Components

План

- стилі для компонент
- binding
- @Inpput
- @Output
- життєвий цикл компонент
- Практика

Component

- є одним з ключових елементів програми. Компонент управляє відображенням даних на екрані.
- Декоратор @Component як параметр приймає об'єкт з конфігурацією, яка вказує фреймворку, як працювати з компонентом і його даними.
- створення компоненти(cli) `ng g c <name>`

```
import { Component } from '@angular/core';
@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {}
```

selector - значення через яке можна вставити компонент **<app-root></app-root>**

templateUrl - шаблон, який побачить користувач при роботі з додатком

template: `<h1>Hello world</h1>` - може бути у вигляді рядка з html

styleUrls - стилі компоненти

styles: ['h1 { font-weight: normal; }'] - може бути у вигляді рядка

Шаблон

{{ ... }} - текст між фігурними дужками часто називається **властивістю компонента**. Angular замінює це ім'я рядковим значенням відповідної властивості компонента

`<p>{{title}}</p>` - виводить значення title

`<p>The sum of 1 + 1 is {{1 + 1}}.</p>` - вираз

`<p>The sum of 1 + 1 is not {{1 + 1 + getVal()}}.</p>` - виводить значення що повертає функція

Зв’язування даних

Angular підтримує механізм прив'язки, завдяки якому дані в компоненті пов’язані з шаблоном

https://metanit.com/web/angular2/2.5.php

Demo

Типи звязування даних (binding)

- одностороннє (one-way) - дані з компоненти в шаблон

```
{{ title }} - виводить властивість в шаблон

<button [disabled]="isUnchanged">Save</button> - кнопка буде неактивна коли isUnchanged
true

<div [class.special]="isSpecial">Special</div> - div буде містити клас special коли
isSpecial true

<div [ngClass]="{'special': isSpecial}"></div> - div буде містити клас special коли
isSpecial true

<button [style.color]="isSpecial ? 'red' : 'green'"> - кнопка буде червоного кольору якщо
isSpecial true і зеленого якщо false
```

- одностороннє (one-way) - дані з події в шаблоні в компоненту

```
<button (click)="onSave()">Save</button> - викликаэться метод onSave на подію клік

<button (click)="isShow=true">Save</button> - ініціалізовується властивість при кліку

<input [value]="currentItem.name" (input)="currentItem.name=$event.target.value" >
$event - обєкт події з DOM

<app-hero-detail (deleteRequest)="deleteHero()"></app-hero-detail> - подія в компоненті
```

- двостороннє (two-way) - дані обмінюються між шаблоном і компонентою

```
<input [(ngModel)]="name" placeholder="name"> {{ name }} - при редагуванні поля буде мінятися властивость name
```

Змінна(variable) шаблону

Ви можете звернутися до змінної шаблону будь-де в шаблоні

```
<input #phone placeholder="phone number">

<button (click)="callPhone(phone.value)">Call</button>
```

https://metanit.com/web/angular2/2.9.php

Demo

@Input()

Передає дані в компонент за допомогою привязування через властивість

```
<app-user [user]="currentUser" [userId]="currentUserId"></app-user>
```

```
@Input() user; - властивість привязана з оригінальною назвою
```

```
@Input('userId') id; - можна змінити назву властивості в середині компоненти
```

Demo

@Output()

привязка до події дочірнього компонента

```
<app-user (delete)=”deleteUser()”></app-user>

<button (click)="delete(id)">-</button>

@Output() deleteUser= new EventEmitter<number>();

delete(id) {
          this.deleteUser.emit(id);

}
```

@viewChild()

доступ до змінних шаблону **#input** в компоненті, або запитів до html елементів (подібно як document.querySelector)

```
<p #pRef>
  Start editing to see some magic happen :)
</p>

@ViewChild('pRef', {static: false}) pRef: ElementRef;

ngAfterViewInit() {
  console.log(this.pRef.nativeElement.innerHTML);
  this.pRef.nativeElement.innerHTML = "DOM updated successfully!!!";
}
```

Стилі

- Імена класів і селектори є локальними для компонента і не стикаються з класами і селекторами, що використовуються в інших програмах
- Зміни стилів у інших програмах не впливають на стилі компонентів

Спеціальні селектори

:host - єдиний спосіб задати стилі до хост елемента, той в якого огорнутий компонент

:host-context() - селектор шукає клас CSS в будь-якому батьківському класі елемента хоста компонента, аж до кореня документа

```
:host-context(.theme-light) h2 {
  background-color: #eef;
}
```

```
@import './variable.scss'; - щоб використати scss змінні потрібно їх імпортнути
```

Lifecycle (життєвий цикл компоненти)

Angular може створити компоненту, показати, перевірити коли зміняться властивості і зруйнувати її, перш ніж видалити його з DOM.

Angular пропонує функції життєвого циклу, які забезпечують видимість цих ключових моментів життя і здатність діяти, коли вони відбуваються.

Demo

Етапи життєвого циклу:

https://angular.io/guide/lifecycle-hooks#lifecycle-sequence
https://metanit.com/web/angular2/2.8.php

constructor

Викликається перед будь-яким lifecycle hook. Використовуйте його для додавання залежностей, але уникайте будь-якої серйозної роботи тут.

ngOnChanges()

- викликається до методу ngOnInit () при початковій установці властивостей, а також при будь-якій зміні значень
- метод в якості параметра приймає об'єкт класу SimpleChanges, який містить попередні і поточні значення властивості

```
@Input() listCart;
ngOnChanges() {
   console.log(this.listCart)
}
```

ngOnInit

- викликається після методу ngOnChanges ()
- викликається один раз після установки властивостей компонента, які беруть участь в прив'язці. Виконує ініціалізацію компонента

```
test = ‘string’;
ngOnInit() {
   console.log(this.test)
}
```

ngAfterViewInit

Викликається після того, як Angular ініціалізує компоненти, які входять в шаблон поточного компонента

ngOnDestroy

викликається перед тим, як фреймворк Angular видалить компонент.

Практика

Посилання

https://metanit.com/web/angular2/2.1.php

https://angular.io/guide/template-syntax
https://angular.io/guide/lifecycle-hooks
https://angular.io/guide/component-styles

https://angular.io/guide/cheatsheet шпаргалка

Відео

https://learn.javascript.ru/screencast/angular#components-input-data