Introducción

Angular ngIf : Cuando trabajamos con Angular, una de las necesidades más comunes es mostrar u ocultar elementos en la página de acuerdo a ciertas condiciones. Para esto, Angular nos ofrece ngIf, una directiva súper útil que nos permite controlar qué se ve y qué no en nuestra aplicación. Pero lo interesante de ngIf es que no solo oculta visualmente los elementos, sino que realmente los elimina o los añade al DOM, lo que ayuda a mejorar el rendimiento de nuestra aplicación.

Hoy vamos a hablar Angular ngIf como directiva y las opciones que soporta. Para ello en un proyecto de Angular nos vamos a construir un componente sencillo que le denominaremos nota. Este componente albergará la variable nota del examen de un alumno.

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

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

  nota:number;

  constructor() {

    this.nota=5;

  }

  ngOnInit() {
```

```
  }

}
```

Una vez tenemos el componente construido nos queda ver como quedaría la plantilla que muestra la nota. En este caso de partida es relativamente sencillo simplemente imprimimos la nota :

```
<p>Nota:{{nota}}</p>
```

El interface de usuario nos mostrará la nota:

> Nota:5

Manejo de Eventos

Es momento de añadir al interface dos botones que nos permitan modificar los valores de la nota e incrementar o decrementarlo según nuestras necesidades:

```
<p>Nota :{{nota}}
<input type="button" value="+" (click)="nota=nota+1">
<input type="button" value="-" (click)="nota=nota-1">
</p>
```

El interface se actualiza y podemos cambiar los valores de la nota:

Nota:7

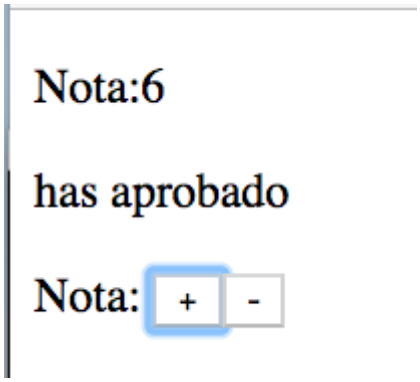
Nota: + -

Angular NgIf

Es momento de usar la directiva ngIf para que nos imprima dependiendo del valor numérico de la nota una calificación. Podríamos tener como punto de partida:

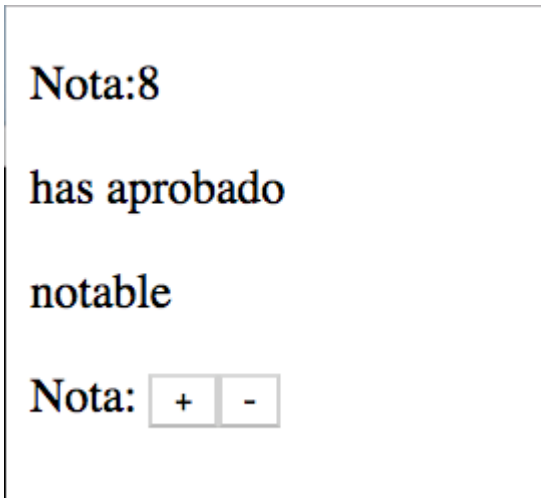
```
<p *ngIf="nota>=5">has aprobado</p>
```

Esto nos imprimirá en la página que hemos aprobado si la nota es superior a 5:



Podemos hacer una operación similar en el caso de que la nota este entre 7 y 9 de tal forma que se imprima notable apoyándonos en un operador and (&&).

```
<p *ngIf="nota>=7 && nota<9">notable</p>
```



Angular ngIf

Modificamos el componente para que incluya dos propiedades adicionales y podamos escribir condiciones más complejas:

```
export class NotaComponent implements OnInit {

  nota:number;
  asignatura:string;
  edad:number;
  constructor() {
    this.asignatura="matematicas";
    this.nota=5;
    this.edad=16;
  }

  ngOnInit() {
  }

}
```

De esta forma podemos construir condiciones como las siguientes:

```
<p *ngIf="(nota>=9 && asignatura=='matematicas') ||
(edad==16)">estudia aeronautica</p>
```

En este caso si la nota es superior a 9 y la asignatura es matemáticas o ha aprobado el examen con 16 años , le recomendamos estudiar Areonaútica.

Nota:9

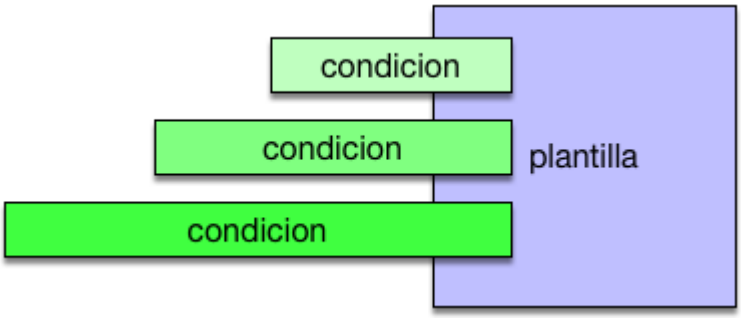
has aprobado

estudia aeronautica

Nota: + -

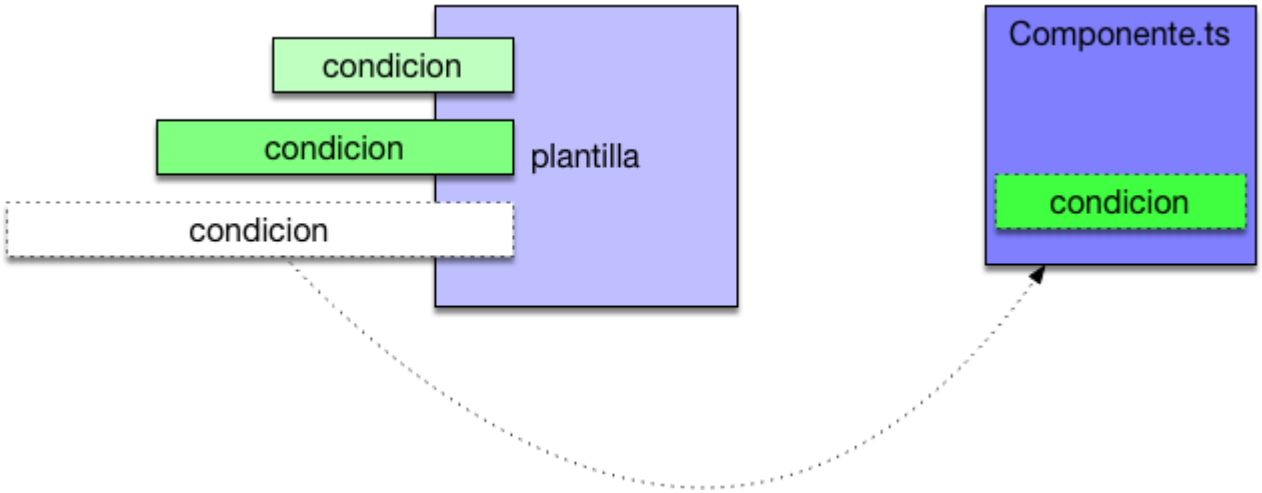
Condiciones y Comportamiento

Sin embargo cuando las condiciones se complican suele ser mejor proceder de otra forma ya que al final las condiciones no quedan claras debido a su tamaño.



Angular ngIf y Componentes

En vez de tener un ngIf inmenso con operadores como && || o ! es preferible definir esa condición a nivel del propio componente de tal forma que el interface de usuario quede mucho más cómodo.



Vamos a verlo:

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

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

     nota:number;
     asignatura:string;
     edad:number;
     constructor() {
        this.asignatura="matematicas";
        this.nota=5;
        this.edad=16;
    }
    esMuyListo():boolean {
        if((this.nota>=9 && this.asignatura=='matematicas') ||
(this.edad==16)) {
                return true;
        }
```

```
          else {
              return false;
          }
      }
}
```

En este caso hemos implementado una función interna al componente que decide si el alumno es muy listo o no. La pasamos a invocar desde la plantilla:

```
<p *ngIf="esMuyListo()">estudia aeronautica</p>
```

Nota:9

has aprobado

estudia aeronautica

Nota: + -

Siempre que la lógica de las sentencias ngIf se complique suele ser mucho más cómodo moverla a los componentes para un uso más claro.

Angular ngif y else

Uso de ngIf con else

Una de las características más útiles de ngIf es que no solo te permite mostrar u ocultar elementos según una condición, sino que también puedes definir qué hacer si esa condición no se cumple, utilizando la opción else. Esto hace que el código sea más limpio y fácil de leer, ya que evitas duplicar estructuras condicionales en tu plantilla.

Veamos un ejemplo básico. Supongamos que queremos mostrar un mensaje si un usuario está autenticado, y otro diferente si no lo está. Aquí es donde entra en juego el uso de else:

```
<div *ngIf="estaAutenticado; else noAutenticado">
  <p>¡Bienvenido de nuevo, usuario!</p>
</div>

<ng-template #noAutenticado>
  <p>Por favor, inicia sesión para continuar.</p>
</ng-template>
```

En este ejemplo, si la variable `estaAutenticado` es verdadera, se mostrará el mensaje de bienvenida. Si es falsa, el bloque definido dentro de la plantilla ng-template con el alias #noAutenticado se encargará de mostrar el mensaje alternativo, indicando que el usuario debe iniciar sesión.

Este enfoque es más eficiente que duplicar elementos en el DOM y también ayuda a organizar mejor el flujo de la lógica en tu componente. Utilizando ngIf con `else` puedes manejar fácilmente diferentes escenarios y mejorar la legibilidad del código.

Otros artículos relacionados

- Java Stream if else y Optionals
- Java Predicate Interface y sus métodos
- Java Predicate Not y como usarlo
- Kotlin Ranges y sentencia de Control

Artículos externos

1. Angular