Angular ngFor Directive

4 Comments / 8 minutes of reading / March 9, 2023

← Child Component in Angular
Angular Tutorial
ngSwitch Directive →

Angular ngFor directive iterates over a collection of data like an array, list, etc, and creates an HTML element for each of the items from an HTML template. It helps us to build lists or tables to display tabular data in a nice way. In this tutorial, we will look at the syntax and how to use `ngFor` to display a list of movies using example code. The `ngFor` also exports several local variables like `Index`, `First`, `Last`, `odd`, `even` & `trackby` .etc. In this article, we will learn the following

1. Use ngFor in a List Box
2. Learn to use it in a Table
3. Use it to display a nested array.
4. How to assign of exported values to the local variable
5. Format the odd & even rows of a table by assigning different classes to them.
6. Find the index of each element in the collection
7. Learn to use `trackBy` clause, which enhances the performance

Table of Contents

Syntax of ngFor

The syntax for the `ngFor` is as shown below

```

<html-element ngFor="let <item> of <items>;”>
    <html-Template></html-Template>
</html-element>

```

<html-element>:
is the element on which we apply `ngFor` directive. it repeats the `<html-element> .. </html-element>` for each item of the collection.

***ngFor** :
The syntax starts with `*ngFor`. The `*` here tells us that ngFor is an Angular structural directive.

let <item> of <items>;
`item` is the *Template input variable*. It represents the currently iterated item from the `<items>`. `<items>` is a collection, which we need to show to the user. It is usually a property on your component class and can be anything that you can iterate over. (Usually an array)

The scope of the `item` is within the `<html-element>..</html-element>`. You can access it anywhere within that, but not outside of it.

ngFor Example

Now let us see how to use ngFor using an example.

Create a new angular Application. If you are new to angular, then you should read Angular create a new project. We are using bootstrap 4 to style our application. Hence you can add the following line to `index.html`

```

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstra

```

Open the `app.component.ts` and add the following code. The code contains a list of Top 10 movies. Let us build a template to display the movies using `ngFor`.

```

import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent {
  title: string ="Top 5 Movies" ;


  movies: Movie[] =[

  {title:'Zootopia',director:'Byron Howard, Rich Moore',cast:'Idris Elba, Ginnifer Goodwin, Ja
  {title:'Batman v Superman: Dawn of Justice',director:'Zack Snyder',cast:'Ben Affleck, Hen
  {title:'Captain American: Civil War',director:'Anthony Russo, Joe Russo',cast:'Scarlett Joha
  {title:'X-Men: Apocalypse',director:'Bryan Singer',cast:'Jennifer Lawrence, Olivia Munn, Os
  {title:'Warcraft',director:'Duncan Jones',cast:'Travis Fimmel, Robert Kazinsky, Ben Foster',
]


}

class Movie {
  title : string;
  director : string;
  cast : string;
  releaseDate : string;
}

```

Source code

Using ngFor

To use ngFor ,

1. Create a block of HTML elements, which can display a single movie.
2. Use the ngFor to repeat the block for each movie in the movies.

Open the app.component.html and add the following code.

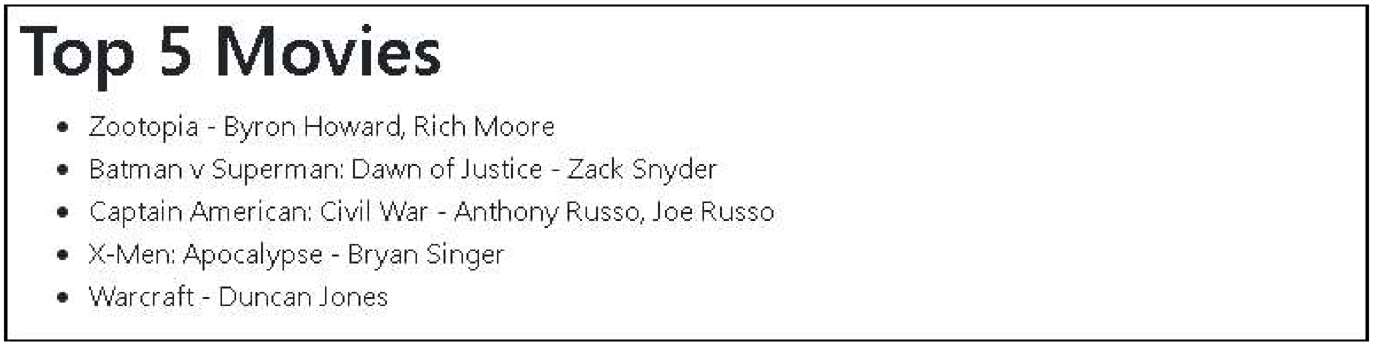
```

<h1> {{title}} </h1>

  <ul>
    <li *ngFor="let movie of movies">
      {{ movie.title }} - {{movie.director}}
    </li>
  </ul>


```

Source code

We use the ul to display the movies. The li element displays a single movie. We need to repeat the li for each movie. Hence we apply the ngFor on the li element.

`let movie of movies` will iterate over the `movies` collection, which is a property on the component class. `movie` is the *Template input variable*, which represents the currently iterated movie from the collection. We use Angular Interpolation to display the movie title & name of the director

Here is the output



Top 5 Movies

- Zootopia - Byron Howard, Rich Moore
- Batman v Superman: Dawn of Justice - Zack Snyder
- Captain American: Civil War - Anthony Russo, Joe Russo
- X-Men: Apocalypse - Bryan Singer
- Warcraft - Duncan Jones

The Angular generates the following code. You can see `li` element for every movie.

```

<ul _ngcontent-gop-c0="">
  <li _ngcontent-gop-c0=""> Zootopia - Byron Howard, Rich Moore </li>
  <li _ngcontent-gop-c0=""> Batman v Superman: Dawn of Justice - Zack Snyder </li>
  <li _ngcontent-gop-c0=""> Captain American: Civil War - Anthony Russo, Joe Russo </li>
  <li _ngcontent-gop-c0=""> X-Men: Apocalypse - Bryan Singer </li>
  <li _ngcontent-gop-c0=""> Warcraft - Duncan Jones </li>
</ul>

```

Similarly, you can use the `table` element to display the movies as shown below. Here we need to repeat the `tr` element for each movie. Hence apply the directive on `tr`

```

<div class='panel panel-primary'>
    <div class='panel-heading'>
        {{title}}
    </div>

    <div class='panel-body'>
        <div class='table-responsive'>
            <table class='table'>
                <thead>
                    <tr>
                        <th>Title</th>
                        <th>Director</th>
                        <th>Cast</th>
                        <th>Release Date</th>
                    </tr>
                </thead>
                <tbody>
                    <tr *ngFor="let movie of movies;">
                        <td>{{movie.title}}</td>
                        <td>{{movie.director}}</td>
                        <td>{{movie.cast}}</td>
                        <td>{{movie.releaseDate}}</td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</div>

```

Source Code

Here is the output

Nested Array

The following example shows how to use the `ngFor` in a nested array. The `employees` array has nested `skills` array.

```
employees = [
  {
    name: "Rahul", email: "rahul@gmail.com",
    skills: [{ skill: 'Angular', exp: '2' },{ skill: 'Javascript', exp: '7' },{ skill: 'TypeScript', ex
    ]
  },
  {
    name: "Sachin", email: "sachin@gmail.com",
    skills: [{ skill: 'Angular', exp: '1' },{ skill: 'Android', exp: '3' },{ skill: 'React', exp: '2' }
    ]
  },
  {
    name: "Laxmna", email: "laxman@gmail.com",
    skills: [{ skill: 'HTML', exp: '2' },{ skill: 'CSS', exp: '2' },{ skill: 'Javascript', exp: '1' }
    ]
  }
]
```

Source code

Inside the main loop, use the local variable `employee` to get the list of `skills` and loop through it using `*ngFor="let skill of employee.skills;"`

```
<div class='card'>
  <div class='card-header'>
    <p>Nested Array</p>
  </div>

  <div class='table-responsive'>
    <table class='table table-bordered table-sm '>
      <thead class="thead-dark">
        <tr>
          <th>Name</th>
          <th>Mail ID</th>
          <th>Skills</th>
        </tr>
      </thead>
    <tbody>
      <tr *ngFor="let employee of employees;">
        <td>{{employee.name}}</td>
        <td>{{employee.email}}</td>
        <td>
          <table class='table table-sm '>
            <tbody>
              <tr *ngFor="let skill of employee.skills;">
                <td>{{skill.skill}}</td>
                <td>{{skill.exp}}</td>
              </tr>
            </tbody>
          </table>

        </td>
      </tr>
    </tbody>
  </table>
</div>
</div>

```

Source code

Local Variables

ngFor exposes several values, which help us to fine-tune display. We assign these values to the local variable and use it in our template

The list of exported values provided by ngFor directive

- index: number : The zero-based index of the current element in the collection.
- count: number : The total no of items in the collection
- first: boolean : True when the item is the first item in the collection.
- last: boolean : Is set to True , when the item is the last item in the collection.
- even: boolean : True when the item has an even index in the collection.
- odd: boolean : is set to True when the item has an odd index in the collection.

Finding the Index

To Find the index, we create another local variable `i` and use the `let` to make it equal to `index`.

```

let i=index;

```

The following code shows the list of movies along with the index.

```

<tr *ngFor="let movie of movies; let i=index;">
   <td> {{i}} </td>
   <td>{{movie.title}}</td>
   <td>{{movie.director}}</td>
   <td>{{movie.cast}}</td>
   <td>{{movie.releaseDate}}</td>
</tr>

```

Source Code

Formatting odd & even rows

We can use the odd & even values to format the odd & even rows alternatively. To do that create two local variables `o` & `e`. Assign the values of odd & even values to these variables using the `let` statement. Then use the ngClass to change the class name to either odd or even. The example code is shown below

```

<tr *ngFor="let movie of movies; let i=index; let o= odd; let e=even;"
[ngClass]="{ odd: o, even: e }">
   <td> {{i}} </td>
   <td>{{movie.title}}</td>
   <td>{{movie.director}}</td>
   <td>{{movie.cast}}</td>
   <td>{{movie.releaseDate}}</td>
</tr>

```

Source Code

Add the appropriate background color to the `odd` and `even` classes as shown below in `app.component.css`

```

.even { background-color: azure; }
.odd { background-color: floralwhite; }

```

Source Code

First and the Last element of a list

Similarly, you can use the `first` & `last` values to style the first & last element. The code below will add CSS classes `first` & `last` to the first and last movie using the ngClass.

```

<div class='table-responsive'>
  <table class='table table-bordered table-sm '>
    <thead class="thead-dark">
      <tr>
        <th>Index</th>
        <th>Title</th>
        <th>Director</th>
        <th>Cast</th>
        <th>Release Date</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let movie of movies; let i=index; let first= first; let last=last;" [ngClass]=
        <td> {{i}} </td>
        <td>{{movie.title}}</td>
        <td>{{movie.director}}</td>
        <td>{{movie.cast}}</td>
        <td>{{movie.releaseDate}}</td>
      </tr>
    </tbody>
  </table>
</div>

```

Source Code

Remember to add the CSS classes to app.component.css

```

.first { background-color: azure; }
.last { background-color: floralwhite; }

```

Source Code

Track By

The angular includes Track By clause, just like AngularJS did. Track By clause allows you to specify your own key to identify objects.

Angular uses the *object identity* to compare the elements in the collection to the DOM nodes. Hence when you add an item or remove an item, the Angular will track it and update only the modified items in the DOM. It does not render the entire list.

But this fails if we update the list from the backend server. That is because the retrieved objects cannot be compared with the existing objects in the list as the reference has changed. The Angular simply removes these elements from DOM and recreates the new elements from the new data. This has a huge performance implication.

Angular trackBy clause eliminates this problem, by telling angular how to identify similar elements. The Angular will use the value returned by the trackBy function to match the elements returned by the database and update the DOM Elements without recreating them.

We should always specify the primary key or unique key as the `trackBy` clause.

Example

In our movie list example, let us make the title of the movie as the identifier.

```

<tr *ngFor="let movie of movies; trackBy:trackByFn;">
   <td>{{movie.title}}</td>
   <td>{{movie.director}}</td>
   <td>{{movie.cast}}</td>
   <td>{{movie.releaseDate}}</td>
</tr>

```

Source Code

In the Component Class create a `trackByFn`. It gets the index and the current item as its argument. It should return the unique id

```

 trackByFn(index, item) {
   return item.title;
 }

```

Source Code