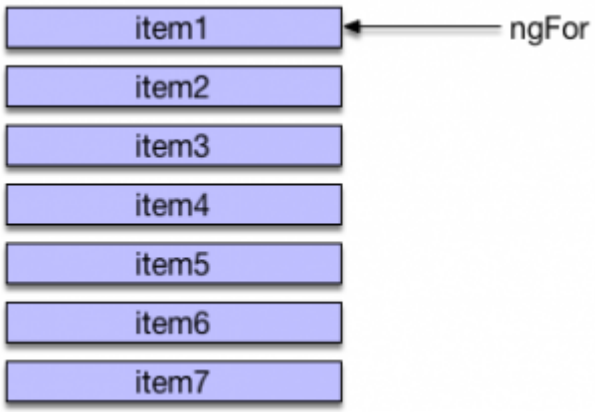
Hoy vamos a hablar de la directiva Angular ngFor y las opciones que soporta. La directiva ngFor es la que se encarga de presentar una lista de elementos en pantalla de una forma sencilla combinando el concepto de bucle y plantilla . Esta directiva soporta muchas opciones, vamos a verlas



Angular ngFor directo

El primer paso va a ser crear un componente que nos devuelva una lista de cadenas (la lista mas básica a manejar). Este componente contendrá un array de Javascript.

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

@Component({
selector: 'app-hola5',
templateUrl: './hola5.component.html',
styleUrls: ['./hola5.component.css']
})
export class Hola5Component implements OnInit {

lista:string[]=["hola","que","tal","estas"];

constructor() { }

ngOnInit() {
```

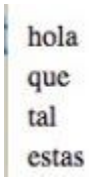
```
}

}
```

Vamos a ver como se construye una primera versión de la plantilla que imprima la lista.

```
<table>
        <tr *ngFor="let item of lista">
    <td>
            {{item}}
        </td>
    </tr>
</table>
```

Este es el ejemplo más sencillo y nos imprime la lista de cadenas como una tabla utilizando la directiva de ngFor



Angular ngFor e indices

La etiqueta ngFor soporta bastantes mas opciones , por ejemplo podemos acceder al indice del elemento e imprimirlo . Recordemos que en JavaScript los arrays comienzan en cero.

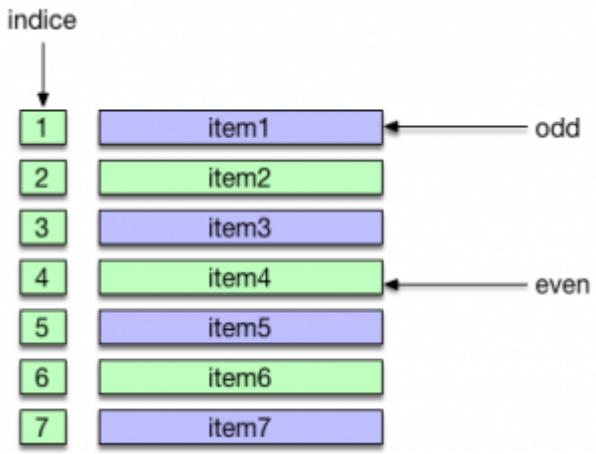
```
<table>
        <tr *ngFor="let item of lista;let indice=index">
                <td> {{indice+1}}
                        <td>{{item}}</td>
                </tr>
        </table>
```

En este caso hemos declarado una variable asociada al bucle y la imprimimos en una columna incrementado su valor.

1 hola
2 que
3 tal
4 estas

Angular filas : pares impares

Vamos a ver ahora como gestionar el manejo de filas pares y filas impares de tal forma que podamos dibujar cada una de un color diferente.



El primer paso es generar los estilos a nivel de la css del componente.

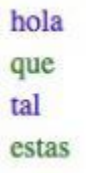
```
.azul {
color:blue;
}

.verde {
color:green;
}
```

Una vez tenemos esto , nos queda ver como modificar la estructura de angular ngFor para que soporte estas opciones.

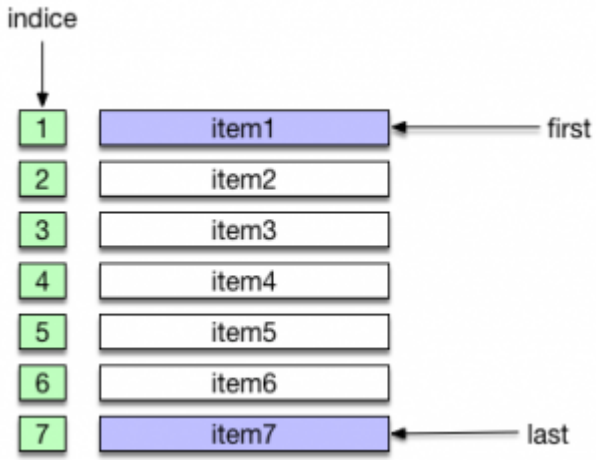
```
<table>
<tr *ngFor="let item of lista;let impar = odd;let par = even;"
[ngClass]="{azul:par,verde:impar}">
<td>{{item}}</td>
</tr>
</table>
```

El resultado en pantalla iluminara las filas con colores alternos:



ngFor first y last

Angular soporta también la posibilidad de seleccionar el primer elemento y el último de una lista.

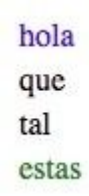


Vamos a verlo en código:

```
<table>
        <tr *ngFor="let item of lista;let primero = first;let ultimo =
last;" [ngClass]="{azul:primero,verde:ultimo}">
                <td>{{item}}</td>
```

```
        </tr>
</table>
```

El resultado lo vemos en la página HTML:



Angular ngFor Objetos y TrackBy

Cuando trabajamos con Angular, una de las cosas más comunes que hacemos es recorrer listas de elementos para mostrarlas en la vista. La directiva *ngFor nos permite hacerlo de manera sencilla y eficiente. Sin embargo, en ciertas situaciones, especialmente cuando trabajamos con grandes listas o datos que se actualizan frecuentemente, necesitamos optimizar el proceso de renderizado. Aquí es donde entra en juego la directiva trackBy.

Vamos a ver cómo usar *ngFor con trackBy en un caso sencillo donde trabajamos con una clase Persona que tiene tres atributos: dni, nombre y apellidos. Veremos cómo utilizar el dni como identificador único para optimizar la manera en que Angular maneja los cambios en la lista.

Definiendo nuestra clase Persona

Empecemos creando la clase Persona, la cual tendrá tres propiedades:

```
export class Persona {
  constructor(
    public dni: string,
    public nombre: string,
    public apellidos: string
  ) {}
```

```
}
```

Aquí tenemos una clase bastante simple. Cada persona tiene un dni, que será único para cada una, y también tiene un nombre y apellidos.

Usando *ngFor y trackBy

Ahora, en nuestro componente de Angular, vamos a trabajar con una lista de personas. Usaremos *ngFor para iterar sobre esa lista y mostrarla en la vista. Para asegurarnos de que Angular gestione los cambios en la lista de manera eficiente, vamos a implementar la función trackBy, que utilizará el dni de cada persona como clave única.

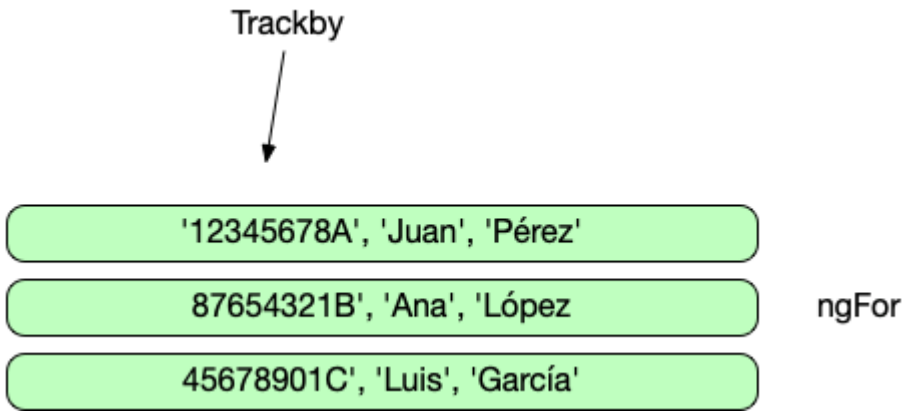
```
import { Component } from '@angular/core';
import { Persona } from './persona.model';

@Component({
  selector: 'app-persona-list',
  template: `
    <ul>
      <li *ngFor="let persona of personas; trackBy: trackByDni">
        {{ persona.dni }} - {{ persona.nombre }} {{ persona.apellidos
}}
      </li>
    </ul>
  `
})
export class PersonaListComponent {
  personas: Persona[] = [
    new Persona('12345678A', 'Juan', 'Pérez'),
    new Persona('87654321B', 'Ana', 'López'),
    new Persona('45678901C', 'Luis', 'García')
  ];
```

```
  // trackBy function to track by dni
  trackByDni(index: number, persona: Persona): string {
    return persona.dni;
  }
}
```

¿Qué está pasando aquí?

- *ngFor: Esta directiva recorre nuestra lista de personas y crea un <li> para cada persona. Dentro del li, mostramos el dni, nombre y apellidos de cada una.
- trackBy: Aquí es donde empieza la optimización. Normalmente, Angular sigue la lista por el índice de los elementos. Esto funciona bien cuando los cambios en la lista no son frecuentes, pero cuando hay cambios constantes o la lista es grande, podemos mejorar el rendimiento diciéndole a Angular cómo identificar de manera única a cada elemento. En nuestro caso, estamos usando el dni de la persona como clave única, lo cual es ideal porque es un valor que no cambia y es exclusivo de cada persona.



¿Por qué usar trackBy?

Cuando Angular detecta un cambio en una lista que está siendo iterada por *ngFor, por defecto, renderiza de nuevo todos los elementos de la lista. Esto puede ser ineficiente, sobre todo si solo cambia un elemento en una lista muy grande. Con trackBy, le decimos a Angular que identifique los elementos de la lista por un atributo único (en este caso el dni), y de esta manera, solo se actualizan los elementos que realmente han cambiado.

Esto es útil en casos como:

- Listas grandes de elementos.
- Listas que cambian frecuentemente por actualizaciones o filtros.
- Mejorar el rendimiento cuando solo un pequeño subconjunto de la lista cambia.

Beneficios de usar trackBy

1. Rendimiento: Usar trackBy evita que Angular tenga que reconstruir todos los elementos de la lista si un solo elemento cambia. Esto puede ser crítico en aplicaciones con gran cantidad de datos.
2. Identificación única: Con trackBy, aseguramos que Angular maneje correctamente cada elemento, especialmente si la lista contiene elementos que podrían tener índices similares, pero diferentes identificadores únicos.
3. Mantenimiento del estado del DOM: Los elementos que no cambian mantienen su estado, lo que es útil si, por ejemplo, hay formularios o elementos interactivos en cada fila.

Conclusión

El uso de trackBy junto con *ngFor es una técnica sencilla, pero potente, que puede mejorar significativamente el rendimiento de nuestra aplicación Angular cuando trabajamos con listas grandes o dinámicas. En este ejemplo, hemos utilizado una clase Persona con dni como clave única, pero podrías aplicar este enfoque a cualquier tipo de dato que manejes en tus listas.

Otros artículos relacionados:

1. Angular 5 Hello World y su funcionamiento
2. React vs Angular 2 , frameworks vs librerias
3. Angular resolve y manejo de rutas asíncronas
4. Vuejs una alternativa a React y Angular
5. Desarrollo Web con Angular.js (nº1 en Amazon.es programación)

Externos:

El api de Angular.js