*ngif

El condicional **“If”** es un **“If”** en Javascript, en Java, en PHP, en Python o en cualquier lenguaje. Angular posibilita utilizar este condicionante embebido en el HTML para mostrar o no un elemento. Su sintaxis es algo particular, está compuesta por un **asterisco** seguido de las iniciales características de Angular **“ng”** y la palabra **“If”**.

```
<div *ngIf="isPlatzi">Hola, soy Platzi</div>
```

Si la condición dentro del **“If”** se cumple, se mostrará el `<div>` con el respectivo contenido dentro. De lo contrario, el usuario no verá dicho elemento en el navegador. En la condición del If puedes colocar **cualquier operador lógico**:

Operador	Símbolo
AND	&&
OR	||
Mayor	>
Menor	<
Igual	==
Distinto	!=
Mayor/Igual	>=
Menor/Igual	<=

If … else

Para usar un **else** en Angular, la sintaxis es algo especial. Debes crear un template en tu código HTML usando la etiqueta que provee Angular llamada `<ng-template>` con una **Variable de Template**, comenzando con `#`, para hacer referencia a este elemento desde tu **If**.

```
<div *ngIf="isPlatzi; else templateElse">Hola, soy Platzi</div>

<ng-template #templateElse>

    <div>No soy Platzi</div>

</ng-template
```

Si la condición del **If** no se cumple, **seguido de punto y coma**, se coloca la sentencia **else** haciendo referencia a `templateElse`, que es el nombre de la variable del template a mostrar en su lugar.

*ngFor

Al igual que con un **If**, Angular permite **iterar** un array de números, de cadenas de caracteres o de objetos usando **"*ngFor"**.Si tienes en tu componente un array de datos:

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  myArray: string[] = [
    'Platzi',
    'es',
    'genial!'
  ];
}
```

Puedes mostrar cada elemento iterando el array en un elemento HTML:

```
<ul>
    <li *ngFor="let str of myArray">
        {{ str }}
    </li>
</ul>
```

El ***ngFor** crea una variable temporal llamada **str** (o el nombre que más te guste) que contiene cada valor de **myArray**. Finalmente, utilizando una interpolación, muestras el valor de **str**.Quedando tu HTML de la siguiente manera:

```
<ul><li>Platzi</li><li>es</li><li>genial!</li></ul>
```

Índice de iteración

ngFor también cuenta con un **índice** con el número de iteraciones. Puedes acceder a este número agregando al **ngFor** `index as i` de la siguiente manera:

```
<ul>
    <li *ngFor="let str of myArray; index as i">
        {{ i }}. {{ str }}
    </li>
</ul>
```

Cada iteración contiene una variable `i` con el índice que le corresponde. Iniciando desde cero, da como resultado:

```
<ul>
    <li>0. Platzi</li>
    <li>1. es</li>
    <li>2. genial!</li>
</ul>
```

*ngFor para Arrays

Puedes utilizar ***ngFor** para iterar y mostrar cada propiedad de un objeto. Considera que en el componente tienes un array de objetos que representan a una persona:

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  peopleArray = [
    {
        firstname: 'Freddy',
        lastname: 'Vega',
```

```
        age: 35
    },
    {
        firstname: 'Nicolas',
        lastname: 'Molina',
        age: 29
    },
    {
        firstname: 'Ángela',
        lastname: 'Ocando',
        age: 30
    }
  ];
}
```

Itera este array en el HTML e imprimimos el valor de cada propiedad de la siguiente manera:

```
<ul *ngFor="let person of peopleArray">
    <li>Nombre: {{ person.firstname }}</li>
    <li>Apellido: {{ person.lastname }}</li>
    <li>Edad: {{ person.age }}</li>
</ul>
```

La variable `person` guarda temporalmente el objeto en cada iteración, pudiendo acceder a cada valor usando un **punto** seguido del **nombre de la propiedad**.

Tipado de objetos con interfaces

El array `peopleArray` puede contener cualquier cosa, y puede ocasionar comportamientos indeseados en tu aplicación. Puedes crear una interfaz de Personas para tipar los objetos del array y asegurar que todos tengas las mismas propiedades.

```
interface Person {
    firstname: string;
```

```
    lastname: string;
    age: number
}
```

Tipando el array de la siguiente manera para indicar que el array es de objetos del tipo Persona:

```
...
peopleArray: Person[] = [
    {
        firstname: 'Freddy',
        lastname: 'Vega',
        age: 35
    },
    ...
 ]
```

ngSwhich

Angular también ofrece la sentencia ***ngSwitch** y ***ngSwitchCase** para determinar el flujo de control de tu aplicación y qué elemento mostrar entre multiples elementos HTML. Además de utilizar un elemento default con ***ngSwitchDefault** en caso de que ninguna condición se cumpla.

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
    color: string = 'verde';
}
```

Ejemplo de *ngSwitchCase

```
<div [ngSwitch]="color">
    <p *ngSwitchCase="'azul'">
        El color el Azul
```

```
    </p>
    <p *ngSwitchCase="'verde'">
        El color el Verde
    </p>
    <p *ngSwitchCase="'rojo'">
        El color el Rojo
    </p>
    <p *ngSwitchDefault>
        No hay ningún color
    </p>
</div>
```

Código:

Html

```
<h1>ngSwitch</h1>
<input type="text" required [(ngModel)]="person.name" />
<div [ngSwitch]="person.name">
  <p *ngSwitchCase="'nicolas'">
    La persona es nicolas
  </p>
  <p *ngSwitchCase="'julian'">
    La persona es Julian
  </p>
  <p *ngSwitchCase="'camilo'">
    La persona es Camilo
  </p>
  <p *ngSwitchDefault>No hace match</p>
</div>
<!-- <p *ngIf="person.name === 'nicolas'">
  La persona es nicolas
```

```
</p>

<p *ngIf="person.name === 'julian'">
  La persona es Julian
</p>

<p *ngIf="person.name === 'camilo'">
  La persona es Camilo
</p> -->
<hr/>
<h1>*ngFor Objs</h1>
<div>
  <div *ngFor="let product of products">
    <img width="250" [src]="product.image" alt="image">
    <h2>{{ product.price }}</h2>
    <p>{{ product.name }}</p>
  </div>
</div>
<hr/>

<h1>*ngFor</h1>
<input type="text" required [(ngModel)]="newName" />
<button (click)="addName()">Add name</button>
<ul>
  <li *ngIf="names.length === 0">No hay nombres</li>
  <li *ngFor="let name of names; index as i">
    {{i}} {{ name }}
    <button (click)="deleteName(i)">Delete</button>
  </li>
</ul>
<hr/>
```

```
<h1>*ngIf</h1>

<input type="text" required [(ngModel)]="person.name" />

<p *ngIf="person.name === 'nicolas'">Soy Nicolas</p>

<p *ngIf="person.name === 'julian' && person.age === 18; else myBlock">Soy Julian</p>

<ng-template #myBlock>

  <p>Bloque de else</p>

</ng-template>

<hr/>

<h1>NgModel</h1>

<p>Nombre {{ person.name }}</p>

<input type="text" required #nameInput="ngModel" [(ngModel)]="person.name" />

<p>Valid: {{ nameInput.valid }}</p>

<p>Age {{ person.age }}</p>

<input type="number" max="18" min="10" required #ageInput="ngModel"
[(ngModel)]="person.age" />

<p>Valid: {{ ageInput.valid }}</p>

<hr/>

<h1>Eventos</h1>

<button [disabled]="btnDisabled">Enviar</button>

<button (click)="toggleButton()">Toggle Button</button>

<p>Edad {{ person.age }}</p>

<button (click)="increaseAge()">Age ++</button>

<div class="box" (scroll)="onScroll($event)">

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Similique, voluptatibus minima.
Deserunt cum ex aliquid quaerat doloremque totam autem excepturi dolorem? Reprehenderit
eius illo sequi consequuntur quaerat eaque ratione similique.</p>

</div>

<p>Nombre {{ person.name }}</p>

<input type="text" [value]="person.name" (keyup)="changeName($event)" />

<hr/>

<h1>Propiedades</h1>

<button [disabled]="btnDisabled">Enviar</button>
```

```
<input type="text" [value]="person.name" />
<progress max="100" [value]="person.age"></progress>
<img width="100" [src]="person.avatar" />
<hr/>
<h1>Hola mundo</h1>
<h2>{{ 'Hola Mundo'.repeat(5) }}</h2>
<p>3 + 3 = {{ 3 + 3 }}</p>
<h3>Hola, soy {{ name }} y tengo {{ age }}</h3>
<img src={{img}} alt="image">
```

Css

```
.box {
  height: 200px;
  width: 200px;
  overflow: auto;
  background: red;
}
```

App.componet.ts

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

import { Product } from './product.model';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
```

```
export class AppComponent {
  name = 'Nicolas';
  age = 18;
  img = 'https://www.w3schools.com/howto/img_avatar.png';
  btnDisabled = true;
  person = {
    name: 'Nicolas',
    age: 18,
    avatar: 'https://www.w3schools.com/howto/img_avatar.png'
  }
  names: string[] = ['Nico', 'Juli', 'Santi'];
  newName = '';
  products: Product[] = [
    {
      name: 'EL mejor juguete',
      price: 565,
      image: './assets/images/toy.jpg',
      category: 'all',
    },
    {
      name: 'Bicicleta casi nueva',
      price: 356,
      image: './assets/images/bike.jpg'
    },
    {
      name: 'Colleción de albumnes',
      price: 34,
      image: './assets/images/album.jpg'
    },
    {
      name: 'Mis libros',
```

```
    price: 23,
    image: './assets/images/books.jpg'
  },
  {
    name: 'Casa para perro',
    price: 34,
    image: './assets/images/house.jpg'
  },
  {
    name: 'Gafas',
    price: 3434,
    image: './assets/images/glasses.jpg'
  }
]

toggleButton() {
  this.btnDisabled = !this.btnDisabled;
}

increaseAge() {
  this.person.age += 1;
}

onScroll(event: Event) {
  const element = event.target as HTMLElement;
  console.log(element.scrollTop);
}

changeName(event: Event) {
  const element = event.target as HTMLInputElement;
  this.person.name = element.value;
```

```
  }

  addName() {
    this.names.push(this.newName);
    this.newName = '';
  }

  deleteName(index: number) {
    this.names.splice(index, 1);
  }
}
```