреслює AppModule як модуль Angular, визначаючи компоненти та директиви, що входять до нього, а також імпорти і бутстрапінг.

   Декларації: Включає AppComponent та OtherIfDirective.

   Імпорти: Включає BrowserModule.

   Бутстрапінг: Визначає AppComponent як стартовий компонент додатку.

3. OtherIfDirective (Директива)

   Декоратор @Directive: Визначає OtherIfDirective як директиву з селектором [appOtherIf].

Конструктор: Використовує TemplateRef і ViewContainerRef для маніпулювання DOM.

Логіка директиви:

@Input() set appOtherIf(condition: boolean):

Якщо condition є false і представлення ще не створене, створюється нове представлення (createEmbeddedView).

Якщо condition є true і представлення вже існує, воно видаляється (clear).

Використовується для умовного відображення контенту в шаблоні компонента.

Ці блоки разом утворюють основу Angular-додатку, демонструючи як компоненти, модулі та директиви взаємодіють у Angular для створення динамічного веб-інтерфейсу.

6) Посилання на розгорнуті додатки

Додаток Directives1 розміщений за адресою: https://trofymovip02laba5-1.web.app/

Додаток Directives2 розміщений за адресою: https://trofymovip02laba5-2.web.app/

Висновки

Директиви в Angular демонструють силу компонентної архітектури. Вони допомагають в розробці модульних, повторно використовуваних компонентів, які можна легко інтегрувати в різні частини додатку. Розробляючи директиви, розробники навчаються більш ефективно управляти DOM. Директиви надають потужні інструменти для динамічного змінення властивостей, стилів і поведінки елементів HTML. Директиви підкреслюють принцип декларативного програмування в Angular, дозволяючи розробникам описувати, що має бути зроблено, а не як саме це зробити. Це сприяє чистоті коду і його легкості для читання.

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.