Angular 8 : Online Class

Class – 10

By: Sahosoft Solutions

Presented by : Chandan Kumar

NgFor (*ngFor)

Angular provides NgForOf directive. It instantiates a template for every element of given iterator. NgForOf has different local variables that can be used in iteration. The local variables are index, first, last, even, odd and ngForOf. NgForOf is used with HTML elements as well as <ng-template>. Whenever the contents of iterator changes, NgForOf performs respective changes in DOM. These changes are tracked by object identity by default. We can change tracking identify by using trackBy. We assign a user defined function to trackBy and that function will return an identity for every element of iterator. When we use trackBy with NgForOf, it starts change propagation tracked by given identity and not by object identity. Using trackBy improves the performance of NgForOf directive.

NgForOf provides several exported values that can be aliased to local variables.
index: Index of current item.
even: True for an even index.
odd: True for an odd index.
first: True for first item.
last: True for last item.
ngForOf: It is useful to alias when expression is more complex than a property access, for example using Async pipe.

NgForOf with HTML Elements

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.

NgForOf directive is used with HTML elements as following.

```
<li *ngFor="let data of allPerson">
-----------------
</li>
```

NgForOf with <ng-template>

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.

NgForOf directive is used with <ng-template> as following.

```
 <ng-template ngFor let-data [ngForOf]=" allPerson">
        ---------------
    </ng-template>
```

index, even and odd with HTML Elements

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.

NgForOf is used with index, even and odd local variables using <div> element as follow

```
<div *ngFor="let person of allPerson; index as i; even as isEven; odd as isOdd">
----------------
</div>
```

index, even and odd with ng-template

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.

NgForOf is used with index, even and odd local variables using <ng-template> element as follow

```
<ng-template ngFor let-person [ngForOf]=" allPerson" let-i="index" let-isEven="even" let-isOdd="odd">
----------------
</ng-template>
```

first and last with HTML Elements

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.

NgForOf is used first and last local variables of NgForOf using <div> element as follow

```
<div *ngFor="let person of allPerson; index as i; first as isFirst; last as isLast">
----------------
</div>
```

first and last with ng-template

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.

NgForOf is used first and last local variables of NgForOf using <**ng-template**> as follow

```
<ng-template ngFor let-person [ngForOf]=" allPerson" let-i="index" let-isFirst="first" let-isLast="last">
----------------
</ ng-template >
```

trackBy with HTML Elements

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.
The use of trackBy it's a performance optimization and is usually not needed by default, it's in principle only needed if running into performance issues.
Suppose we have some data coming from some API request into the collection and we need to change the data over the web page using ngFor directive. In this case, Angular 7 will remove all the DOM elements that associated with the data and will create them again in the DOM tree, even the same data is coming (as shown in below screenshot). That means a lot of Dom manipulations will happen if a large amount of data coming from the API.

- The solution is, we can use trackBy function, which will be helpful for Angular to track the items which have been added or removed.
- TrackBy function will take two arguments first is index and second is current item and we can return the unique identifier as a return argument.

```
<tr *ngFor="let person of allPerson; index as i ; trackBy: personTrackByfn">

----------------
</ tr>
```

trackBy with ng-template

ngFor is a directive that you can use in your Angular templates to loop through lists and arrays of data in order to display it.
The use of trackBy it's a performance optimization and is usually not needed by default, it's in principle only needed if running into performance issues.
Suppose we have some data coming from some API request into the collection and we need to change the data over the web page using ngFor directive. In this case, Angular 7 will remove all the DOM elements that associated with the data and will create them again in the DOM tree, even the same data is coming (as shown in below screenshot). That means a lot of Dom manipulations will happen if a large amount of data coming from the API.

- The solution is, we can use trackBy function, which will be helpful for Angular to track the items which have been added or removed.
- TrackBy function will take two arguments first is index and second is current item and we can return the unique identifier as a return argument.

```
<ng-template ngFor let-person [ngForOf]=" allPerson" let-i="index"
[ngForTrackBy]="personTrackByFn" >

----------------
</ ng-template >
```