Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт по лабораторній роботі № 2

“Робота з компонентами. Взаємодія між компонентами. Прив’язка до подій дочірнього компоненту. Життєвий цикл компоненту”

з дисципліни: «Реактивне програмування»

Студент: Зусько Владислав Юрійович

Група: ІП-02

Дата захисту роботи:

Викладач: доц. Полупан Юлія Вікторівна

Захищено з оцінкою:

Київ, 2023

Зміст

Стилі та шаблони компонента

Стилі компонента в Angular визначають зовнішній вигляд та оформлення елементів, які належать до цього компонента. Стилі можна встановлювати безпосередньо в шаблоні компонента або в окремому файлі стилів (наприклад, CSS або SCSS), який імпортується в компонент. Angular також підтримує використання локальних стилів, що дозволяє звести до мінімуму вплив стилів одного компонента на інші частини додатку.

Шаблони компонента в Angular визначають структуру та вміст, який відображається при рендерингу компонента на сторінці. Шаблони написані у вигляді HTML-коду з додатковими директивами Angular, які дозволяють вставляти дані компонента у вигляді виразів, контролювати відображення та поведінку, а також виконувати ітерації та умовні вирази.

Крім того, у шаблонах можуть бути використані компоненти, які вбудовуються у вміст іншого компонента за допомогою тегу та відповідної директиви. Це дозволяє створювати багаторівневі та повторно використовувані структури для відображення інформації на сторінці.

Шаблони та стилі компонента об'єднуються для створення комплексних та інтерактивних інтерфейсів в Angular, де дані та логіка відділені від відображення, що сприяє підтримці розділення відповідальностей та забезпечує більшу гнучкість при розробці додатків.

Селектор: host. Призначення та використання

Селектор :host в Angular використовується для визначення стилів і поведінки елемента господаря (host element) компонента. Елемент господар може бути елементом, який оточує ваш компонент в шаблоні. Цей селектор дозволяє вам змінювати вигляд або поведінку самого компонента в контексті стилів.

Основне призначення селектора :host включає:

1. Зміна стилів компонента: ви можете визначити стилі, які будуть застосовані до кореневого елемента вашого компонента. Є можливість встановити фоновий колір, розмір шрифту, відступи тощо.
2. Зміна поведінки компонента: можна використовувати селектор :host для визначення певних поведінкових аспектів компонента. Це може включати в себе визначення подій, які спрацьовують на кореневому елементі компонента, або встановлення атрибутів на кореневому елементі.

Селектор :host дозволяє зберігати стилі та поведінку компонента в одному місці і робить код компонента більш читабельним і підтримуваним.

Підключення зовнішніх файлів стилів та шаблонів

У фреймворку Angular можна підключати зовнішні файли стилів і шаблонів для ваших компонентів за допомогою метаданих styleUrls і templateUrl в описі компонента. Ці метадані дозволяють визначити шляхи до зовнішніх файлів стилів і шаблонів для компонента. Давайте розглянемо кожен з цих атрибутів детальніше. styleUrls:

- styleUrls визначає масив шляхів до зовнішніх файлів стилів, які будуть застосовані до компонента.
- Кожен файл стилів має бути окремим файлом CSS. Всі стилі імпортуються і застосовуються до шаблону компонента.
- Використовується, коли вам потрібно визначити стилі для конкретного компонента або шаблону.

templateUrl:

- templateUrl визначає шлях до зовнішнього файлу шаблону для компонента.
- Шаблон може бути написаний на HTML і містити мітки для вставки даних та директив Angular.
- Використовується, коли вам потрібно розділити HTML-код компонента на окремий файл для полегшення обслуговування та використання шаблону в інших компонентах.

Зовнішні файли стилів і шаблонів роблять код компонента більш структурованим і дозволяють легко управляти структурою і зовнішнім виглядом компонента. Це особливо корисно при роботі з більшими проектами, де кілька розробників можуть працювати над різними частинами додатка.

ng-content: призначення та використання

ng-content - це директива в фреймворку Angular, яка використовується для передачі та відображення контенту між визначеними мітками в шаблоні компонента. ng-content дозволяє вбудовувати контент в компоненти, що розробляються, і створювати більш вгадливі та зручні компоненти для використання. Деталі щодо призначення та використання ng-content:

1. ng-content дозволяє передавати контент між відкриваючим і закриваючим тегами цієї директиви в шаблоні компонента. Весь контент, розташований між мітками <ng-content></ng-content>, буде включений у місце, де розміщено ng-content.
2. Контент, який передається через ng-content, може бути динамічною частиною компонента. Це означає, що ви можете вставляти різний контент у компонент в залежності від того, як ви використовуєте компонент.
3. ng-content дозволяє вставляти зовнішній контент у компоненти. Це особливо корисно, коли ви хочете створювати загальні компоненти, які можуть приймати різний контент в залежності від використання.
4. Ви можете використовувати ng-content для розділення контенту на верхній та нижній. Це означає, що ви можете мати дві різні мітки ng-content у шаблоні компонента, і контент буде вставлятися в різні частини компонента.
5. Крім вставки HTML-контенту, ng-content також може передавати дані з зовнішнього контексту в компонент. Ви можете використовувати атрибути та директиви для цього.

Загалом, ng-content робить компоненти більш гнучкими та можливою зміну їх зовнішнього та внутрішнього вмісту в залежності від потреб вашого додатка.

Ця функціональність корисна, коли ви створюєте загальні компоненти, які можуть використовуватися в різних контекстах і з різними видами контенту.

Взаємодія між компонентами. Передача даних у дочірній компонент

Взаємодія між компонентами та передача даних у дочірній компонент є важливою частиною розробки в Angular. Це дозволяє створювати багатокомпонентні додатки та передавати дані між різними частинами додатка. Основні способи передачі даних включають наступне:

1. Можна передати дані в дочірній компонент через властивості (properties). Це означає, що ви додаєте властивості до компонента-дитини, і ці властивості можуть бути використані у шаблоні дочірнього компонента. Властивості можуть передавати будь-який тип даних.
2. Можна використовувати вхідні властивості для передачі даних з батьківського компонента до дочірнього. Вони визначаються за допомогою декоратора @Input() і дозволяють батьківському компоненту передавати дані в дочірній компонент через прив'язку до властивості.
3. Є змога використовувати сервіси для обміну даними між компонентами. Сервіси - це спеціальні класи, які можуть бути внедрені в компоненти і надають зручний спосіб обміну даними між компонентами, незалежно від їх ієрархії.
4. Доступно створювати вихідні події за допомогою декоратора @Output() в дочірньому компоненті і висилати їх до батьківського компонента. Це дозволяє передавати дані вгору по ієрархії компонентів.
5. Ви можете використовувати бібліотеку RxJS для створення спостережуваних об'єктів (Observables) і передавати дані асинхронно між компонентами. Це особливо корисно для складних сценаріїв обміну даними.

Залежно від конкретної задачі та архітектури додатку, можна вибрати один або кілька з цих способів взаємодії між компонентами та передачі даних до дочірніх компонентів.

Прив'язка до сетера

Прив'язка до сетера (setter binding) в Angular дозволяє вам встановлювати значення властивостей компонента через шаблон, використовуючи сетер, що визначений у компоненті. Це корисний механізм для обробки вхідних даних та виконання додаткових дій при встановленні значень властивостей. Основні кроки для використання прив'язки до сетера:

- У компоненті визначте сетер для властивості, яку ви хочете змінювати через прив'язку до сетера. Сетер - це спеціальний метод, ім'я якого співпадає з назвою властивості, і він приймає один параметр - нове значення властивості.
- У вашому шаблоні використовуйте прив'язку до сетера для встановлення значення властивості в компоненті. Використовуйте квадратні дужки [()] разом з ім'ям сетера.
- У сетері можна додатково обробити нове значення перед тим, як воно буде прийнято та збережено в компоненті. Ви можете виконати перевірки, обчислення або інші дії перед збереженням значення.

Прив'язка до сетера дозволяє забезпечити контроль над тим, як дані передаються між компонентами та виконувати додаткові дії, коли значення змінюються. Вона особливо корисна, коли потрібно валідувати або обробляти дані перед їх збереженням.

Прив'язка до подій дочірнього компонента

Прив'язка до подій дочірнього компонента (event binding) в Angular дозволяє вам реагувати на події, які виникають в дочірньому компоненті, і виконувати код у батьківському компоненті на ці події. Основні кроки для використання прив'язки до подій дочірнього компонента включають наступне:

- Визначення події в дочірньому компоненті: у дочірньому компоненті визначте подію за допомогою декоратора @Output(). Цей декоратор дозволяє визначити вихідну подію, яку можна викликати у дочірньому компоненті.
- Генерація події в дочірньому компоненті: можна викликати визначену подію у дочірньому компоненті за допомогою методу, коли відбувається певна дія або подія у дочірньому компоненті.
- Прив'язка події у батьківському компоненті: у шаблоні батьківського компонента можна використовувати прив'язку до подій, щоб підписатися на подію з дочірнього компонента. Ви вказуєте ім'я події, яке було визначено в дочірньому компоненті, і вказуєте, яку функцію викликати, коли подія спрацює.

Прив'язка до подій дочірнього компонента дозволяє створювати взаємодію між компонентами та передавати дані або виконувати дії, коли стається певна подія у дочірньому компоненті. Це особливо корисно при створенні багатокомпонентних додатків і взаємодії між їх різними частинами.

Двостороння прив'язка

Двостороння прив'язка (two-way binding) в Angular дозволяє автоматично оновлювати значення властивостей в компоненті і їх відображення в шаблоні, а також забезпечує автоматичну синхронізацію між властивістю та її значенням у шаблоні. Використання директиви ngModel дозволяє застосовувати двосторонню прив'язку до елементів вводу, таких як тексти, чекбокси і радіокнопки.

Двостороння прив'язка дозволяє створювати інтерактивні і динамічні шаблони, де зміни в одному місці автоматично відображаються в іншому. Це особливо корисно для форм, де ви можете забезпечити автоматичну синхронізацію між вводом користувача і властивістю компонента без необхідності власноручно обробляти всі зміни.

Життєвий цикл компоненту

Життєвий цикл компонента в Angular визначає послідовність подій та методів, які виконуються в процесі створення, оновлення та знищення компонента. Це допомагає контролювати та виконувати дії на різних етапах життєвого циклу компонента. Основні фази життєвого циклу включають такі події та методи:

- Створення (Creation): constructor(): Конструктор компонента викликається при створенні нового екземпляра компонента і виконує початкову ініціалізацію.
- Ініціалізація властивостей (Initialization): ngOnChanges(changes: SimpleChanges): Метод викликається, коли компонент отримує нові значення властивостей, які призначені через прив'язку до властивостей.
- Ініціалізація компонента (Initialization): ngOnInit(): Метод викликається один раз після того, як компонент був створений і властивості ініціалізовані.
- Зміна даних (Data Change): ngDoCheck(): Метод викликається при кожному перевірці змін в компоненті, дозволяючи виконувати власну логіку перевірки.
- Перед показом (Before View Rendered): ngAfterContentInit(): Викликається після ініціалізації контенту, перед відображенням компонента.
- Після показу (After View Rendered): ngAfterViewInit(): Викликається після ініціалізації шаблону і відображення компонента.
- Оновлення (Update): ngOnChanges(changes: SimpleChanges): Якщо значення властивостей змінюються, метод ngOnChanges викликається знову.

- Знищення (Destruction): ngOnDestroy(): Метод викликається перед тим, як компонент буде знищено, і служить для вивільнення ресурсів та скасування підписок.

Користуючись цими методами та подіями, ви можете контролювати та виконувати необхідні дії в різних етапах життєвого циклу компонента для досягнення потрібної функціональності та оптимізації вашого додатка.

Література

1) https://angular.io/
2) https://github.com/angular/angular
3) https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Angular_getting_started