Komponenty

@Component

Dekorator @Component wzbogaca klasę o odpowiednie metadane, które pozwalają identyfikować klasy jako komponenty.

Celem metadanych jest poinformowanie jaki selektor reprezentuje komponent oraz wskazanie plików zawierających szablon i style.

```
1   import { Component } from '@angular/core';
    ...
2   @Component({
3     selector: 'app-root',
4     templateUrl: './app.component.html',
5     styleUrls: ['./app.component.scss']
6   })
7   export class AppComponent {
8     title = 'angular';
9   }
10
```

selector	Określa selektor, który reprezentuje komponent. Wspierane są podstawowe selektory CSS, takie jak element, .class, [atrybut] oraz :not().
templateUrl	Ścieżka do pliku zawierającego szablon komponentu. ./app.component.html'
template	Szablon komponentu w postaci ciągu znaków (string). ' Hello world! '
styleUrls	Lista ścieżek do plików zawierających style. ['./app.component.css']
styles	Lista stringów zawierających style. ['a {color: black}']
providers	Lista wstrzykiwanych zależności
viewProviders	Lista wstrzykiwanych zależności z uwzględnieniem ng-content.

Metadane

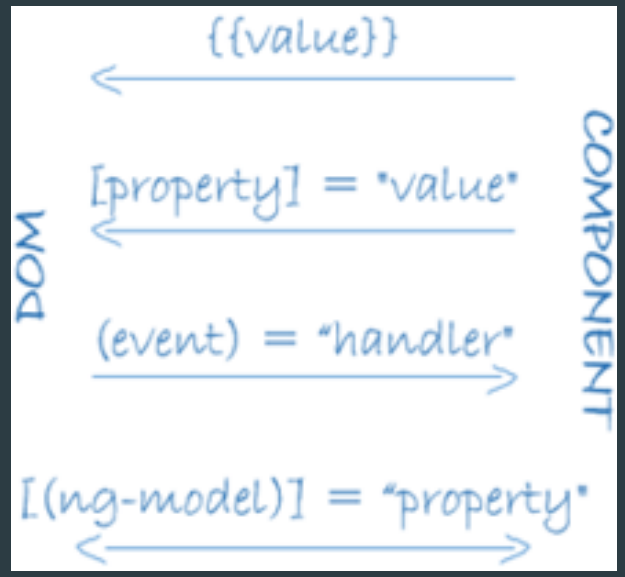
moduleId	Ustawienie parametru na wartość module.id pozwala na korzystanie z relatywnych ścieżek do plików.
changeDetection	Parametr determinujący zachowanie ChangeDetector.
encapsulation	Parametr decydujący o hermetyzacji stylów CSS komponentu.

Metadane

Interakcja z widokiem

Interpolacja	{{,tekst'+model}}
Zdarzenie	(click)=„metoda($event)" https://developer.mozilla.org/en-US/docs/Web/Events
Rotacja klasy elementu	[class.underline]="isUnderlined"
Zarządzanie wieloma klasami	[ngClass]="{ 'underline': isUnderlined, 'active': isActive }"
Styl elementu	[style.width.px]="containerWidth"
Zarządzanie wieloma stylami	[ngStyle]="{ 'text-decoration': isUnderlined ? 'underline' : 'none', 'font-size.px': fontSize }"
Atrybuty	[attr.role]="role"
Wyświetlanie HTML-a	[innerHTML]="htmlContent"
Pętla	<li *ngFor="let element of numList; let i = index"> numList[{{ i }}] = {{ element }}

Binding



Jednokierunkowy, model danych do widoku,	{{expression}} [target]="expression"
Jednokierunkowy, widok do modelu danych,	(target)="statement"
Two-way data binding, wiązanie dwukierunkowe.	[(target)]="expression"

Cykl życia

constructor()	Konstruktor wykorzystujemy niemalże jedynie do wstrzykiwania zależności.
ngOnChanges(change)	Zdarzenie wywoływane przy każdej zmianie składowych @Input()
ngOnInit()	Zdarzenie wywoływane po inicjalizacji składowych @Input(), pierwszym zdarzeniu ngOnChanges().
ngDoCheck()	Zdarzenie wywoływane przy każdorazowej detekcji zmian składowych komponentu.
ngAfterContentInit()	Zdarzenie wywoływane po inicjalizacji ng-content.
ngAfterContentChecked()	Zdarzenie wywoływane po każdorazowej detekcji zmian składowych z ng-content.
ngAfterViewInit()	Zdarzenie wywoływane po inicjalizacji szablonu komponentu (przed ngAfterContentInit()).
ngAfterViewChecked()	Zdarzenie wywoływane po każdorazowej detekcji zmian składowych komponentu.
ngOnDestroy()	Zdarzenie wywoływane przed zniszczeniem instancji komponentu.