Angular 8 : Online Class

Class – 9

By: Sahosoft Solutions

Presented by : Chandan Kumar

NgIf with Async Pipe

Short Introduction Pipe:

AngularJS 1.x has filters which are used for many common uses, like formatting dates, string display in upper or lower case etc. These filters are known as "Pipes" in Angular 2 and higher.

Pipes allow us to change the data inside the template. Normally, a pipe takes the data and transforms this input to the desired output. There are many built-in pipes in Angular 2 and higher. Like **Date Pipes, Uppercase Pipe, Lowercase Pipe, Percentages/Number Pipe, Currency Pipe, Slice Pipe, Async Pipe etc**

NgIf with Async Pipe

Short Introduction Async Pipe:

Angular 2 and higher provides a new special pipe "Async". It allows us to bind the value of a variable directly with template which arrives asynchronously. It provides great ability for working with promises and observables. If we have Observable or Promise instance then we use it directly with AsyncPipe using directive such as NgFor, NgIf and NgSwitch. AsyncPipe belongs to angular common module. AsyncPipe subscribes to Observable or Promise and returns the latest data.

NgIf with Async Pipe

we will use NgIf with AsyncPipe. We can store conditional result in a variable. This approach is useful when initially the value is null or undefined and we want to avoid exception. As we know that to avoid exception we can access value of an object using safe- nevigation operator?. and these types of scenarios arise when we fetch value over HTTP. Before getting the HTTP response the object may be undefined or null. In case of NgIf angular provides better approach to handle exception without using safe- nevigation operator ?.

Now simply to store conditional result in a variable, we can do as follows without using async.

NgIf with dynamic then and else block

More than one <ng-template> for then and else block (NgIf-Then-Else) :

Short Intro @ViewChild() :

@ViewChild() is a decorator. @ViewChild() decorator can be used to get the first element or the directive matching the selector from the view DOM.

we will use NgIf with then and else. then template is the inline template of NgIf and when it is bound to different value then it displays <ng-template> body having template reference value as then value.

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.

NgFor (*ngFor)

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 such as (obsPersons | async) .

NgForOf with HTML Elements

NgForOf directive is used with HTML elements as following.

```
<li *ngFor="let item of items; index as i; even as isEven; odd as isOdd; first as isFirst; last as isLast;
trackBy: trackByFn">  ------</li>
```

NgForOf with <ng-template>

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

```
<ng-template ngFor let-item [ngForOf]="items" let-i="index" let-isEven="even" let-isOdd="odd" let-isFirst="first" let-isLast="last" [ngForTrackBy]="trackByFn">  <li> ------ </li></ng-template>
```

index, even and odd with HTML Elements

we will use NgForOf with index, even and odd local variables using element.

index, even and odd with ng-template

we will use NgForOf with index, even and odd local variables using element.