4. Direktive

Sadržaj

- Ugrađene direktive
- Specifičnosti strukturnih direktiva
- Kreiranje nove direktive
- Dekoratori `@HostListener` i `@HostBinding`

Direktiva `NgFor`

- Strukturna direktiva - menja strukturu DOM-a
- Ponavlja HTML element po jednom za svaku vrednost u nizu kroz koji prolazi
- Svaki put pri iteraciji prosleđuje aktuelni element niza kao kontekst za string interpolaciju ili *binding*
- Sintaksa

```
*ngFor = “let <vrednost> of <kolekcija>”
```

Indeks petlje

- Ako je potreban indeks elementa kome se pristupa pri iteraciji može se koristiti predefinisana promenljiva `index`

```
<ul *ngFor="let p of players; let i=index"'>
  <li>
    {{i+1}}. Ime: {{p.name}}, Ranking: {{p.rank}}
  </li>
</ul>
```

- Vrednost promenljive `index` ide od 0 do n-1

Grupisani podaci

```
export class PlayersByCountryComponent {

  playersByCountry: any[] = [
    { "country": "Španija", "players": [{"name": "Rafael Nadal","rank": 1}] },
    { "country": "Srbija", "players": [{"name": "Novak Đoković", "rank": 6 },
                                       {"name": "Laslo Đere", "rank": 89}] },
    { "country": "Švajcarska", "players": [{"name": "Rodžer Federer", "rank": 2},
                                           {"name": "Stan Vavrinka", "rank": 101}] }
  ]
```

Prikaz grupisanih podataka

- Podaci koji su grupisani na način prikazan na prethodnom slajdu mogu se prikazati korišćenjem dve ugneždene liste sa odgovarajućim `*ngFor` direktivama

```
<ul *ngFor='let group of playersByCountry'>
  <li>{{ group.country}}</li>
  <ul *ngFor='let player of group.players'>
    <li>{{ player.name }} ({{player.rank}})</li>
  </ul>
</ul>
```

Direktiva `NgIf`

- Koristi se kada se želi zavisno od ispunjenosti nekog uslova prikazati HTML element
- Koristi se sintaksa:
  `*ngIf="<condition>"`
- Ako je vrednost uslovnog izraza `false` element kome je pridružena direktiva će biti uklonjen iz DOM-a
- Postavljanje obeležja hidden na `false` samo sakriva element i ne uklanja ga iz DOM-a, što se razlikuje od efekta primene `NgIf` direktive

Prikaz igrača sa renkingom manjim od 100

- Za *list item* element navodimo uslov na osnovu čije vrednosti će se stavka liste dodati ili ukloniti iz DOM-a
- Pri korišćenju direktiva mora se imati u vidu ograničenje da na jednom HTML elementu ne možemo imati više od jedne strukturne direktive (počinju znakom `*`)
- Jedan od načina da se kombinuju `*ngFor` i `*ngIf` je da se primene na različitim HTML elementima

```
<ul *ngFor='let player of players'>
  <li *ngIf='player.rank<100'>
    Ime: {{player.name}}, Ranking: {{player.rank}}
  </li>
</ul>
```

- Ime: Rafael Nadal, Ranking: 1
- Ime: Novak Đoković, Ranking: 6
- Ime: Laslo Đere, Ranking: 89
- Ime: Rodžer Federer, Ranking: 2

Direktiva `NgSwitch`

- Zavisno od vrednosti izraza navedenog u direktivi prikazaće se HTML element sa `*ngSwitchCase` direktivom kojoj je dodeljena odgovarajuća vrednost

```
<mat-list *ngFor='let player of players' [ngSwitch]="player.country">
  <mat-list-item *ngSwitchCase="'Srbija'" style="color:blue;">
    Ime: {{ player.name}}, Ranking: {{player.rank}}
  </mat-list-item>
  <mat-list-item *ngSwitchCase="'Švajcarska'" style="color:red;">
    Ime: {{ player.name}}, Ranking: {{player.rank}}
  </mat-list-item>
  <mat-list-item *ngSwitchDefault style="color:green;">
    Ime: {{ player.name}}, Ranking: {{player.rank}}
  </mat-list-item>
</mat-list>
```

Ime: Rafael Nadal, Ranking: 1

Ime: Novak Đoković, Ranking: 6

Ime: Laslo Đere, Ranking: 89

Ime: Rodžer Federer, Ranking: 2

Ime: Stan Vavrinka, Ranking: 101

Direktiva NgStyle

- Prosleđuje se objekat koji definiše stil za element
- Vrednost obeležja ne mora biti literal, može biti funkcija koja vraća odgovarajuću vrednost

```
<mat-list *ngFor='let player of players' >
  <mat-list-item  [ngStyle]="{color: getColor(player.country)}">
    Ime: {{player.name}}, Ranking: {{player.rank}}
  </mat-list-item>
</mat-list>
```

```
getColor(country) {
  switch(country) {
    case 'Srbija': return 'blue';
    case 'Švajcarska': return 'red';
    default: return 'green';
  }
}
```

Direktiva `NgClass`

- Omogućava dinamičko postavljanje CSS klase
- Kada se koristi literal koji predstavlja objekat ključevi su klase koji se dodaju elementu ukoliko je vrednost za taj ključ jednaka `true`

```
[ngClass]="{'stil1': someBoolean,
            'stil2': !someBoolean}"
```

ili

```
[ngClass]="{'stil1':player.country==='Srbija',
            'stil2':player.country==='Španija'}"
```

Korišćenje direktive `NgClass`

```
<mat-list *ngFor='let player of players' >
   <mat-list-item [ngClass]="{'serbian-player': player.country == 'Srbija',
                              'swiss-player': player.country == 'Švajcarska',
                              'spanish-player': player.country == 'Španija'}">
                         Ime: {{player.name}}, Ranking: {{player.rank}}
   </mat-list-item>
</mat-list>
```

```
.spanish-player {
  background-color: yellow;
  color: red;
  padding: 10px;
}
.serbian-player {
  background-color: red;
  color: white;
  padding: 10px;
} ...
```

Direktiva `NgNonBindable`

- Koristi se kada želimo da Angular ignoriše određeni sadržaj
- Tipičan primer je kada želimo da ispišemo dvostruke vitičaste zagrade na stranici i želimo da izbegnemo string interpolaciju
- Umesto

```
<div>
  Ispis naziva promenljive <pre>{{name}}</pre>
</div>
```

pisaćemo

```
<div>
  Ispis naziva promenljive <pre ngNonBindable>{{name}}</pre>
</div>
```

Strukturne direktive

- Direktive koje menjaju strukturu DOM-a dodajući ili uklanjajući elemente
  - `NgIf`
  - `NgFor`
  - `NgSwitch`
- Rade koristeći HTML5 tag `<template>`
- Za korišćenje strukturnih direktiva ne moraju se znati detalji implementacije
- *Objašnjenje korišćenja `<template>` taga u implementaciji strukturnih direktiva izlazi van okvira ovog kursa*

Kreiranje direktive

- Komponente su vrsta direktiva koje pored svih osobina direktiva imaju i svoj prikaz (*view*) koji se injektuje u DOM
- Druga razlika je da jedan HTML element može imati pridruženu samo jednu komponentu, dok za direktive to ne važi
- Direktive mogu da slušaju događaje i menjaju vrednost obeležja *host* elementa
- Prilikom definisanja koristi se dekorator `@Directive`

Kreiranje direktive `appHover` (1)

- Direktivu ćemo povezivati sa elementom na sledeći način:
  `<div appHover>...</div>`
- Angular preporučuje da se direktive kao atributi koriste sa prefiksom “app” kako bi se izbegla preklapanja imena
- Klasa direktive ima dekorator `@Directive`

```
import { Directive } from '@angular/core';
…
@Directive({
  selector:"[appHover]"
})
class HoverDirective { … }
```

Kreiranje direktive `appHover` (2)

- Selektor direktive se navodi između znakova `[]` da bi se direktiva na mestima gde je navedena primenila kao atribut HTML elementa
- Unutar definicije klase direktive moramo injektovati element kome će kao atribut biti navedena naša direktiva
- Napravićemo direktivu koja će prilikom prelaska pointera preko komponente menjati boju pozadine na zelenu

```
export class HoverDirective implements OnInit {
  constructor(private elementRef: ElementRef) {
  }
  ngOnInit() {
    this.elementRef.nativeElement.style.backgroundColor='green';
  }
}
```

Kreiranje direktive `dirHover` (3)

- U modulu je potrebno u deklaracijama navesti direktivu koju smo kreirali

```
@NgModule({
  declarations: [
    AppComponent,
    PlayersComponent,
    PlayersByCountryComponent,
    HoverDirective
  ],
  imports: [
    BrowserModule,
    MatListModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Kreiranje direktive pomoću Angular CLI

- Dobra praksa je da se direktive smeštaju u odvojen direktorijum
- Ukoliko ne želimo generisanje `.spec` fajla možemo navesti odgovarajući *switch* (isto je moguće i kod generisanja komponenti):

```
ng g d directives/hover --spec false
```

- Automatski se doda sufiks “Directive” u naziv klase i dekorator
- Podrazumevani prefiks u selektoru je “app”
- Automatski se doda import direktive u modul

Bolji način menjanja stila elementu

- Umesto direktnog pristupa elementu koristiti *renderer*

```
export class HoverDirective implements OnInit {
  constructor(private elementRef: ElementRef,
              private renderer: Renderer2){
  }

  ngOnInit() {
    this.renderer.setStyle(elementRef.nativeElement,
                           'background-color',
                           'green');
  }
}
```

ne DOM obeležje `backgroundColor`

Dekorator `@HostListener`

- Omogućava slušanje događaja nad *host* elementom

```
  ...
  @HostListener('mouseenter') mouseIn(eventData: Event) {
    this.renderer.setStyle(this.elementRef.nativeElement, 'background-color', 'green');
  }

  @HostListener('mouseleave') mouseOut(eventData: Event) {
    this.renderer.setStyle(this.elementRef.nativeElement, 'background-color', 'transparent');
  }
}
```

Dekorator `@HostBinding`

- Omogućava definisanje obeležja prilikom čije promene Angular ažurira *host* element

```
export class HoverDirective implements OnInit {
  @HostBinding('style.backgroundColor') backgroundColor = 'transparent';
  ...
  @HostListener('mouseenter') mouseIn(eventData: Event) {
    this.backgroundColor = 'green';
  }

  @HostListener('mouseleave') mouseOut(eventData: Event) {
    this.backgroundColor = 'transparent';
  }
}
```

Konfigurisanje direktive (1)

- Ukoliko ne želimo da korišćene boje budu *hardcoded*, možemo definisati *input* obeležja za direktivu

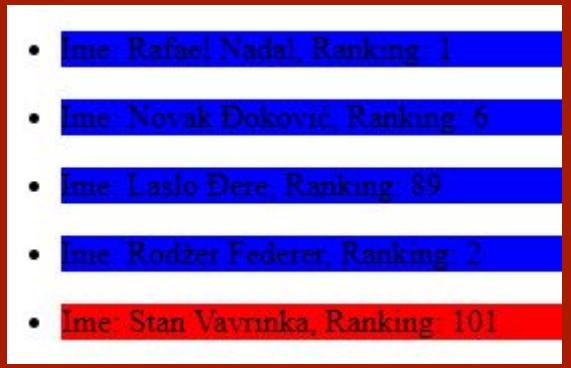
```
export class HoverDirective implements OnInit {
  @Input() defaultColor = 'gray';
  @Input() highlightedColor = 'yellow';
  @HostBinding('style.backgroundColor') backgroundColor;

  constructor(private elementRef: ElementRef, private renderer: Renderer2) {}
  ngOnInit() {
    this.backgroundColor = this.defaultColor;
  }
  @HostListener('mouseenter') mouseIn(eventData: Event) {
    this.backgroundColor = this.highlightedColor;
  }
  @HostListener('mouseleave') mouseOut(eventData: Event) {
    this.backgroundColor = this.defaultColor;
  }
}
```

Konfigurisanje direktive (2)

- Direktivu konfigurišemo postavljanjem obeležja u *template*-u

```
<ul  *ngFor='let group of playersByCountry'>
  <li>{{ group.country}}</li>
  <ul *ngFor='let player of group.players'>
    <li appHover [defaultColor]="'blue'"
                 [highlightedColor]="'red'">{{ player.name }} ({{player.rank}})</li>
  </ul>
</ul>
```



Aliasi *input* obeležja direktive

- Kada se naziv *input* obeležja direktive poklapa sa nazivom atributa elementa za koji se direktiva koristi, koristite se aliasi za *input* obeležja direktive
- Angular omogućava kraće navođenje *input* obeležja direktive i vrednosti za konfigurisanje

```
<li appHover defaultColor="blue" highlightedColor="red">
  Ime: {{player.name}}, Ranking: {{player.rank}}
</li>
```

umesto

```
<li appHover [defaultColor]="'blue'" [highlightedColor]="'red'">
  Ime: {{player.name}}, Ranking: {{player.rank}}
</li>
```

- Imati na umu da su to i dalje obeležja direktive a ne atributi elementa

Rezime

- Direktive se povezuju sa HTML elementom navodeći se kao atributi
- Strukturne direktive menjaju strukturu DOM-a
- Angular omogućava kreiranje novih direktiva
- Dekorator `@HostListener` omogućava slušanje događaja nad *host* elementom
- Dekorator `@HostBinding` omogućava povezivanje obeležja HTML elementa sa obeležjem direktive
- Za pravljenje fleksibilnije direktive moguće je obezbediti konfigurisanje