

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**  
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»**



**Лабораторна робота №5**

**з дисципліни «Реактивне програмування»**

**на тему:**

**«Робота з директивами. Атрибутивні та  
структурні директиви»**

Перевірила  
доцент  
Полупан Ю. В.

Виконав  
студент ФІОТ  
групи ІС-01  
Адамов Денис

**Київ 2023**

## **Зміст**

Зміст .....	2
Лабораторне заняття №5: Робота з директивами. Атрибутивні та структурні директиви .....	3
Атрибутивні директиви .....	4
Структурні директиви .....	6
Directives5 .....	8
Directives6 .....	9

## **Лабораторне заняття №5: Робота з директивами. Атрибутивні та структурні директиви.**

**Мета:** Навчитися створювати та використовувати директиви в Angular.

**Завдання:** Створити чотири Angular-додатки під назвою Directives1, Directives2, Directives3 та Directives4.

I) Для Angular-додатку Directives1 виконати вправу 1;

II) Для Angular-додатку Directives2 виконати вправу 2;

III) Для Angular-додатку Directives3 виконати вправу 3 (виконати самостійне завдання);

IV) для Angular-додатку Directives4 виконати вправу 4 (виконати самостійне завдання).

V) Виконати самостійне завдання зі створенням Angular-додатків Directives5 та Directives6;

VI) Зробити звіт по роботі.

VII) Angular-додатки Directives1 та Directives2 розгорнути на платформі Firebase у проектах з ім'ям «ПрізвищеГрупаLaba5-1» та «ПрізвищеГрупаLaba5-2», наприклад «KovalenkoIP01Laba5-1» та «KovalenkoIP01Laba5-2».

У Angular є три типи директив:

- 1) Компоненти: компонент по суті є директивою, а декоратор `@Component` розширює можливості декоратора `@Directive` за допомогою додавання функціоналу по роботі з шаблонами.
- 2) Атрибутивні: вони змінюють поведінку існуючого елемента, до якого вони застосовуються. Наприклад, `ngModel`, `ngStyle`, `ngClass`
- 3) Структурні: вони змінюють структуру DOM за допомогою додавання, зміни чи видалення елементів HTML. Наприклад, це директиви `ngFor` та `ngIf`.

## Атрибутивні директиви

### `ngClass`

Директива `ngClass` дозволяє визначити набір класів, які застосовуватимуться до елемента. В якості значень вона приймає набір класів у такому вигляді: `[ngClass]={`  
"клас1": true/false,  
"клас2": true/false,  
}

```
<div [ngClass]="{ verdanaFont: isVerdana }">  
  <h1>Hello Angular</h1>  
  <p [ngClass]="{ segoePrintFont: isSegoe }">  
    Angular представляє модульну архітектуру додатку  
  </p>  
</div>
```

---

## Hello Angular

*Angular представляє модульну архітектуру додатку*

### `ngStyle`

Директива `ngStyle` дозволяє встановити набір стилів, які застосовуються до елемента. Як значення директива приймає js-об'єкт, у якому ключі - назви властивостей CSS.

```
<p [ngStyle]="{ 'font-size': '16px', 'font-family': 'Segoe Print' }">  
  Використання атрибутивної директиви [ngStyle]  
</p>
```

## Вивчаємо директиви

*Використання атрибутивної директиви [ngStyle]*

Можна створювати власні атрибутивні директиви

```
lab5 > directives2 > src > app > TS bold.directive.ts > ...  
1  import { Directive, ElementRef } from '@angular/core';  
2  @Directive({  
3    selector: '[bold]',  
4  })  
5  export class BoldDirective {  
6    constructor(private elementRef: ElementRef) {  
7      this.elementRef.nativeElement.style.fontWeight = 'bold';  
8    }  
9  }
```

```
lab5 > directives2 > src > app > TS italic.directive.ts > ...  
1  import { Directive, ElementRef, Renderer2 } from '@angular/core';  
2  
3  @Directive({  
4    selector: '[italic]',  
5  })  
6  export class ItalicDirective {  
7    constructor(private elementRef: ElementRef, private renderer: Renderer2) {  
8      this.renderer.setStyle(  
9        this.elementRef.nativeElement,  
10        'font-style',  
11        'italic'  
12      );  
13    }  
14  }
```

## Вивчаю директиви

### Створення атрибутивних директив

#### *Вивчаю директиви*

### Створення атрибутивних директив

## Структурні директиви

### ngIf

Директива ngIf дозволяє видалити або, навпаки, додати елемент за певної умови.

```
<p *ngIf="condition">Привіт світ!</p>
<p *ngIf="!condition">Пока світ!</p>
<p *ngIf="condition; else unset">Привіт Angular!</p>
<ng-template #unset>
|   <p>Пока Angular!</p>
</ng-template>
<div *ngIf="condition; then thenBlock; else elseBlock"></div>
<ng-template #thenBlock>Then template</ng-template>
<ng-template #elseBlock>Else template</ng-template>
<br />
<button (click)="toggle()">Toggle</button>
```

Привіт світ!

Пока світ!

Привіт Angular!

Пока Angular!

Then template

Else template

Toggle

Toggle

## ngFor

Директива ngFor дозволяє перебрати елементи масиву в шаблоні.

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

```
export class AppComponent {
  condition: boolean = true;
  toggle() {
    this.condition = !this.condition;
  }
  items = ['Tom', 'Bob', 'Sam', 'Bill'];
  count: number = 5;
}
```

- Tom
- Bob
- Sam
- Bill

## ngSwitch

За допомогою директиви ngSwitch можна вбудувати в шаблон конструкцію switch...case та в залежності від її результату виконання виводити той чи інший блок.

```
<div [ngSwitch]="count">
  <ng-template ngSwitchCase="1">{{ count * 10 }}</ng-template>
  <ng-template ngSwitchCase="2">{{ count * 100 }}</ng-template>
  <ng-template ngSwitchDefault>{{ count * 1000 }}</ng-template>
</div>
```

5000



## Directives5

Розробити Angular-додаток Directives5, в якому створити директиву SumDirective для отримання суми двох доданків.

```
lab5 > directives5 > src > app > TS sum.directive.ts > ...
1  import { Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';
2
3  @Directive({
4    selector: '[sum]',
5  })
6  export class SumDirective {
7    constructor(
8      private templateRef: TemplateRef<any>,
9      private viewContainer: ViewContainerRef
10   ) {}
11
12   @Input('sumFrom') from = 0;
13   @Input('sumAnd') and = 0;
14
15   ngOnChanges(): void {
16     this.viewContainer.clear();
17     this.viewContainer.createEmbeddedView(this.templateRef, {
18       $implicit: this.from + this.and,
19     });
20   }
21 }
```

```
lab5 > directives5 > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2  @Component({
3    selector: 'app-root',
4    template: `<label>Введіть a: </label>
5      <input type="number" [(ngModel)]="a" />
6      <br />
7      <label>Введіть b: </label>
8      <input type="number" [(ngModel)]="b" />
9      <hr />
10     <div *sum="let result; from: a; and: b">Сума = {{ result }}</div>`,
11  })
12  export class AppComponent {
13    a = 20;
14    b = 30;
15  }
```



## Directives6

Розробити Angular-додаток Directives6, в якому створити директиву OtherIfDirective. Директива OtherIfDirective робить протилежне NgIf. NgIf відображає вміст шаблону, коли умова дорівнює true. OtherIfDirective повинна відображати вміст, коли умова дорівнює false. Також в шаблоні встановити кнопку <button>, при активізації якої змінюється стан умови condition з false на true і навпаки.

```
lab5 > directives6 > src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2  @Component({
3    selector: 'app-root',
4    template: `<p *OtherIf="condition" class="otherif a">
5      (A) Condition is false.
6    </p>
7    <p *OtherIf="!condition" class="otherif b">
8      (B) Although the condition is true, this paragraph is displayed.
9    </p>
10   <button (click)="toggle()">Toggle</button>
11   <p>Condition: {{ condition }}</p>`,
12  })
13  export class AppComponent {
14    condition: boolean = false;
15    toggle() {
16      this.condition = !this.condition;
17    }
18  }
```

```
lab5 > directives6 > src > app > TS otherIf.directive.ts > ...
1  import { Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';
2
3  @Directive({ selector: '[OtherIf]' })
4  export class OtherIfDirective {
5    constructor(
6      private templateRef: TemplateRef<any>,
7      private viewContainer: ViewContainerRef
8    ) {}
9
10   @Input() set OtherIf(condition: boolean) {
11     if (!condition) {
12       this.viewContainer.createEmbeddedView(this.templateRef);
13     } else {
14       this.viewContainer.clear();
15     }
16   }
17 }
```