

Bab 5

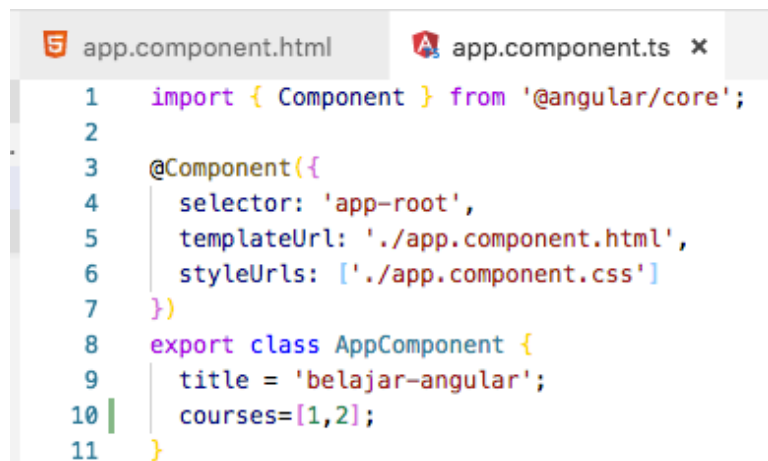
Directive

Directive adalah sebuah attribute yang biasanya berawalan dengan prefix ng- dan berfungsi sebagai **marker** layaknya class dan id ataupun sebagai **event driven**. Bisa dikatakan juga bahwa directive adalah tag-khususnya si Angular. Directive digunakan untuk modifikasi DOM, ada 2 cara untuk memodifikasi directive yaitu directive structural dan directive attribute. Jika menggunakan directive structural maka yang dimodifikasi adalah struktur DOM sedangkan jika directive attribute maka yang dimodifikasi adalah attribute DOM. Ada beberapa macam directive yang akan kita gunakan selama pembahasan ini.

6.1 ngIf

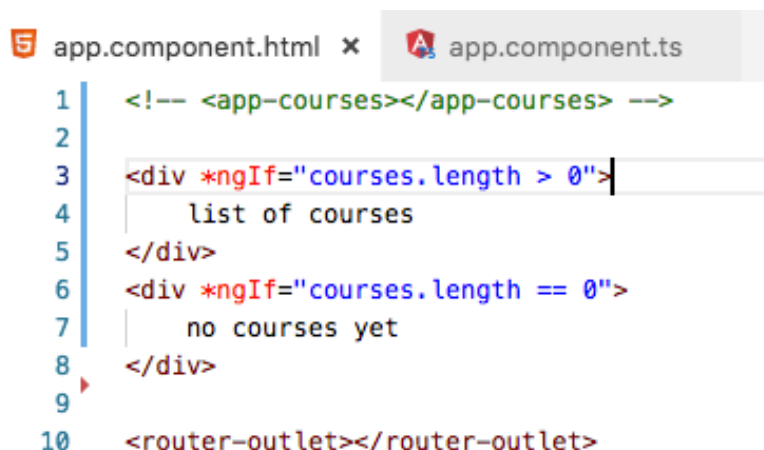
Directive ngIf digunakan untuk sebuah kondisi percabangan. Ada beberapa contoh penggunaan ngIf. Contoh Pertama

- buka file

A screenshot of a code editor showing the TypeScript file app.component.ts. The code defines an Angular component named AppComponent. It imports the Component class from @angular/core. The @Component decorator is configured with selector 'app-root', templateUrl './app.component.html', and styleUrls ['./app.component.css']. The class AppComponent has a title property set to 'belajar-angular' and a courses array containing the numbers 1 and 2.

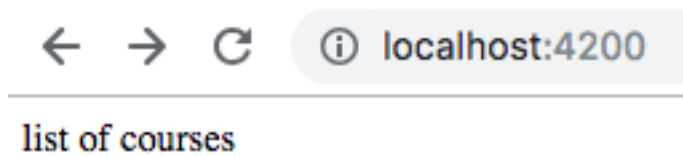
```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'belajar-angular';
10  courses=[1,2];
11 }
```

- buka file

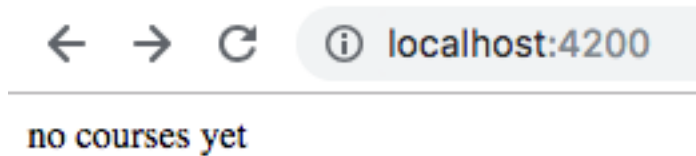
A screenshot of a code editor showing the HTML file app.component.html. The code uses the ngIf directive to conditionally render a list of courses. If the courses array has a length greater than 0, it displays 'list of courses'. If the length is 0, it displays 'no courses yet'. The code is placed within a <router-outlet> tag.

```
1 <!-- <app-courses></app-courses> -->
2
3 <div *ngIf="courses.length > 0">
4   list of courses
5 </div>
6 <div *ngIf="courses.length == 0">
7   no courses yet
8 </div>
9
10 <router-outlet></router-outlet>
```

- jalankan localhost maka hasilnya seperti berikut



- Jika array pada **app.component.ts** (**courses=[];**) dikosongkan maka hasilnya seperti berikut



contoh kedua menggunakan else:

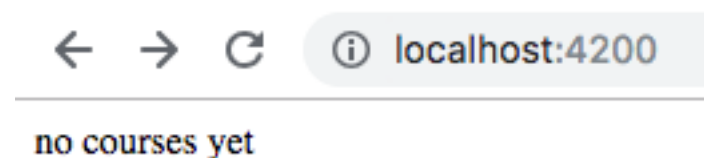
- buka file app.component.html modifikasi kodenya menjadi berikut

```
app.component.html x app.component.ts
9
10 <!-- percabangan kedua -->
11 <div *ngIf="courses.length > 0; else noCourses">
12 |   list of courses
13 </div>
14 <ng-template #noCourses>
15 |   no course yet
16 </ng-template>
17
18
19 <router-outlet></router-outlet>
```

- jalankan localhost dengan kondisi array pada app.component
 - o dengan array kosong

```
app.component.html app.component.ts x
8 export class AppComponent {
9 |   title = 'belajar-angular';
10 |   courses=[];
11 }
```

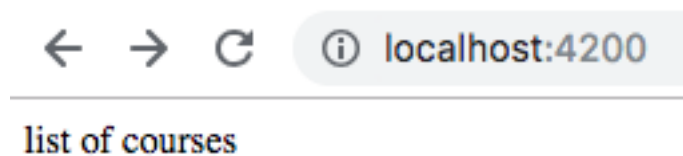
hasilnya :



- dengan array ada isi

```
app.component.html  app.component.ts x
8   export class AppComponent {
9   |   title = 'belajar-angular';
10  |   courses=[1,2];
11  | }
```

hasilnya



cara ketiga :

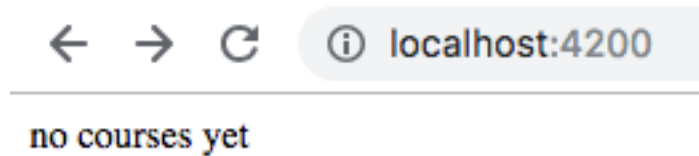
- buka file app.component.html modifikasi kodenya menjadi berikut

```
app.component.html x  app.component.ts
17
18 <!-- percabangan ketiga -->
19 <div *ngIf="courses.length > 0; then coursesList else noCourses"></div>
20 <ng-template #coursesList>
21 |   list of courses
22 </ng-template>
23
24 <ng-template #noCourses>
25 |   no course yet
26 </ng-template>
27
28 <router-outlet></router-outlet>
```

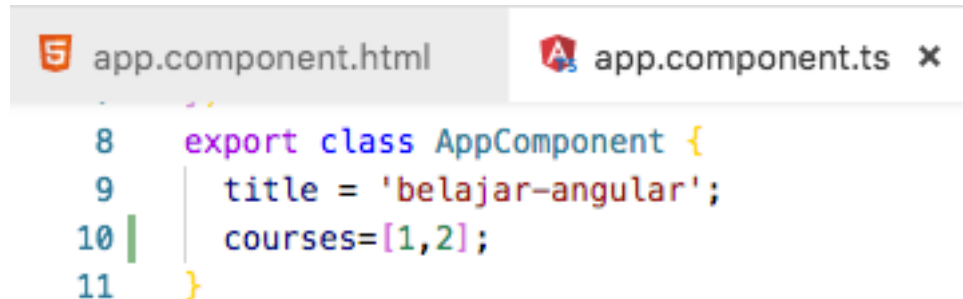
- jalankan localhost dengan kondisi array pada app.component
 - dengan array kosong

```
app.component.html  app.component.ts x
8   export class AppComponent {
9   |   title = 'belajar-angular';
10  |   courses=[];
11  | }
```

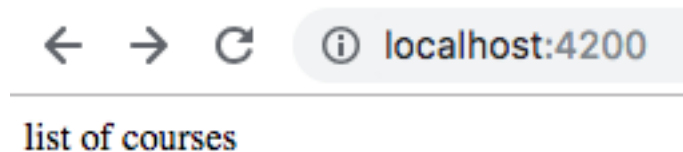
hasilnya :



- o dengan array ada isi



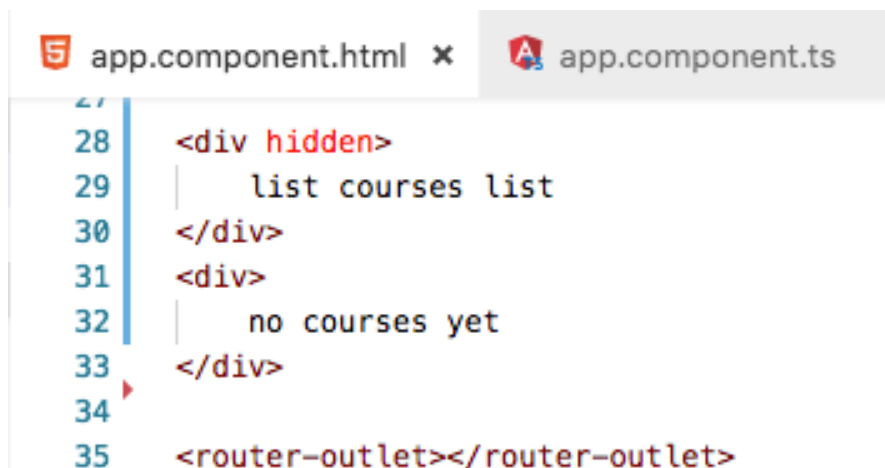
hasilnya



6.1 Hidden Property

Selain menggunakan directive ngIF kita juga dapat menggunakan property Hidden untuk menampilkan salah satu data. Jika menggunakan ngIF element yang bernilai salah atau false akan dihilangkan dari DOM sedangkan jika menggunakan Hidden property jika element tersebut bernilai true maka element tersebut yang akan dihidden. Untuk lebih jelasnya ikuti langkah berikut

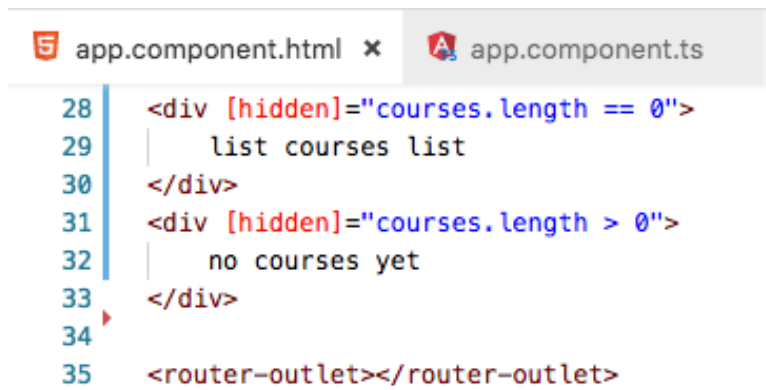
- buka app.component.html



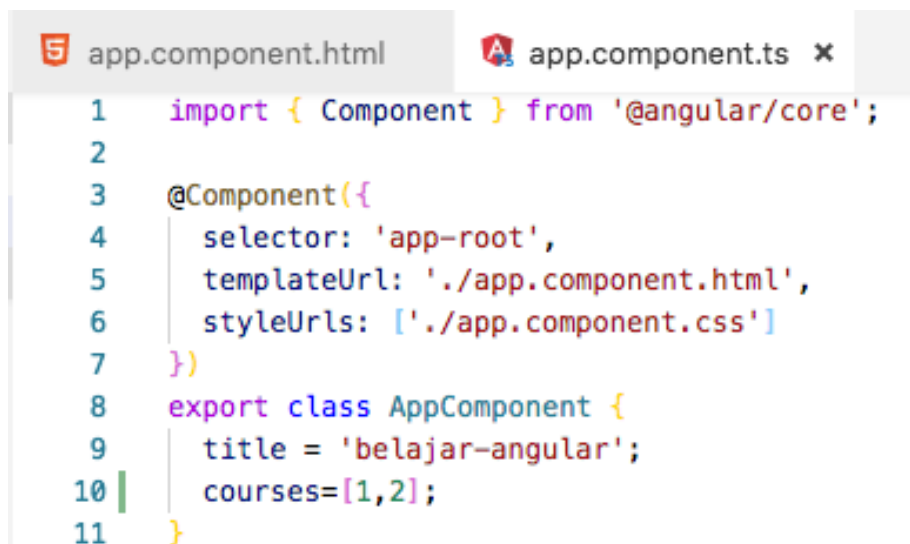
- jalankan localhost maka hasilnya seperti berikut



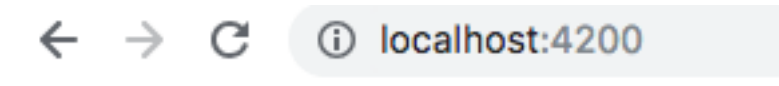
- selain contoh diatas kita juga dapat memberikan property seperti berikut



dengan catatan pada app.component.ts pada courses terdapat array courses dengan nilai 1 dan 2



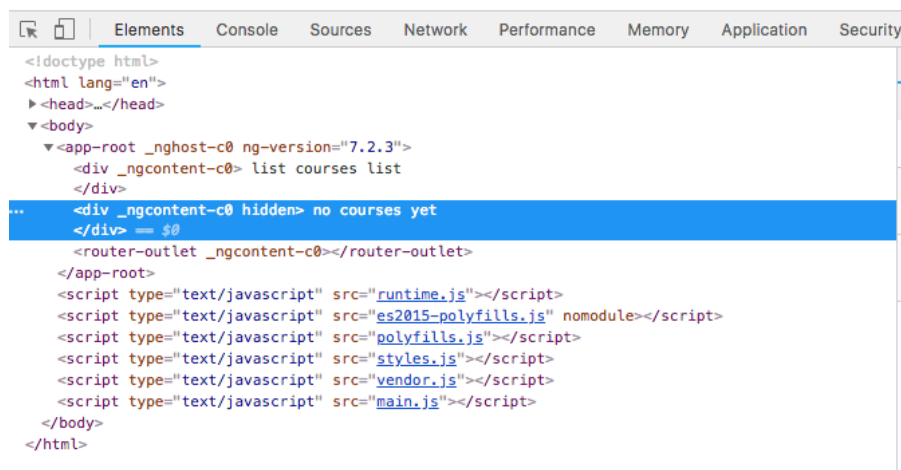
- jalankan localhost (berbeda hasil jika pada array courses tidak terdapat isinya)



list courses list

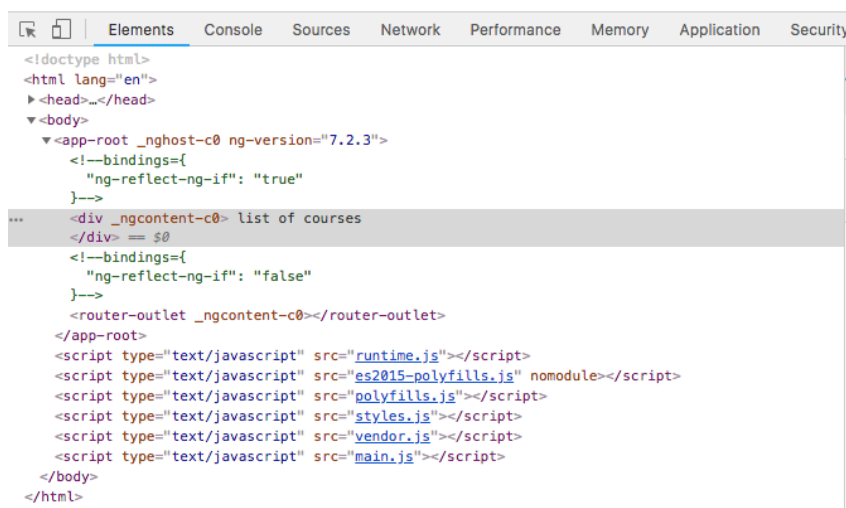
Jika kita inspect element maka akan terlihat **property hidden** tidak terdapat **kondisi true** ataupun **false**.

list courses list



berbeda jika kita menggunakan **ngIF** terdapat **bindings** dengan nilai **false**

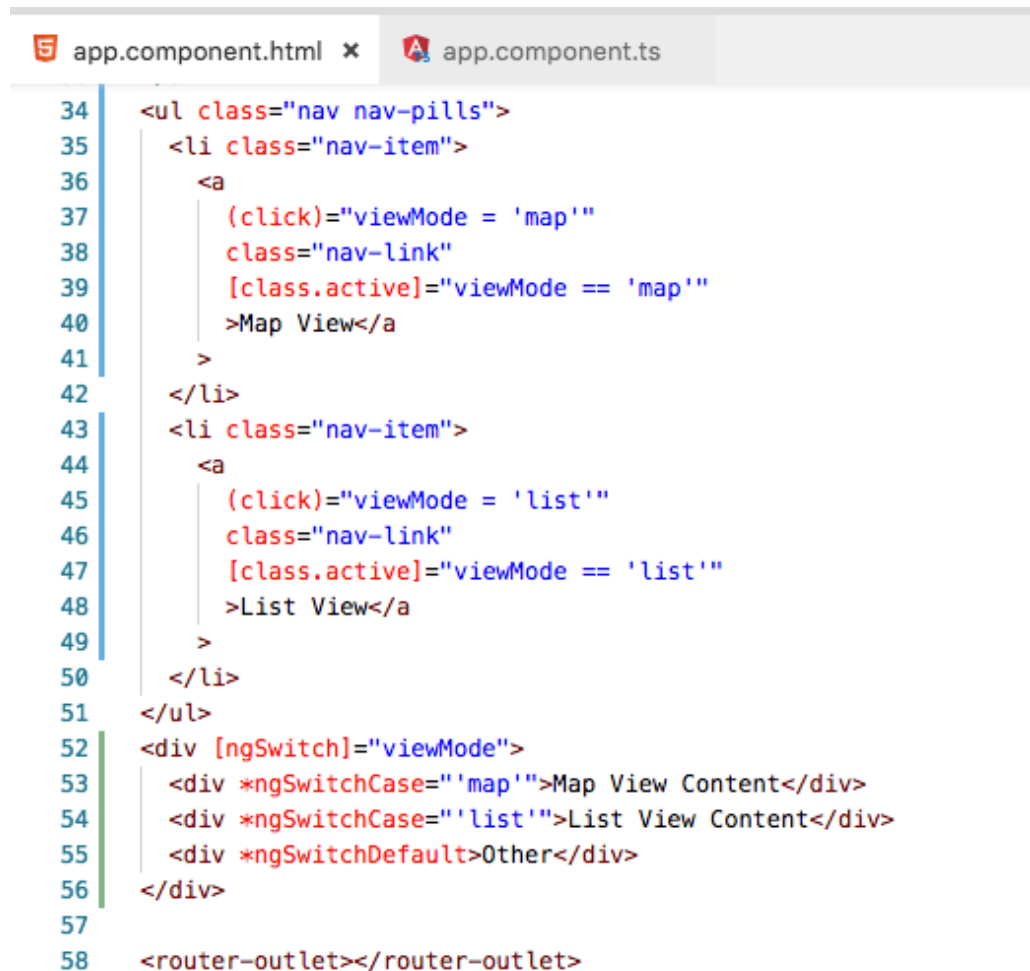
list of courses



6.1 ngSwitchCase

Menyediakan ekspresi switch untuk menyamakan dengan ekspresi ngSwitch yang tersedia. Ketika ekspresi sama, maka template NgSwitchCase menampilkan. Untuk lebih jelasnya ikuti langkah berikut :

- buka file app.component.html modifikasi codenya menjadi seperti berikut



```
34 <ul class="nav nav-pills">
35   <li class="nav-item">
36     <a
37       (click)="viewMode = 'map'"
38       class="nav-link"
39       [class.active]="viewMode == 'map'"
40     >Map View</a>
41   >
42 </li>
43   <li class="nav-item">
44     <a
45       (click)="viewMode = 'list'"
46       class="nav-link"
47       [class.active]="viewMode == 'list'"
48     >List View</a>
49   >
50 </li>
51 </ul>
52 <div [ngSwitch]="viewMode">
53   <div *ngSwitchCase="'map'">Map View Content</div>
54   <div *ngSwitchCase="'list'">List View Content</div>
55   <div *ngSwitchDefault>Other</div>
56 </div>
57
58 <router-outlet></router-outlet>
```

Penjelasan Code

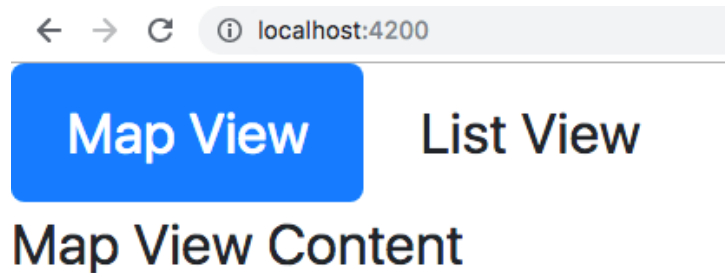
- line 45 terdapat **event binding** `(click)="viewMode='list'"`
- line 47 pada tag a ditambah **class binding**
`[class.active]="viewMode=='list'"`
- line 52 pada tag div terdapat **property binding** `[ngSwitch]`
- line 53, 54, 55 pada tag div ditambah **directive ngSwitchCase**

- buka file app.component.ts tambahkan **property viewMode** (line 12)



```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'belajar-angular';
10  courses=[1,2];
11
12  viewMode='map';
13
14 }
```

- run localhost maka hasilnya seperti berikut saat diklik list view maka akan muncul list view content dan jika kita pilih ListView maka akan tampil tulisan List View Content



6.1 ngFor

ngFor adalah sebuah directive pada angular yang berfungsi untuk melakukan looping terhadap beberapa data misal data kita adalah sebuah array, jadi directive ini akan melopping array kita pada bagian template. Untuk lebih jelasnya ikuti langkah-langkah berikut :

- buka app.component.ts property CoursesFor yang berisikan array (**line 12**)


```
app.component.html  app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'belajar-angular';
10   courses=[1,2];
11
12   coursesFor=[
13     {id:1,name:'courses1'},
14     {id:2,name:'courses2'},
15     {id:3,name:'courses3'},
16     {id:4,name:'courses4'},
17     {id:5,name:'courses5'},
18   ]
19
20   viewMode='map';
21
22 }
```

- buka file app.component.html tambahkan directive ngFor pada element li

```
app.component.html x  app.component.ts
58  <ul>
59    <li *ngFor="let item of coursesFor; index as i">
60      {{i}}-{{item.name}}
61    </li>
62  </ul>
63  <router-outlet></router-outlet>
```

Penjelasan code

- line 59 terdapat directive ngFor dengan let item dari property courseFor pada app.component.ts dan dengan index alias i
 - linr 60 adalah string interpolasi
- jika dijalankan maka hasilnya seperti berikut

← → ↻ ⓘ localhost:4200

- 0-courses1
- 1-courses2
- 2-courses3
- 3-courses4
- 4-courses5

- kita juga dapat memberi tanda tertentu pada index yang bernilai ganjil dengan menggunakan isEven <https://angular.io/api/common/NgForOf>

```
app.component.html x app.component.ts
64 <ul>
65   <li *ngFor="let item of coursesFor; even as isEven">
66     {{item.name}}
67     <span *ngIf="isEven">(Even)</span>
68   </li>
69 </ul>
70 <router-outlet></router-outlet>
```

← → ↻ ⓘ localhost:4200

- courses1 (Even)
- courses2
- courses3 (Even)
- courses4
- courses5 (Even)

6.1 ngFor dan change Detection

Dengan memodifikasi directive ngFor kita dapat melakukan penambahan atau pengurangan data. Untuk lebih jelasnya ikuti langkah berikut :

- Pada percobaan ini kita akan menambahkan sebuah data array pada **coursesFor**
- tambahkan button pada app.component.html (line 64)

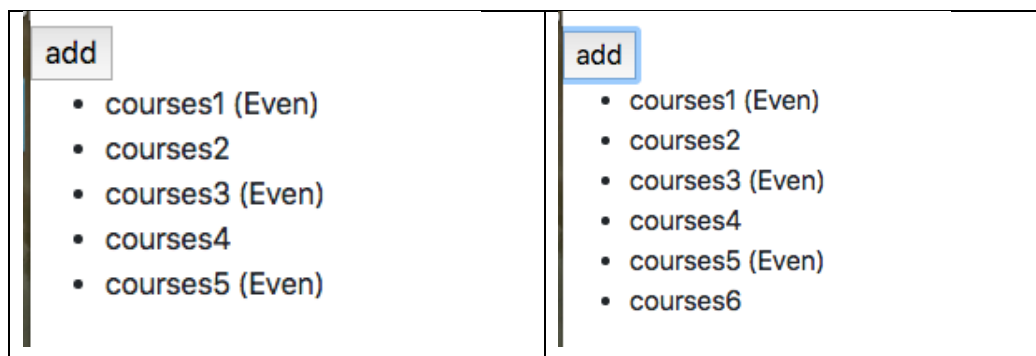
```
app.component.html x app.component.ts
64 <button (click)="onAdd()">add</button>
65 <ul>
66   <li *ngFor="let item of coursesFor; even as isEven">
67     {{item.name}}
68     <span *ngIf="isEven">(Even)</span>
69   </li>
70 </ul>
71 <router-outlet></router-outlet>
```

- tambahkan method **onAdd()**

```
app.component.html app.component.ts x
11
12 coursesFor=[
13   {id:1,name:'courses1'},
14   {id:2,name:'courses2'},
15   {id:3,name:'courses3'},
16   {id:4,name:'courses4'},
17   {id:5,name:'courses5'},
18 ]
19
20 viewMode='map';
21
22 onAdd(){
23   this.coursesFor.push({id:6,name:'courses6'});
24 }
```

Penjelasan code

- line 22-24 adalah sebuah method yang digunakan untuk melakukan push data (id:6, name:'courses6') ke coursesFor
- sehingga hasilnya seperti berikut (jika kita tekan button add maka akan ditambahkan sebuah data courses 6)



- setelah kita berhasil menambahkan sebuah data array pada `courseFor` maka untuk selanjutnya kita akan mencoba untuk melakukan penghapusan data.
- Tambahkan sebuah method `onRemove` pada `app.component.ts`

```

26 |     onRemove(item){
27 |         console.log(item);
28 |         let index=this.coursesFor.indexOf(item);
29 |         console.log(index);
30 |         this.coursesFor.splice(index,1);
31 |     }

```

Penjelasan code

- line 26 sebuah method dengan parameter `item`. Jika kita **`console.log(item)`** maka hasilnya pada saat kita hapus salah satu data maka hasilnya akan seperti berikut

```

▼ {id: 5, name: "courses5"} ⓘ
  id: 5
  name: "courses5"
  ► __proto__: Object

```

- line 27 sebuah variabel baru dengan nama `index` bertipe data number. Untuk lebih jelasnya kita bisa tambahkan **`console.log(index)`** pada **method `onRemove`** dan hasilnya pada saat kita hapus salah satu data maka akan seperti berikut

```

▼ {id: 5, name: "courses5"} ⓘ
  id: 5
  name: "courses5"
  ► __proto__: Object
4

```

- Buka `app.component.html` dan tambahkan sebuah button untuk menghapus (line69)

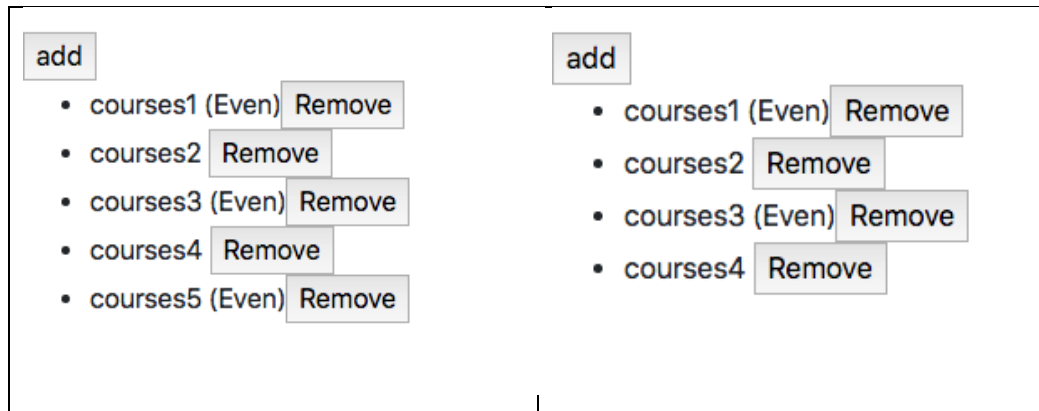
```

app.component.html × app.component.ts
64 | <button (click)="onAdd()">add</button>
65 | <ul>
66 |   <li *ngFor="let item of coursesFor; even as isEven">
67 |     {{item.name}}
68 |     <span *ngIf="isEven">(Even)</span>
69 |     <button (click)="onRemove(item)"> Remove</button>
70 |   </li>
71 | </ul>
72 | <router-outlet></router-outlet>

```

penjelasan code :

- line 69 adalah sebuah button dimana terdapat event click yang memanggil method onRemove dengan parameter item
- hasilnya seperti berikut (pada saat button remove diclick maka salah satu data akan hilang sementara)



- selain itu kita juga dapat melakukan perubahan status menggunakan event click
- buka file app.component.html tambahkan button (line 70)

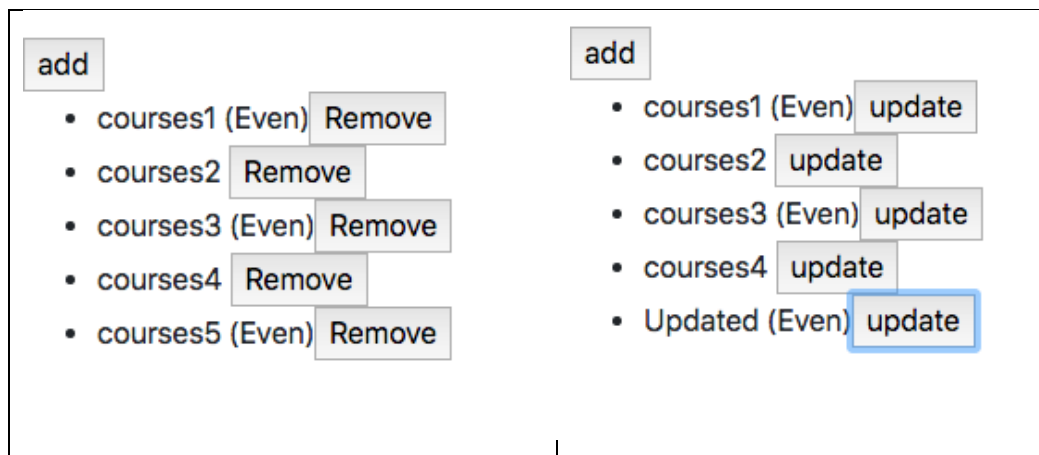
```
app.component.html x app.component.ts
65 <ul>
66   <li *ngFor="let item of coursesFor; even as isEven">
67     {{item.name}}
68     <span *ngIf="isEven">(Even)</span>
69     <!-- <button (click)="onRemove(item)"> Remove</button> -->
70     <button (click)="onChange(item)"> update</button>
71   </li>
72 </ul>
73 <router-outlet></router-outlet>
```

penjelasan code :

- pada line 70 terdapat sebuah button dan event click dimana event click ini memanggil method onChange dengan parameter item
- buka file app.component.ts buatlah sebuah method onChange dengan parameter item dan didalam method tersebut adanya perubahan string menjadi updated

```
app.component.html app.component.ts x
33 onChange(item){
34   item.name='Updated';
35 }
36
```

- hasilnya



6.1 ngFor dan trackby

Pada studi kasus berikut ini kita akan menampilkan isi array saat button diclick. Pemasalahan yang muncul jika button click tersebut diclick berkali-kali, maka secara tidak langsung data akan mereload ulang. Untuk lebih jelasnya ikuti langkah berikut:

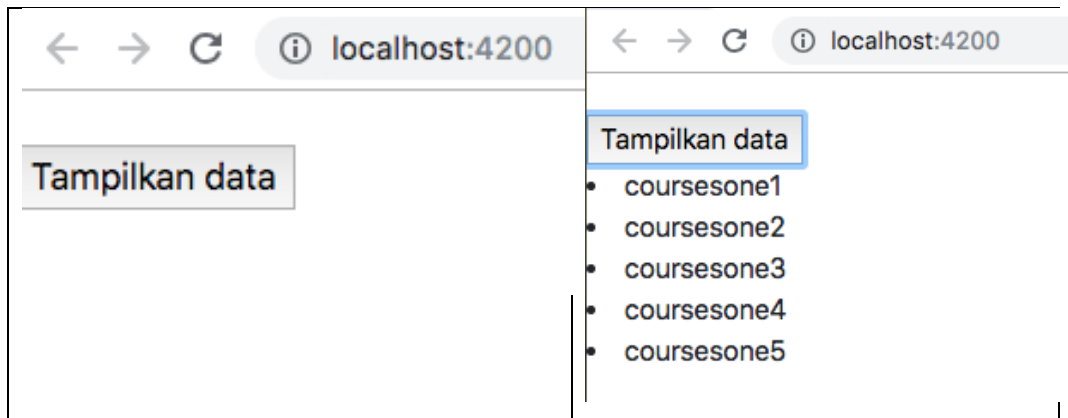
- Buka file app.component.ts buatlah sebuah method dengan nama loadCourses (line 38) tapi sebelumnya buat sebuah property dengan nama coursesForOne (line 37)



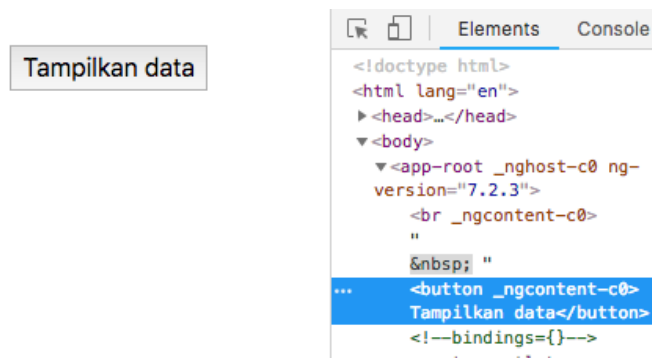
- buka file app.component.html dan tambahkan code seperti pada gambar

```
app.component.html x app.component.ts
73 <button (click)="loadCourses()"> Tampilkan data</button>
74 <li *ngFor="let itemone of coursesForOne">
75   {{itemone.nama}}
76 </li>
77
78 <router-outlet></router-outlet>
```

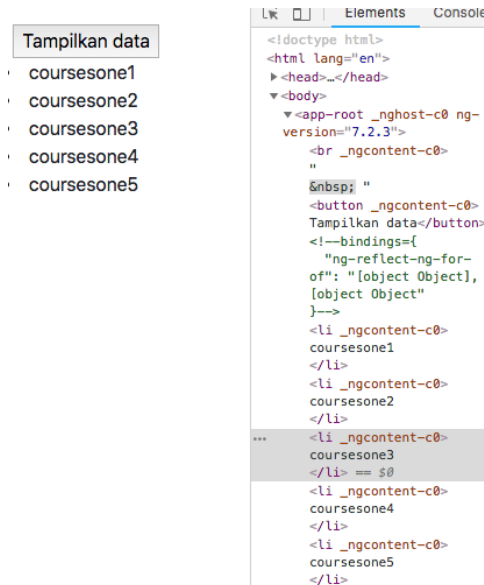
- hasilnya saat button diclick



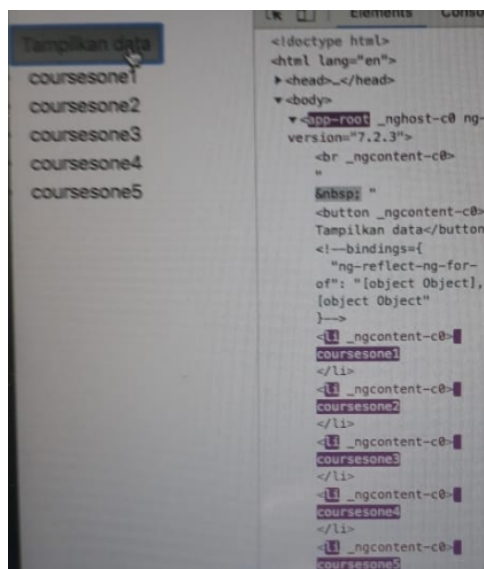
- sekarang kita lakukan analisa buka inspect element, saat button belum diclick seperti pada gambar berikut



- dan saat button diklik maka element ul akan muncul seperti pada gambar berikut



- jika kita lakukan klik kembali maka button akan merespon kembali dengan menampilkan data yang sudah ada dalam arti mengunduh ulang yang sudah ada (jika kita lakukan klik kembali maka pada inspect element akan muncul highlight warna ungu dimasing-masing element li)



- oleh karena itu kita membutuhkan **TrackBy** yang nanti digunakan untuk mengecek jika data sudah ada maka button tidak perlu melakukan actionnya kembali
- untuk menambahkan **TrackBy** dengan cara menambahkan pada `app.component.html` pada **directive ngFor**


```
app.component.html x app.component.ts
73 <br>
74 &nbsp; <button (click)="loadCourses()"> Tampilkan data</button>
75 <li *ngFor="let itemone of coursesForOne; TrackBy:trackCourse">
76   {{itemone.nama}}
77 </li>
78
79 <router-outlet></router-outlet>
```

- selain itu tambahkan juga sebuah method trackCourse dengan parameter index dan itemone pada app.component.ts

```
app.component.html app.component.ts x
47 trackCourse(index, itemone){
48   return itemone ? itemone.id : undefined;
49 }
```

Penjelasan code:

- line 48 sebuah nilai kembalian jika itemone bernilai true maka itemone.id akan ditampilkan dan jika itemone bernilai false maka itemone tidak terdefinisi
- jika berhasil maka pada saat button tampilkan data diklik untuk kedua kalinya pada inspect element tidak ada muncul highlight ungu pada masing-masing element li

6.1 The leading Asterik

leading asterik adalah tanda asterisk yang ada pada directive seperti *ngFor dsb. adapun maksud leading asterisk pada sebuah directive adalah bahwa kita memberi tahu angular untuk menulis ulang markup tertentu, seperti code berikut

```
11 <div *ngIf="courses.length > 0">
12   list of courses
13 </div>
```

pada gambar diatas terdapat sebuah directive *ngIf dimana fungsi dari asterisk pada ngIf adalah memberi tahu angular untuk menulis ulang markup dari div tersebut menggunakan ng template dengan menggunakan property binding seperti pada gambar berikut

```

15 <ng-template [ngIf]="courses.length>0">
16   <div>
17     | list of course
18   </div>
19 </ng-template>

```

6.1 ngClass

jika pada percobaan sebelumnya kita membuat component favorite menggunakan 2 class binding yaitu class binding fa-star dan fa-star-o seperti pada gambar berikut

```

9   <span
10   class="fa"
11   [class.fa-star]="isSelected"
12   [class.fa-star-o]="!isSelected"
13   (click)="onClickAlias()"></span>

```

selain menggunakan class binding seperti diatas kita juga dapat menggunakan attribute directive class seperti berikut

```

15 <span
16   class="fa"
17   [ngClass]="{
18     'fa-star':isSelected,
19     'fa-star-o':!isSelected
20   }"
21   (click)="onClickAlias()"></span>

```

6.1 Custom Directive

pada directive kita juga dapat membuat sebuah custom directive, sebagai contoh kita ingin membuat semua inputan pada textbox secara otomatis menjadi lowercase. Untuk lebih jelasnya ikuti langkah-langkah berikut :

- pertama kita harus membuat directive dengan nama input-format terlebih dahulu dengan perintah seperti berikut

```

Dimass-MacBook-Pro:belajar-angular-3(directive) dimaswahyu$ ng g d input-format
CREATE src/app/input-format.directive.spec.ts (245 bytes)
CREATE src/app/input-format.directive.ts (151 bytes)
UPDATE src/app/app.module.ts (636 bytes)

```

Penjelasan :

ng g d input-format => ng generate directive nama-directivenya

- jika directive berhasil digenerate maka kita pastikan di app.module.ts pada @NgModule terdapat nama directive yang kita buat tadi



```
10 @NgModule({
11   declarations: [
12     AppComponent,
13     CoursesComponent,
14     InputFormatDirective
15   ],
```

- buka input-format.directive.ts dan tambahkan decorator HostListener seperti pada gambar berikut



```
1 import { Directive, HostListener } from '@angular/core';
2
3 @Directive({
4   selector: '[appInputFormat]'
5 })
6 export class InputFormatDirective {
7
8   @HostListener('focus') onfocus(){
9     console.log("onFocus");
10  }
11
12   @HostListener('blur') onblur(){
13     console.log("onBlur");
14  }
15   constructor() { }
16
17 }
```

Penjelasan code :

- line 1 kita tambahkan (mengimport) sebuah fungsi decorator HostListener agar dapat menggunakan event DOM yang menyediakan method untuk menangani sebuah event yang sedang berjalan. Untuk lebih jelasnya penggunaan HostListener buatlah 2 buah method dengan nama onfocus dan onblur
- line 8 adalah sebuah method onfocus dengan menggunakan function decorator @HostListener dengan parameter event windows focus (<https://developer.mozilla.org/en-US/docs/Web/Events/focus>)

- line 9 sebuah method onBlur dengan menggunakan function decorator @HostListener dengan parameter event windows blur (<https://developer.mozilla.org/en-US/docs/Web/Events/focus>)
- Buka file app.component.html dan tambahkan code berikut

```

input-format.directive.ts
88 <br>
89 &nbsp;<input type="text" appInputFormat>
90 <router-outlet></router-outlet>

```

Penjelasan code

- line 89 terdapat appInputFormat adalah selector pada **input-format.directive.ts**

```

3 @Directive({
4   selector: '[appInputFormat]'
5 })

```

- Kita jalan localhost:4200 setelah itu lakukan percobaan click pada textbox dan click diluar textbox. Amati dengan menggunakan console inspect element. Pada saat kita click pada textbox maka pada console akan muncul onFocus tetapi jika click diluar textbox maka console akan keluar onBlur
- Setelah kita memahami fungsi dari decorator @HostListener maka untuk berikutnya kita akan mencoba membuat logika untuk merubah value dari textbox menjadi lowercase
- buka file input-format.directive.ts dan modifikasi codenya menjadi berikut

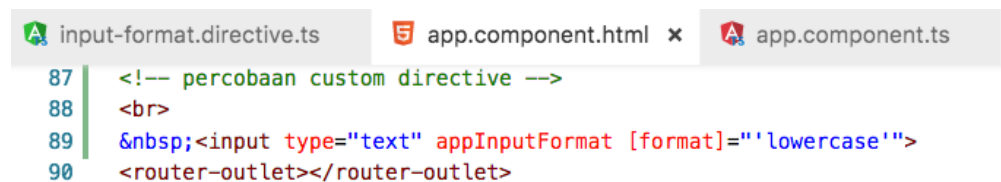
```

input-format.directive.ts x app.component.html app.component.ts
1 import { Directive, HostListener, ElementRef } from '@angular/core';
2
3 @Directive({
4   selector: '[appInputFormat]'
5 })
6 export class InputFormatDirective {
7
8   constructor(private el:ElementRef) { }
9   @HostListener('blur') onBlur(){
10     // console.log("onBlur")
11     let value:string = this.el.nativeElement.value;
12     this.el.nativeElement.value = value.toLowerCase();
13   }
14 }

```

Penjelasan code

- line 8 kita membutuhkan sebuah reference untuk ke host element maka kita tambahkan (inject) pada constructor reference object ElementRef. Dengan kata lain ElementRef digunakan untuk mengakses DOM object
- line 11 sebuah property dengan nama value dengan tipe data string yang berisi sebuah property nativeElement pada class ElementRef
- line 12 untuk merubah inputan menjadi huruf kecil semua
- Jalankan localhost dan berikan masukan dengan huruf besar semua setelah itu tekan tab, jika berhasil maka valuenya akan berubah menjadi huruf kecil semua
- atau kita bisa menggunakan cara lain dengan menggunakan **property binding**
- buka file app.component.html dan tambahkan property binding dengan nama format



```
87 <!-- percobaan custom directive -->
88 <br>
89 &nbsp;<input type="text" appInputFormat [format]='lowercase'>
90 </router-outlet></router-outlet>
```

- buka file input-format.directive.ts tambahkan decorator input dan modifikasi codenya seperti pada gambar berikut



```
1 import { Directive, HostListener, ElementRef, Input } from '@angular/core';
2
3 @Directive({
4   selector: '[appInputFormat]'
5 })
6 export class InputFormatDirective {
7   @Input('format') format;
8
9   constructor(private el:ElementRef) { }
10  @HostListener('blur') onblur(){
11    // console.log("onBlur")
12
13    let value:string = this.el.nativeElement.value;
14    if(this.format=='lowercase')
15      this.el.nativeElement.value = value.toLowerCase();
16    else
17      this.el.nativeElement.value = value.toUpperCase();
18  }
19 }
20 }
```

Penjelasan code

- line 7 adalah sebuah decorator input dengan nama property format

- line 14 – 16 adalah sebuah kondisi dimana jika format sama dengan lowercase maka semua inputan akan dibuat menjadi huruf kecil semua sedangkan jika selain lowercase maka akan dibuat huruf besar
- cat:format disini adalah kondisi di **app.component.html**

```

87 | <!-- percobaan custom directive -->
88 | <br>
89 | &nbsp;<input type="text" appInputFormat [format]='uppercase'>
90 | </router-outlet></router-outlet>

```

- atau kita juga dapat menggunakan cara lain yaitu menggunakan nama selector sebagai property binding
- buka app.component.html modifikasi codenya menjadi berikut

```

87 | <!-- percobaan custom directive -->
88 | <br>
89 | <!-- &nbsp;<input type="text" appInputFormat [format]='uppercase' -->
90 | &nbsp;<input type="text" [appInputFormat]='uppercase'>
91 | </router-outlet></router-outlet>

```

- buka input-format.directive.ts dan tambahkan decorator input dengan parameter appInputFormat

```

5 | }
6 | export class InputFormatDirective {
7 |   // @Input('format') format;
8 |   @Input('appInputFormat') format;

```

- Jika dijalankan sebagai contoh kita memasukkan kalimat dengan huruf kecil dan pada saat kita tab maka akan berubah menjadi huruf besar semua seperti berikut

