Angular Trackby To Improve NgFor Performance

1 Comment / March 9, 2023 / 4 minutes of reading

← ngStyle | Angular Tutorial | Custom Directive →

Angular Trackby option improves the Performance of the ngFor if the collection has a large no of items and keeps changing. Learn why we need it and how to use it to improve the performance of the `ngFor`.

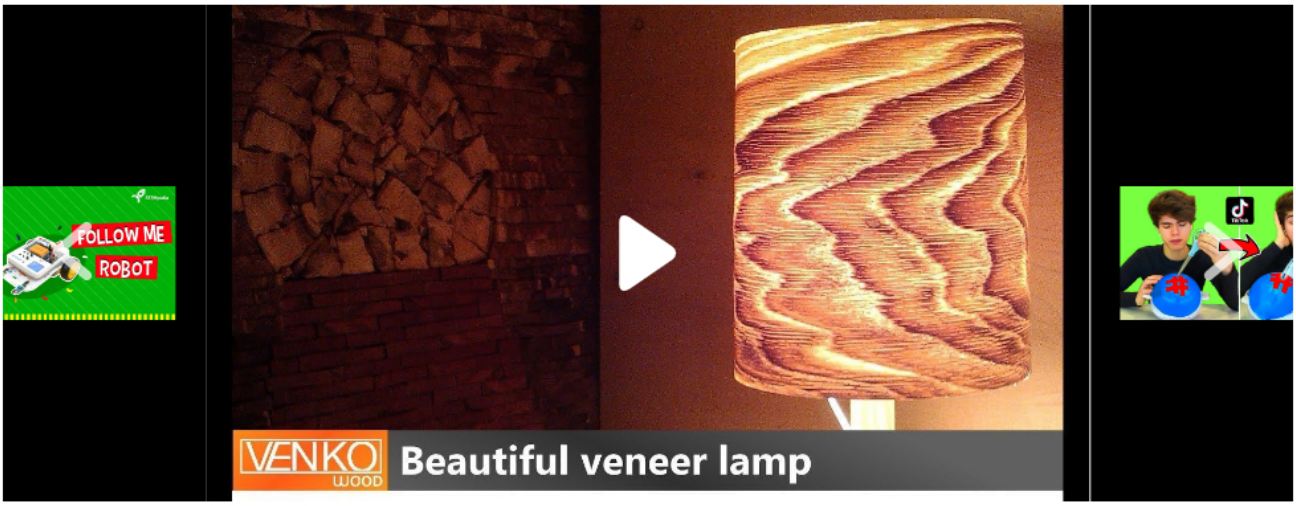
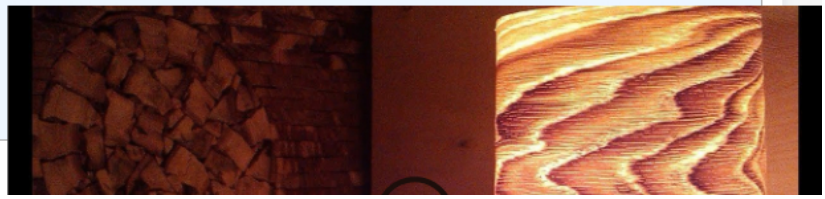
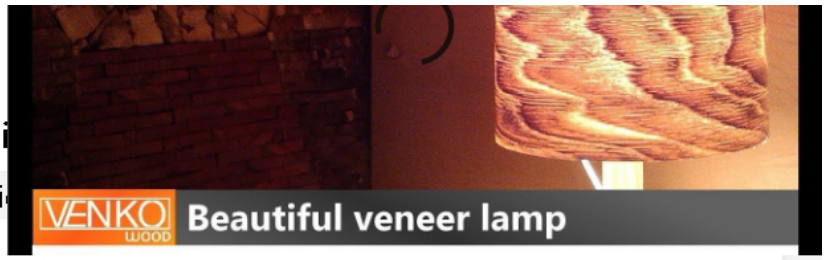


Table of Contents



Trackby in ngFor



We use `ngFor` to display a iterable items li
the following code iterates over the `movi`
`ul`

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

The Angular creates a `li` element for each movie. So if there are `n` number of movies, the angular inserts the `n` number of `li` nodes into the DOM

But the data will not remain constant. The user will add a new movie, delete a movie, sort the list in a different order, or simply refresh the movie from the back end. This will force the angular to render the template again.

The easiest way to achieve that is to remove the entire list and render the DOM again. But this is inefficient and if the list is large it is a very expensive process.

To avoid that the Angular uses the *object identity* to track 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.

But if you refresh the entire list from the back end, it will replace the objects in the movie collection with the new objects. Even if the movies are the same, Angular will not be able to detect as the object references have changed. Hence it considers them new and renders them again after destroying the old ones.

The following example shows what happens when we refresh the entire list. The App

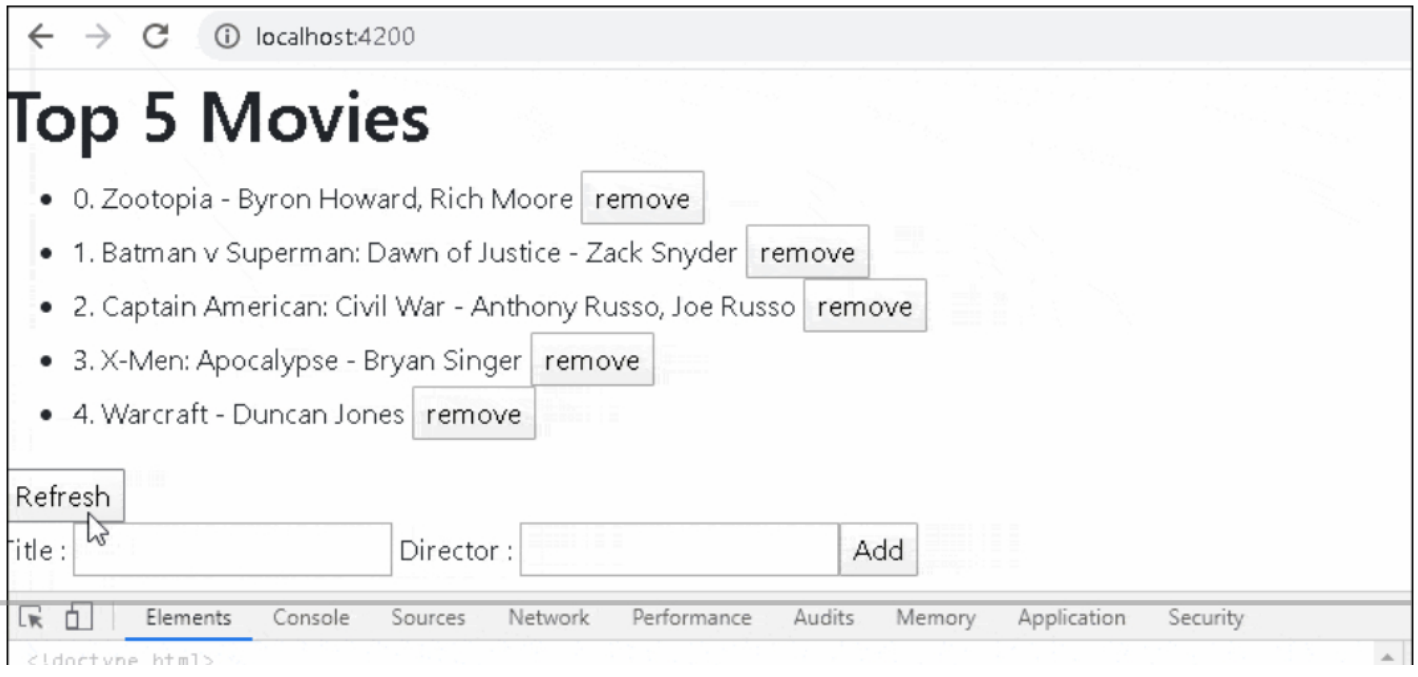
displays the list of movies. it has option to add a movie, remove a movie and refresh the entire movie.

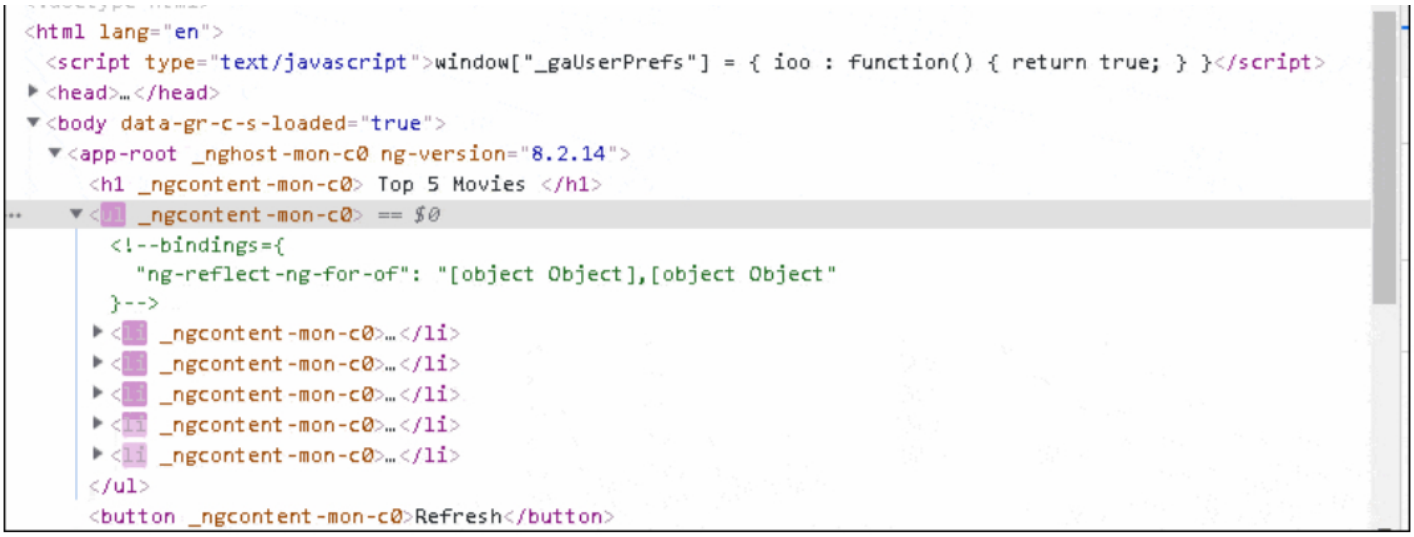
```
import { Component, OnInit } from '@angular/core';

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

  movies=[];

  mTitle:string="";
  mDirector:string="";

  ngOnInit() {
    this.Refresh();
  }

  remove(i) {
    this.movies.splice(i,1);
  }

  addMovie() {
    this.movies.push({ title: this.mTitle, director: this.mDirector})
    this.mTitle=""
    this.mDirector=""
  }

  Refresh() {
    console.log("refresh")
    this.movies = [
      { title: 'Zootopia', director: 'Byron Howard, Rich Moore'},
      { title: 'Batman v Superman: Dawn of Justice', director: 'Zack Snyder'},
      { title: 'Captain American: Civil War', director: 'Anthony Russo, Joe Russo'},
      { title: 'X-Men: Apocalypse', director: 'Bryan Singer'},
      { title: 'Warcraft', director: 'Duncan Jones'},
    ]
  }
}

class Movie {
  title: string;
  director: string;
```

```
}
```



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

<ul>
  <li *ngFor="let movie of movies; let i=index;trackBy: trackByFn;">
    {{i}}. {{ movie.title }} - {{movie.director}} <button (click)="remove(i)">remove</button
  </li>
</ul>

<button (click)="Refresh()">Refresh</button> <br>

Title    : <input type="text" [(ngModel)]="mTitle">
Director  : <input type="text" [(ngModel)]="mDirector">
<button  (click)="addMovie()">Add</button>
```





You can see from the above example, that Angular renders the entire DOM every time we click on refresh.

Trackby

We can solve this problem by providing a function to the `trackBy` option that returns a unique id for each item. The `ngFor` will use the unique id returned by the `trackBy` function to track the items. Hence even if we refresh the data from the back end, the unique id will remain the same and the list will not be rendered again.

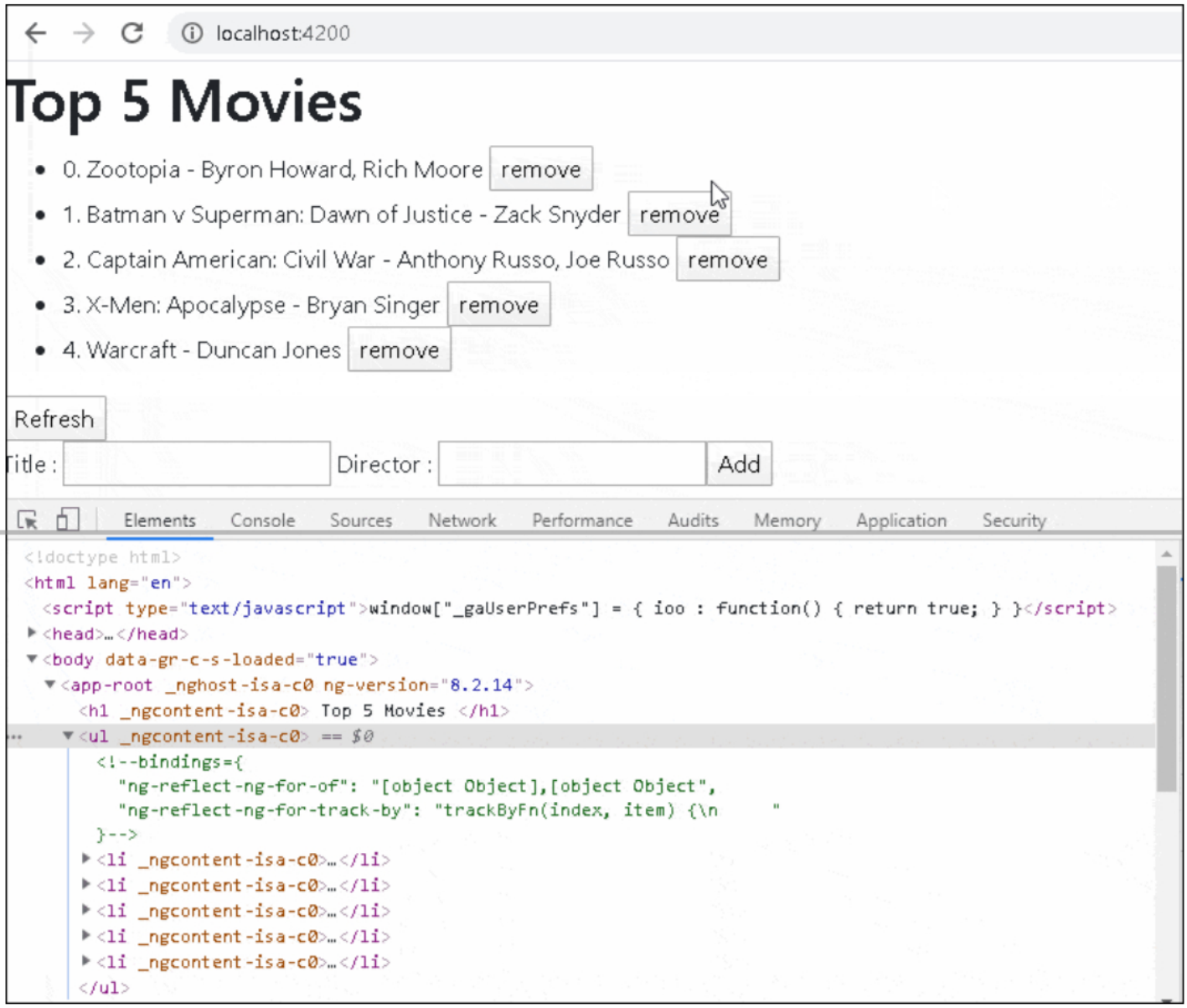
The `trackBy` takes a function that has two arguments: `index` and the current `item`. It must return a `id` that uniquely identifies the item. The following example returns the `title` as the unique id.

```

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

```

In the template assign the newly created `trackByFn` to `trackBy` option in the `ngFor` statement.

```
<li *ngFor="let movie of movies; let i = index;trackBy: trackByFn;">
```

```
<li *ngFor="let movie of movies; let i=index;trackBy: trackByFn;" >
```



Trackby multiple fields

You can also trackby multiple fields as shown below

```
<li *ngFor="let movie of movies; let i=index;trackBy: trackByFnMultipleFields;">
```

```
trackByFnMultipleFields(index, item) {
  return item.title + item.director;
}
```

```
5
6
```

Reference

1. Change Propagation in ngFor

Read More

1. Angular Tutorial
2. Angular Directives
3. ngFor
4. ngSwitch
5. ngIf
6. ngClass
7. ngStyle
8. ngFor Trackby
9. Custom Directive

← ngStyle | Angular Tutorial | Custom Directive →

1 thought on “Angular Trackby to improve ngFor Performance”

JAMSHED AHMAD

DECEMBER 14, 2020 AT 10:07 PM

nice explanation.

Reply

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Post Comment »

This site uses Akismet to reduce spam. Learn how your comment data is processed.