Angular databinding

Data Binding

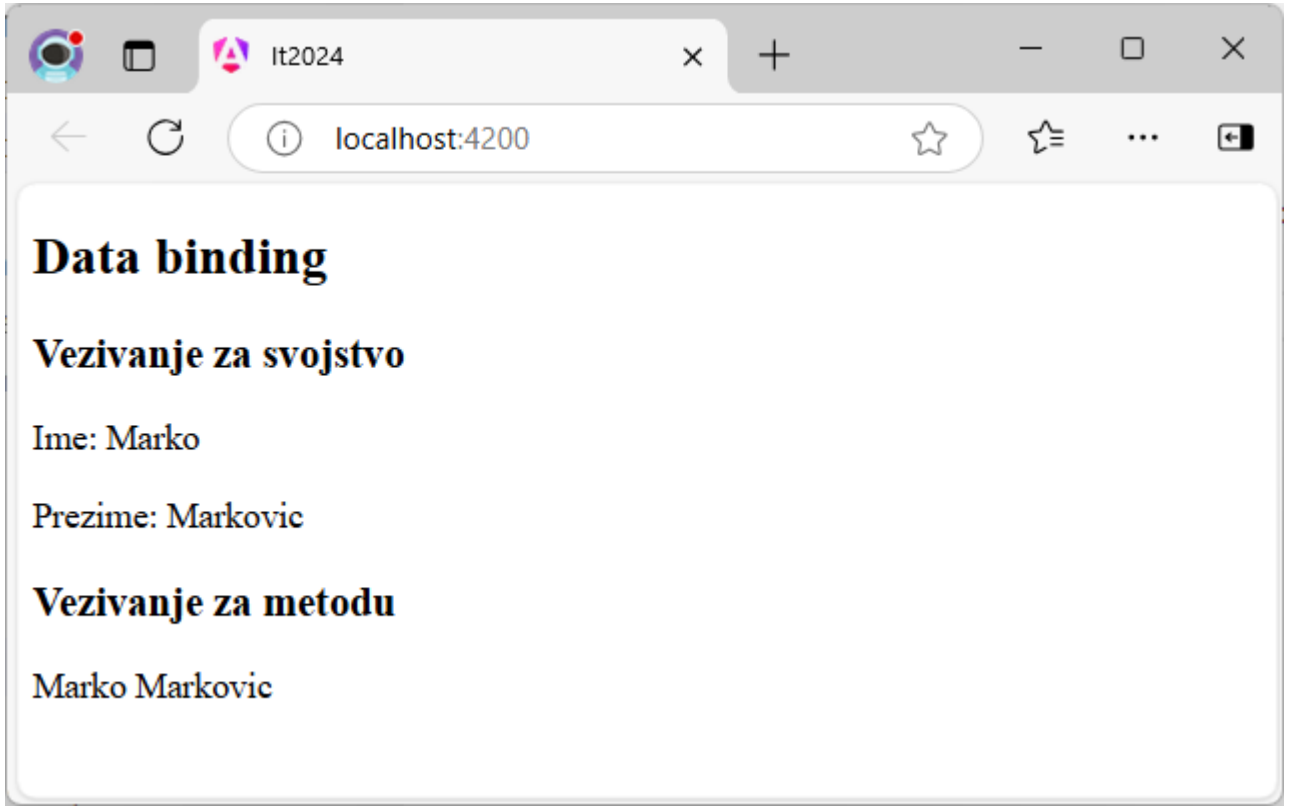
- Data Binding je proces sinhronizacije podataka između klase komponente i njenog pogleda
- Omogućava povezivanje DOM elemenata sa svojstvima klase komponente
- Postoji 4 načina povezivanja podataka u Angularu:
  - Interpolacija
  - Povezivanje svojstva (property binding)
  - Povezivanje događaja (event binding)
  - Dvosmerno povezivanje (two way binding)

Interpolacija -klasa komponente

```
export class App {
  naslov = 'Data binding';
  osoba = {ime: 'Marko', prezime: 'Markovic', grad: 'Beograd'};

  citajIme(): string {
    return this.osoba.ime + ' ' + this.osoba.prezime;
  }
}
```

Interpolacija-pogled

```
<h2>{{ naslov }}</h2>
<h3>Vezivanje za svojstvo</h3>
<p>Ime: {{ osoba.ime }}</p>
<p>Prezime: {{ osoba.prezime }}</p>
<h3>Vezivanje za metodu</h3>
<p>{{ citajIme() }}</p>
```



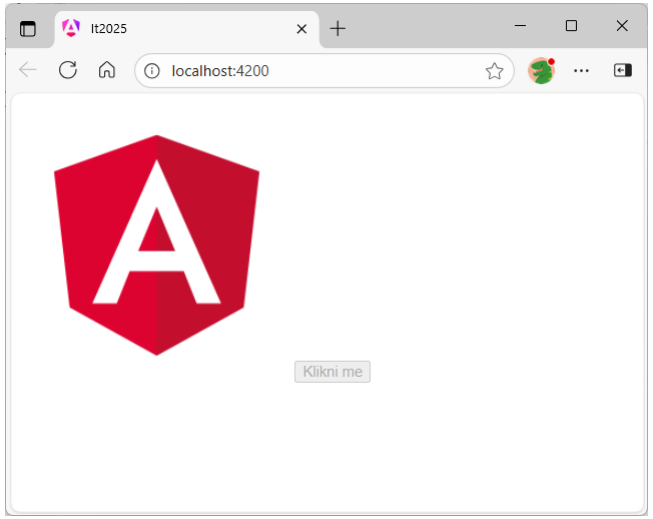
Property binding

- Jednosmerna komunikacija između klase komponente i pogleda
- Podaci se prosleđuju iz klase komponente ka pogledu komponente
- Koristi se kada želimo da postavimo vrednost HTML svojstva (npr. src, href, disabled, itd.) na osnovu podataka iz komponente
- Sintaksa koristi srednje zagrade [ ] oko HTML atributa kako bi se označilo da se vrednost vezuje za neko svojstvo komponente
- Svojstvo komponente čija se vrednost vezuje navodi se unutar navodnika

Property binding

```ts
export class Primer02 {
  slikaUrl = 'https://angular.io/assets/images/logos/angular/angular.png';
  dugmeDisabled = true;
}
```

```html
<!-- Property binding za src -->
<img [src]="slikaUrl" alt="Angular logo" />


<!-- Property binding za disabled -->
<button [disabled]="dugmeDisabled">
  Klikni me
</button>
```



Event binding

- Podaci se prosleđuju iz pogleda komponente ka klasi komponente
- Izvršava se akcija u komponenti kada korisnik pokrene neku radnju u korisničkom interfejsu, npr. klikne na dugme u pogledu
- Ime događaja se stavlja unutar malih zagrada, npr. (click)
- Zatim se događaju dodeljuje funkcija iz komponente

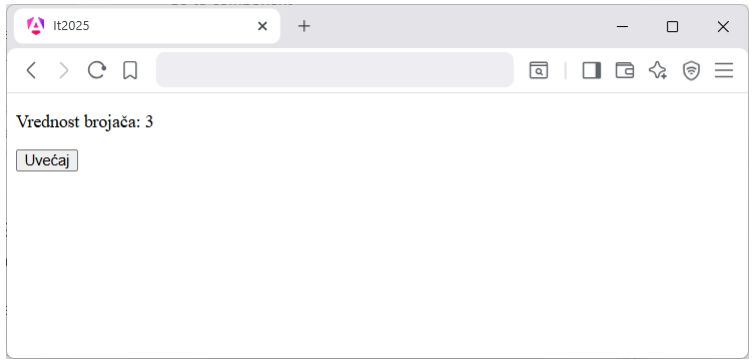
Event binding

klasa komponente:

```
export class Primer03 {
  brojac = 0;

  uvecaj(): void{
    this.brojac++;
  }
}
```

šablon komponente:

```
<p>Vrednost brojača: {{ brojac }}</p>


<button (click)="uvecaj()">Uvećaj</button>
```



Two Way Binding (dvosmerno povezivanje)

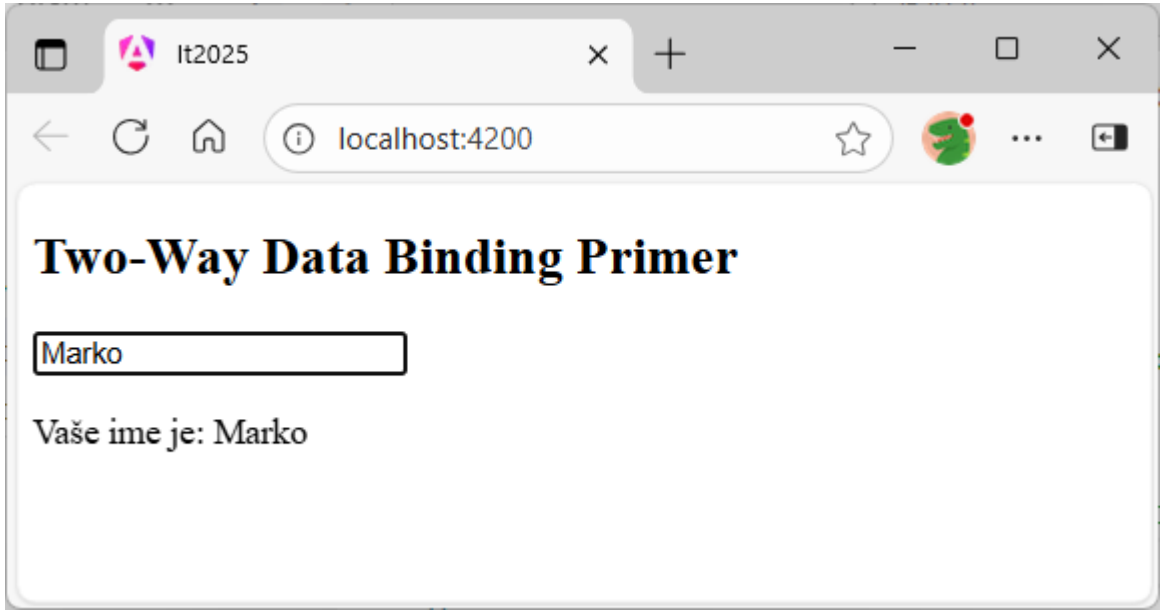
- Promene u klasi komponente se propagiraju ka pogledu
- Promene u pogledu menjaju podatke u klasi komponente
- Dvosmerno povezivanje se koristi kod formi za unos podataka
- Angular koristi kombinaciju povezivanja sa svojstvom (od komponente ka pogledu) i povezivanja sa događajem (od pogleda ka komponenti) da bi ostvario dvosmerno povezivanje
- Koristi se **ngModel** direktiva
- Potrebno je da se u standalone komponentu importuje **FormsModule** da bi se koristitila ngModel direktiva

Klasa komponente

```
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';


@Component({
  selector: 'app-primer04',
  imports: [FormsModule],
  templateUrl: './primer04.html',
  styleUrl: './primer04.css',
})
export class Primer04 {
ime ='';
}
```

Šablon komponente

```
<div>
  <h2>Two-Way Data Binding Primer</h2>
  <input [(ngModel)]="ime" placeholder="Unesite ime">
  <p>Vaše ime je: {{ ime }}</p>
</div>
```



Angular direktive

Referenciranje bootstrap biblioteke u fajlu index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>It2025</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css"
integrity="sha384-xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0nLMwNLD69Npy4HI+N"
crossorigin="anonymous">

</head>
<body>
  <app-root></app-root>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Fy6S3B9q64WdZWQUiU+q4/2Lc9npb8tCaSX9FK7E8HnRr0Jz8D6OP9dO5Vg3Q9ct"
crossorigin="anonymous"></script>
</body>
</html>
```

Angular direktive

- Olakšavaju manipulaciju DOM elementima
- Ako se koristi standalone komponenta, potrebno je importovati **CommonModule** u komponentu da bi se koristile direktive
- Tri tipa direktiva:
  - **Komponente**
  - **Strukturalne directive** (kontrola toka u template-u) dodaju ili uklanjaju DOM elemente:
    - ***ngFor**
    - ***ngIf**
    - ***ngSwitch**
  - **Atributske direktive** menjaju izgled ili ponašanje DOM elementa:
    - **ngModel**
    - **ngClass**
    - **ngStyle**

Nova kontrola toka u angularu

- Angular uvodi novu, moderniju i bržu sintaksu za kontrolu toka u template-ima
- Cilj: bolje performanse, jasniji kod i jednostavnije održavanje
- Stare direktive (*ngIf, *ngFor, ngSwitch) su zastarele (deprecated)
- Nova sintaksa:
  - @for → zamena za *ngFor
  - @if → zamena za *ngIf
  - @switch, @case, @default → zamena za ngSwitch
- Stara sintaksa i dalje radi, ali će biti uklonjena u budućim verzijama
- Preporuka: koristiti novu sintaksu u svim novim projektima

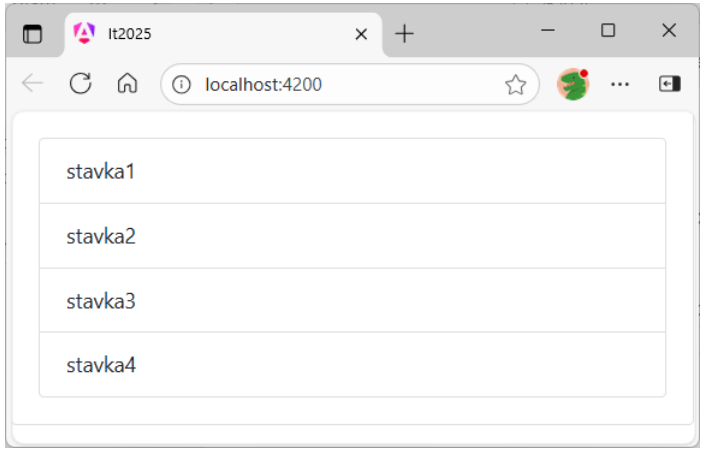
Uključivanje modula CommonModule

```
import { CommonModule } from '@angular/common';
import { Component } from '@angular/core';


@Component({
  selector: 'app-primer05',
  imports: [CommonModule],
  templateUrl: './primer05.html',
  styleUrl: './primer05.css',
})
export class Primer05 {
stavke: string[] = ['stavka1', 'stavka2', 'stavka3', 'stavka4'];


}
```

Direktiva *ngFor

Omogućava kreiranje DOM elemenata za svaki član niza

```
<div class="card">
  <div class="card-body">
    <ul class="list-group">
      <li class="list-group-item" *ngFor="let s of stavke">
        {{ s }}
      </li>
    </ul>
  </div>
</div>
```



Kontrolna struktura @for

```
@for (item of kolekcija; track item) {
  <!-- sadržaj -->
}
```

- @for je nova Angular kontrolna struktura za iteraciju u šablonu
- Za korišćenje kontrolne strukture @for u standalone komponentama potrebno je importovati CommonModule
- Zamenjuje zastarelu direktivu *ngFor
- Deo je novog control flow sistema u Angularu 17+
- track određuje kako Angular prati elemente radi boljih performansi
- Omogućava brži i efikasniji rendering u odnosu na staru sintaksu

Kontrolna struktura @for

```
<div class="card">
  <div class="card-body">
    <ul class="list-group">
      @for (s of stavke; track s) {
        <li class="list-group-item">
          {{ s }}
        </li>
      }
    </ul>
  </div>
</div>
```

Iteracija kroz objekte pomoću @for

```
@for (item of objekti; track item.id) {
  <div>{{ item.naziv }}</div>
}
```

- Kontrolna struktura @for može da iterira kroz niz objekata, ne samo kroz proste tipove
- item predstavlja jedan objekat iz kolekcije
- track item.id koristi jedinstveni identifikator objekta radi optimalnog renderovanja
- Ovo poboljšava performanse kada se objekti dodaju, uklanjaju ili menjaju

Model klasa

```
ng g class proizvod
```

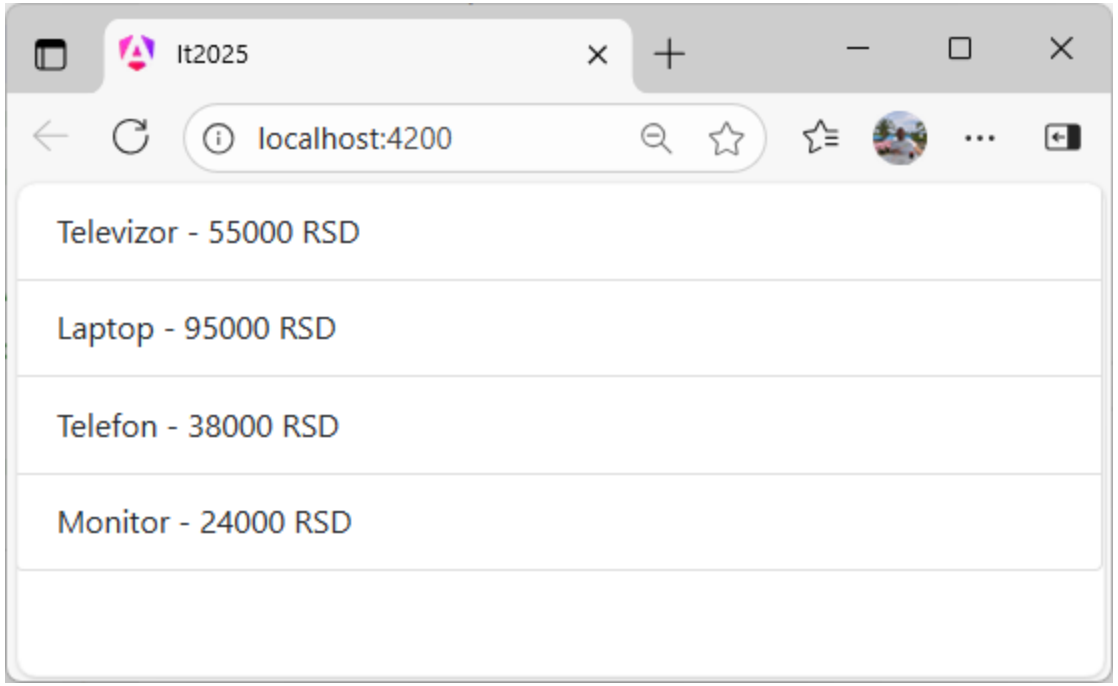
```
export class Proizvod {
    constructor(public id: number, public naziv: string, public cena: number) {}
}
```

Klasa komponente

```
import { Component } from '@angular/core';
import { Proizvod } from '../proizvod';
import { CommonModule } from '@angular/common';
@Component({
  selector: 'app-proizvodi',
  imports: [CommonModule],
  templateUrl: './proizvodi.html',
  styleUrl: './proizvodi.css',
})
export class ProizvodiComponent {
  proizvodi = [
    new Proizvod(1, 'Televizor', 55000),
    new Proizvod(2, 'Laptop', 95000),
    new Proizvod(3, 'Telefon', 38000),
    new Proizvod(4, 'Monitor', 24000)
  ];
}
```

Šablon komponente

```
<ul class="list-group">
  @for (p of proizvodi; track p.id) {
    <li class="list-group-item">{{ p.naziv }} - {{ p.cena }} RSD</li>
  }
</ul>
```



Direktiva *ngIf(zastarela)

Omogućava uslovno prikazivanje elementa.

```
export class Primer06 {
  osoba = {
    ime: 'Marko',
    prezime: 'Markovic',
    starost: 22,
    grad: 'Beograd'
  };
  prikaziDetalje = false;
  dugmeTekst = 'Detalji';

  toggleDetalji() {
    this.prikaziDetalje = !this.prikaziDetalje;
    this.dugmeTekst = this.prikaziDetalje
      ? 'Ukloni detalje'
      : 'Detalji';
  }
}
```

Direktiva *ngIf – šablon komponente

```
<div class="p-3">
  <p><strong>Ime:</strong> {{ osoba.ime }}</p>
  <p><strong>Prezime:</strong> {{ osoba.prezime }}</p>


  <button (click)="toggleDetalji()">
    {{ dugmeTekst }}
  </button>


  <div *ngIf="prikaziDetalje">
    <p><strong>Starost:</strong> {{ osoba.starost }}</p>
    <p><strong>Grad:</strong> {{ osoba.grad }}</p>
  </div>
</div>
```

Kontrolna struktura @if

- @if je nova Angular kontrolna struktura za uslovno prikazivanje elemenata
- Zamenjuje zastarelu direktivu *ngIf
- Deo je novog control flow sistema u Angularu 17+
- Omogućava jasniju i čitljiviju sintaksu u šablonu
- Podržava @else blokove i ugnježdene izraze
- Za korišćenje u standalone komponentama potrebno je importovati **CommonModule**

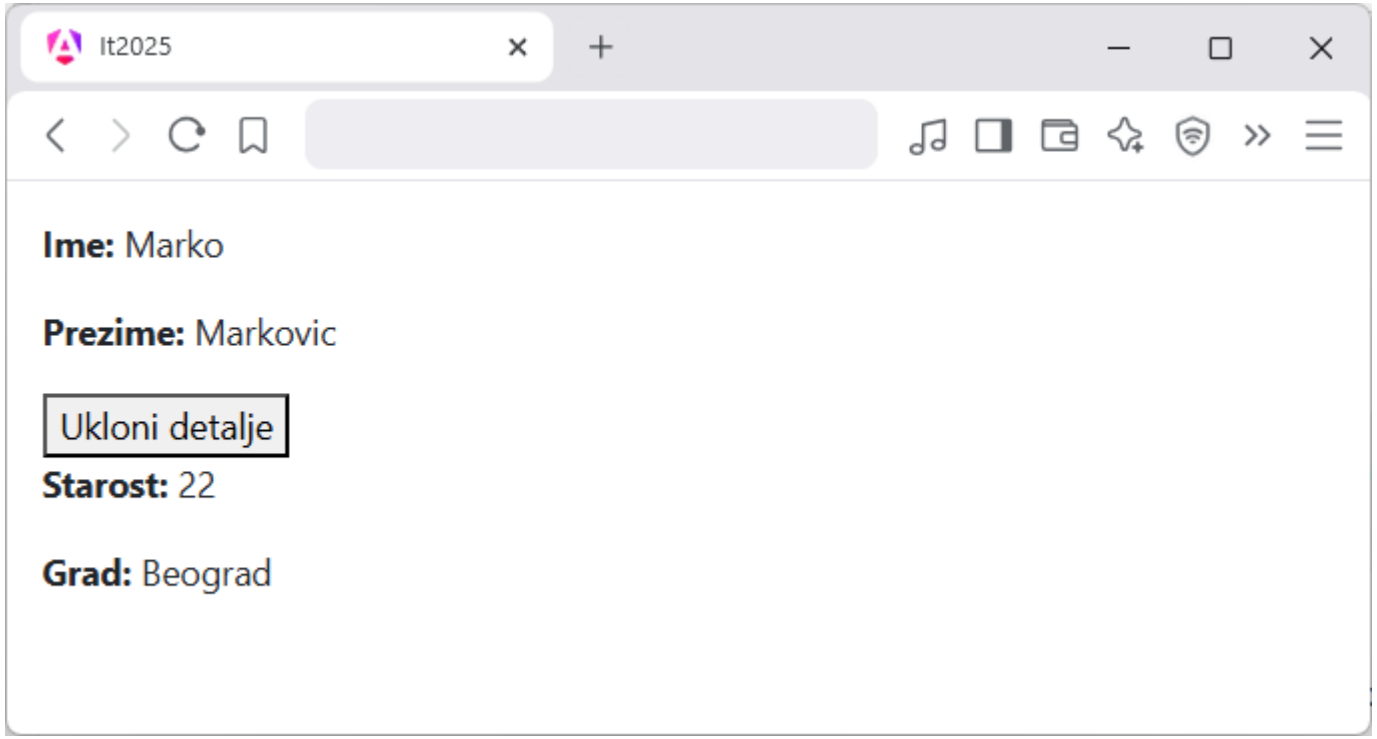
Kontrolna strukture @if i @else

```
@if (ulogovan) {
  <p>Dobrodošli nazad!</p>
} @else {
  <p>Molimo prijavite se.</p>
}
```

Kontrolna struktura @if

```
<div class="p-3">
  <p><strong>Ime:</strong> {{ osoba.ime }}</p>
  <p><strong>Prezime:</strong> {{ osoba.prezime }}</p>


  <button (click)="toggleDetalji()">
    {{ dugmeTekst }}
  </button>


  @if (prikaziDetalje) {
    <div>
      <p><strong>Starost:</strong> {{ osoba.starost }}</p>
      <p><strong>Grad:</strong> {{ osoba.grad }}</p>
    </div>
  }
</div>
```

Direktiva *ngIf, kontrolna struktura @if



Direktiva *ngSwitch

Omogućava prikaz različitih elemenata u zavisnosti od uslova

```
export class Primer07 {
  viewMode: string = 'list';
}
```

Direktiva ngSwitch je zastarela i ne koristi se od Angular 17 verzije

Direktiva *ngSwitch

```
<div class="btn-group">
  <button
(click)="viewMode='list'"  [class.active]="viewMode==='list'">List</button>
  <button
(click)="viewMode='grid'"  [class.active]="viewMode==='grid'">Grid</button>
  <button (click)="viewMode='other'"
[class.active]="viewMode==='other'">Other</button>
</div>


<div [ngSwitch]="viewMode">
  <p *ngSwitchCase="'list'">Lista</p>
  <p *ngSwitchCase="'grid'">Mreža</p>
  <p *ngSwitchCase="'other'">Druga opcija</p>
  <p *ngSwitchDefault>Default</p>
</div>
```

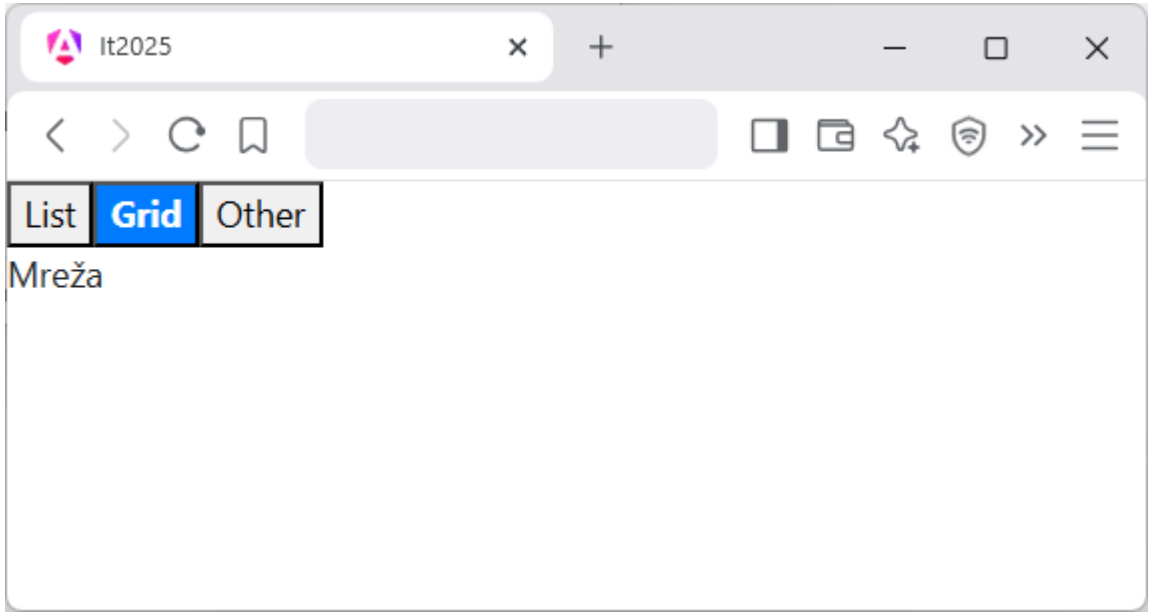
Kontrolna struktura @switch

- @switch je nova Angular kontrolna struktura za grananje u šablonu
- Zamenjuje zastareli ngSwitch i pripadajuće *ngSwitchCase i *ngSwitchDefault
- Deo je novog control flow sistema u Angularu 17+
- Koristi se za prikaz različitog sadržaja na osnovu vrednosti izraza
- Sintaksa je čitljivija i jednostavnija od stare direktive
- Podržava @case i @default blokove
- Za korišćenje u standalone komponentama potrebno je importovati **CommonModule**

Kontrolna struktura @switch

```
<div class="btn-group">
  <button
(click)="viewMode='list'"  [class.active]="viewMode==='list'">List</button>
  <button
(click)="viewMode='grid'"  [class.active]="viewMode==='grid'">Grid</button>
  <button (click)="viewMode='other'"
[class.active]="viewMode==='other'">Other</button>
</div>

@switch (viewMode) {
  @case ('list')  { <p>Lista</p> }
  @case ('grid')  { <p>Mreža</p> }
  @case ('other') { <p>Druga opcija</p> }
  @default        { <p>Default</p> }
}
```

Direktiva *ngSwitch, kontrolna struktura @switch



Direktiva ngClass

- Atributska direktiva za dinamičko dodavanje ili uklanjanje CSS klasa
- Koristi se za stilizovanje elemenata na osnovu stanja ili izraza
- Može prihvatiti:
  - string – jedna ili više klasa
  - niz – lista klasa
  - objekat – klase sa uslovima (najkorisniji oblik)
- Za standalone komponente dostupna preko CommonModule

Direktiva ngClass

```
export class Primer08 {
isRed = true;
isBlue = false;


promeniBoju() {
  this.isRed = !this.isRed;
  this.isBlue = !this.isBlue;
}
```

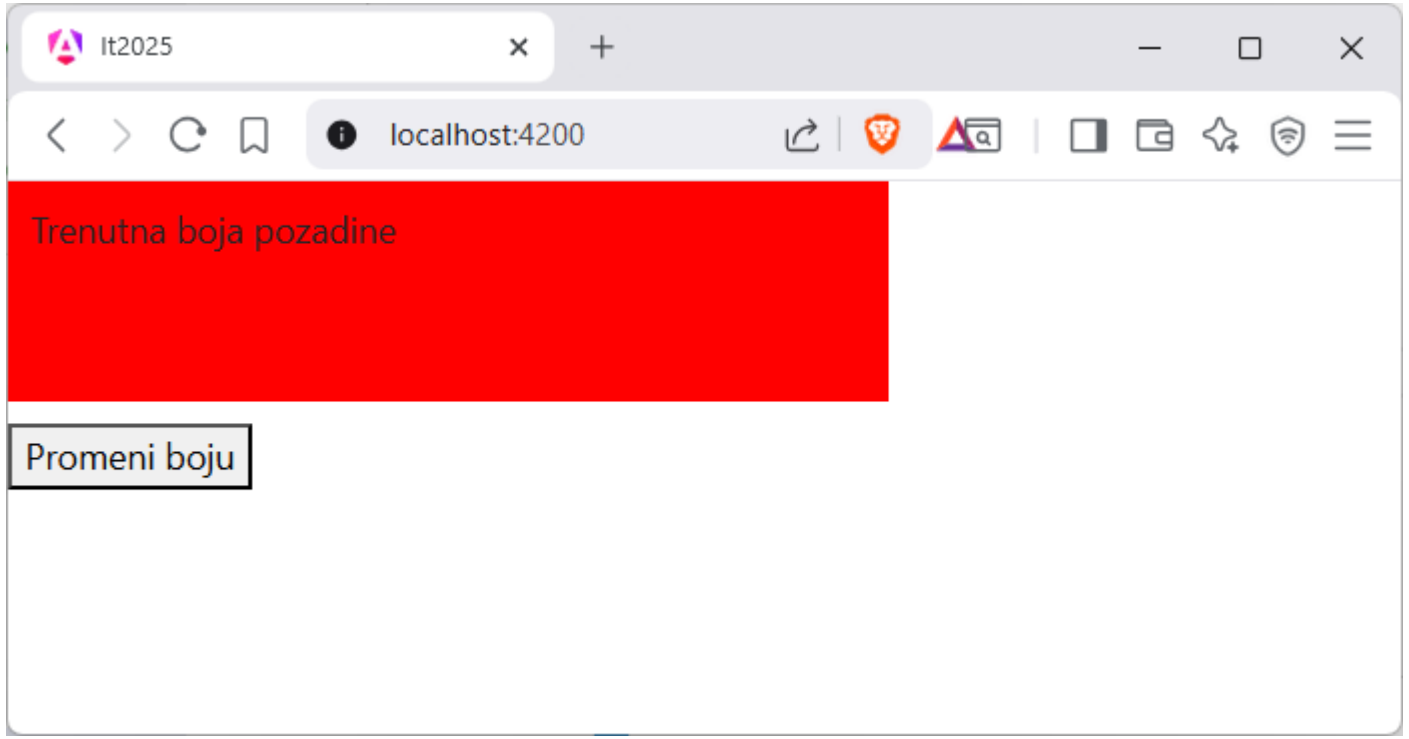
CSS fajl komponente

```css
div {
  width: 400px;
  height: 100px;
  padding: 10px;
  margin-bottom: 10px;
}


.crvena-boja {
  background-color: red;
}


.plava-boja {
  background-color: blue;
}
```

Šablon komponente

```
<div [ngClass]="{ 'crvena-boja': isRed, 'plava-boja': isBlue }">
  Trenutna boja pozadine
</div>


<button (click)="promeniBoju()">
  Promeni boju
</button>
```

ngClass direktiva



Uključivanje ili isključivanje jedne CSS klase

- Koristi se kada želimo jednu konkretnu klasu da uključimo ili isključimo
- Sintaksa: **[class.nazivKlase]="uslov"**
- Kada je uslov true, Angular dodaje klasu na element
- Kada je uslov false, Angular uklanja klasu
- Jednostavnije od ngClass kada ne radimo sa više klasa odjednom
- Pogodno za toggle stanja, aktivne elemente i vizuelna isticanja
- Dostupno i u standalone komponentama (preko CommonModule)

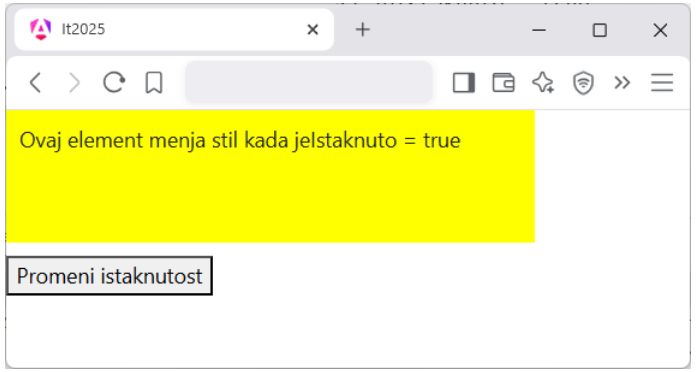
Klasa komponente i CSS fajl komponente

```
export class Primer08 {
jeIstaknuto = false;

promeniIstaknuto() {
  this.jeIstaknuto = !this.jeIstaknuto;
}

}
```

```
.istaknuto {
  background-color: yellow;
  padding: 10px;
}
```

Šablon komponente

```
<div [class.istaknuto]="jeIstaknuto">
  Ovaj element menja stil kada jeIstaknuto =
true
</div>


<button (click)="jeIstaknuto = !jeIstaknuto">
  Promeni istaknutost
</button>
```



Pitanje 1

Angular komponenta ima javno polje pod nazivom ime. Na koji način se sadržaj ovog polja prikazuje u šablonu komponente

a. [ime]
b. {{ ime }}
c. { ime }

Odgovor: b

Pitanje 2

Angular komponenta ima javno polje: **sakrivanje** tipa boolean. Na koji način se svojstvo hidden nekog div elementa u šablonu vezuje za vrednost ovog polja?

a. <div [hidden]= "sakrivanje">Div 1
b. <div {hidden}= "sakrivanje">Div 1
c. <div {{hidden}}= "sakrivanje">Div 1

Odgovor: a

Pitanje 3

Klasa komponente ima funkciju prikazi() koja nema ulazne parameter i ne vraća vrednost. U šablonu komponente treba omogućiti poziv ove funkcije pri kliku na dugme:

a. <input type="button" value="Click" (click)="prikazi" />
b. <input type="button" value="Click" [click]="prikazi()" />
c. <input type="button" value="Click" (click)="prikazi()" />

Odgovor: c

Pitanje 4

Definisano je polje klase komponente tipa string pod nazivom **ime**. Za dvosmerno povezivanje vrednosti tekstualnog polja šablona komponente sa poljem ime komponente koristimo izraz:

a. <input type="text" [(ngModel)]="ime">
b. <input type="text" ngModel="ime">
c. <input type="text" (ngBind)="ime">

Odgovor: a

Pitanje 5

Angular 17+ aplikacija u klasi komponente ima definisan niz
stavke: string[] = ['stavka1', 'stavka2', 'stavka3', 'stavka4'];
Potrebno je prikazati listu (<ul>) čiji se <li> elementi generišu iz niza stavke. Koji kod treba upisati unutar <ul> taga (Angular)?

a. @if (stavke) { {{ stavke }} }
b. @for (s of stavke) { {{ s }} }
c. @for (s of stavke; track s) { {{ s }} }

Odgovor: c

Pitanje 6

U Angular 17+ aplikaciji definisano je polje klase komponente tipa boolean pod nazivom prikazi. Potrebno je uslovno prikazati element samo ako je prikazi === true. Koji kod treba upisati?

a. @if (prikazi) { }
b. @for (p of prikazi; track p) { }
c. @if (!prikazi) { }

Odgovor: a

Pitanje 7

Data je lista objekata:
proizvodi = [
{ id: 1, naziv: 'Laptop', cena: 95000 },
{ id: 2, naziv: 'Telefon', cena: 38000 }];

Koji kod unutar <ul> taga ispravno prikazuje naziv i cenu svakog proizvoda?

a. @if (proizvodi) { {{ proizvodi.naziv }} – {{ proizvodi.cena }} }
b. @for (p of proizvodi) { {{ p }} }
c. @for (p of proizvodi; track p.id) { {{ p.naziv }} – {{ p.cena }} }

Odgovor: c