Directivas en ANGULAR 5

DIRECTIVAS

- Las plantillas en ANGULAR son:

  ➔ la parte HTML de los COMPONENTES:

  ➔La VISTA del Modelo MVC

  ➔Y SON DINÁMICAS:

  Cuando Angular las renderiza, TRANSFORMA EL DOM de acuerdo a las instrucciones dadas por las DIRECTIVAS.

- Una directiva es una **clase con un @Directive decorador.**
  - Esto sólo lo apreciaremos si hacemos nuestras propias directivas.
- Un componente, en realidad, es una **directiva con una plantilla.**
  - Pero se suelen tratar aparte.

Tipos de Directivas

- **ESTRUCTURALES**
  - Son las que **modifican el DOM**, su ***estructura***:
    - Añadiendo,
    - Borrando o
    - Manipulando elementos del mismo

```
<li *ngFor="let hero of heroes"></li>
<app-hero-detail *ngIf="selectedHero"> </app-hero-detail>
```

  - **Sólo se puede aplicar una directiva estructural por elemento del DOM.**
    - Es decir, ***en un mismo elemento no puede haber un *ngFor y *ngIf:***

```
<div *ngFor="let hero of heroes" *ngIf="hero.nombre=='Lucas'">
        …
</div>
```

  - Las **más típicas** son:
    - ***ngIf**
    - ***ngFor**
    - **ngSwitch**

Tipos de Directivas

- ¿USAR 2 DIRECTIVAS ESTRUCTURALES EN MISMO ELEMENTO?:
  - Por defecto, NO SE PUEDE.
  - SOLUCIONES:
    1. Buscar soluciones añadiendo de HTML y agrupando así.
    2. Usar el elemento de agrupamiento: <ng-container> de Angular.

       Ventajas de <ng-container>:

       - *No aparece en el DOM*, pues es de Angular sólo.
       - *No interfiere con las CSS* que pudiera haber; no le afectan.
       - *No es una directiva, ni clase, ni componente*
       - *Permite simular un *ngIf combinado con *ngFor como si estuviesen en el mismo elemento*.

Tipos de Directivas

- **EJEMPLO DE USO**:

```
<div>
  Pick your favorite hero
  (<label><input type="checkbox" checked (change)="showSad = !showSad">show sad</label>)
</div>
<select [(ngModel)]="hero">
  <ng-container *ngFor="let h of heroes">
    <ng-container *ngIf="showSad || h.emotion !== 'sad'">
      <option [ngValue]="h">{{h.name}} ({{h.emotion}})</option>
    </ng-container>
  </ng-container>
</select>
```

Tipos de Directivas

DE ATRIBUTO

- Cambian la apariencia o comportamiento de un elemento del DOM o componente existente.
- En las plantillas, parecen atributos HTML (*no lo son).*
  - *De ahí su nombre.*
- Las más típicas son:
  - ngStyle
  - ngClass
  - ngModel

```
<input [(ngModel)]="hero.name">
```

ngModel modifica el comportamiento, normalmente de un elemento *<input>* asignándole la **propiedad "value" a mostrar** y **respondiendo a eventos de cambio.**

DE COMPONENTES

- Son los componentes
- @Component

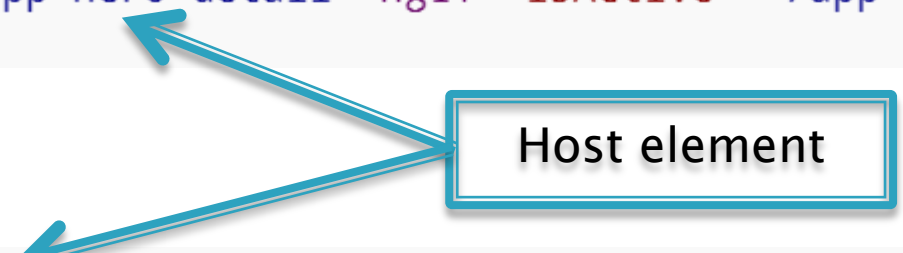
Directivas built-in más típicas

Directivas Estructurales

*ngIf

- Permite **añadir** o **eliminar elementos del DOM** según se cumpla o no una condición.
- Se aplica sobre un elemento del DOM que se llama **elemento HOST**.

```
<app-hero-detail *ngIf="isActive"></app-hero-detail>
```



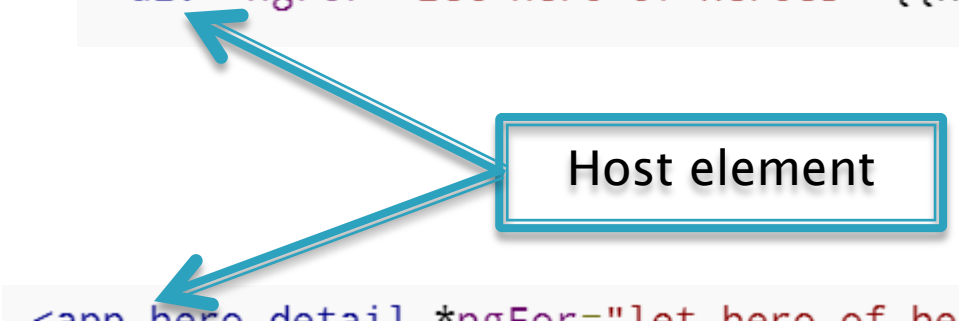

```
<p *ngIf="true">
  Expression is true and ngIf is true.
  This paragraph is in the DOM.
</p>
```

Directivas Estructurales

*ngFor

- Permite **crear una lista de elementos DOM, mediante la REPETICIÓN del elemento HOST sobre el que se aplica.**
- Se aplica sobre un elemento del DOM que se llama **elemento HOST**.

```
<div *ngFor="let hero of heroes">{{hero.name}}</div>
```




```
<app-hero-detail *ngFor="let hero of heroes"
[hero]="hero"></app-hero-detail>
```

- "*let hero of heroes*" → para cada **hero** en el array **heroes**, almacénalo en la variable **hero** y crea un elemento HOST en el DOM.

Directivas estructurales

- **Propiedades más usadas de *ngFor:**
  - **index** → devuelve el índice del elemento en cada iteración.

src/app/app.component.html

```
<div *ngFor="let hero of heroes; let i=index">{{i + 1}} -
{{hero.name}}</div>
```

  - **even**→ devuelve true si el número de la iteración es par.
  - **odd**→ devuelve true si el número de la iteración es impar.

```
1  <tr *ngFor="let hero of heroes; let even = even; let odd = odd"
2      [ngClass]="{ odd: odd, even: even }">
3      <td>{{hero.name}}</td>
4  </tr>
```

  - **first**→ devuelve true si es el elemento de la primera iteración.
  - **last** → devuelve true si es el elemento de la última iteración.

```
1  <tr *ngFor="let hero of heroes; let first = first; let last = last"
2      [ngClass]="{ first: first, last: last }">
3      <td>{{hero.name}}</td>
4  </tr>
```

Directivas Estructurales

ngSwitch

- ngSwitch no es "*ngSwitch" → IMPORTANTE: no lleva *.
- En realidad son varias directivas cooperando entre ellas:
  - ngSwitch → *se considera directiva de atributo, no estructural por eso se puede combinar con *ngFor.* ***Por eso no lleva *.***
  - *ngSwitchCase
  - *ngSwitchDefault

  } *directivas estructurales*
- Es como un switch de Java.
- Muestra un elemento de entre varios posibles en función de una condición o valor de una propiedad.
- Veremos **varios ejemplos de uso**:

Directivas Estructurales

```
<ul [ngSwitch]="person">
  <li *ngSwitchCase="'Mohan'">Hello Mohan</li>
  <li *ngSwitchCase="'Sohan'">Hello Sohan</li>
  <li *ngSwitchCase="'Vijay'">Hello Vijay</li>
  <li *ngSwitchDefault>Bye Bye</li>
</ul>
```

```
<div *ngFor="let id of ids">
  Id is {{id}}
  <div ngSwitch="{{id%2}}">
        <div *ngSwitchCase="'0'" [ngClass]="'one'">I am Even.</div>
        <div *ngSwitchCase="'1'" [ngClass]="'two'">I am Odd.</div>
        <div *ngSwitchDefault>Nothing Found.</div>
  </div>
</div>
```

Directivas Estructurales

```
<ul *ngFor="let person of people" [ngSwitch]="person.country">
  <li *ngSwitchCase="'IN'"
     class="text-success">
   {{ person.name }} ({{ person.country }})
  </li>
  <li *ngSwitchCase="'USA'"
     class="text-primary">
   {{ person.name }} ({{ person.country }})
  </li>
  <li *ngSwitchCase="'UK'"
     class="text-danger">
   {{ person.name }} ({{ person.country }})
  </li>
  <li *ngSwitchCase="'NP'"
     class="text-danger">
   {{ person.name }} ({{ person.country }})
  </li>
  <li *ngSwitchDefault
     class="text-warning">
   {{ person.name }} ({{ person.country }})
  </li>
</ul>`
```

NgSwitch

- Anil Singh (IN)
- Alok Singh (USA)
- Raju Cha (UK)
- Sushil Sin (NP)

Directivas de Atributo

ngClass

- Añade y elimina un conjunto de clases CSS.
- Si sólo queremos añadir una clase, usaremos el binding de atributo.

```
<div [class.special]="isSpecial">The class binding is
special</div>
```

- Si queremos añadir varias clases simultáneamente, usaremos esta directiva "ngClass".

```
<div [ngClass]=" ... ">
  ...
</div>
```

donde [ngClass]:

- debe recibir un objeto compuesto de key:value
  - cada key será el nombre de una clase CSS y,
  - el value será un booleano con valor true si queremos añadir esa clase y false en caso contrario.

Directivas de Atributo

- Ejemplo de uso:

```
<div [ngClass]="myClasses">
  ...
</div>
```

En el component, habría una propiedad “myClasses” como esta:

```
myClasses = {
  important: this.isImportant,
  inactive: !this.isActive,
  saved: this.isSaved,
  long: this.name.length > 6
}
```

El objeto enlazado a [ngClass] también puede ser devuelto por un método:

```
<div [ngClass]="setMyClasses()">
  ...
</div>
```

```
setMyClasses() {
  let classes = {
    important: this.isImportant,
    inactive: !this.isActive,
    saved: this.isSaved,
    long: this.name.length > 6
  };
  return classes;
}
```

Directivas de Atributo

- Otro ejemplo de uso:

src/app/app.component.ts

```
currentClasses: {};
setCurrentClasses() {
  // CSS classes: added/removed per current state of
component properties
  this.currentClasses =  {
    'saveable': this.canSave,
    'modified': !this.isUnchanged,
    'special':  this.isSpecial
  };
}
```

src/app/app.component.html

```
<div [ngClass]="currentClasses">This div is initially
saveable, unchanged, and special</div>
```

Tendríamos que llamar a setCurrentClasses() para que currentClasses tomase valor.

Directivas de Atributo

ngStyle

- Añade y elimina un conjunto de estilos HTML.
- Podemos seleccionar dinámicamente estilos inline basados en el estado de un componente.
- Si sólo queremos añadir un estilo, usaremos el binding de estilo.

```
<div [style.font-size]="isSpecial ? 'x-large' :
'smaller'" >
  This div is x-large or smaller.
</div>
```

```
<p [style.font-size.em]="'3'">
  A paragraph at 3em!
</p>
```

- Si queremos añadir varios estilos simultáneamente, usaremos esta directiva "ngStyle".

```
<p [ngStyle]="... ">
  You say tomato, I say tomato
</p>
```

donde [ngStyle]:

- debe recibir un objeto compuesto de key:value
  - cada key será el nombre de un estilo y,
  - el value será lo que sea apropiado para ese estilo.

Directivas de Atributo

- Ejemplo de uso:

```
<p [ngStyle]="myStyles">
  You say tomato, I say tomato
</p>
```

En el component, habría una propiedad “myStyles” como esta:

```
myStyles = {
'background-color': 'lime',
'font-size': '20px',
'font-weight': 'bold'
}
```

El objeto enlazado a [ngStyle] también puede ser dado INLINE:

```
<p [ngStyle]="{'background-color': 'lime',
    'font-size': '20px',
    'font-weight': 'bold'}">
  You say tomato, I say tomato
</p>
```

Directivas de Atributo

- Más ejemplos:

El objeto enlazado a [ngStyle] también puede ser devuelto por un método:

```
<p [ngStyle]="setMyStyles()">
  You say tomato, I say tomato
</p>
```

```
setMyStyles() {
  let styles = {
    'background-color': this.user.isExpired ? 'red' : 'transparent',
    'font-weight': this.isImportant ? 'bold' : 'normal'
  };
  return styles;
}
```

Directivas de Atributo

ngModel

- Se usa en formularios: *entrada datos del usuario*.
- Permite mostrar los datos de una propiedad en el DOM y actualizar desde el DOM esa propiedad.

```
<input [(ngModel)]="currentHero.name">
```

- Equivale a:

```
<input [value]="currentHero.name"
       (input)="currentHero.name=$event.target.value" >
```

¿Cómo usarla?

1. Se necesita importar FormsModule en app.module.ts
2. Añadirlo en los imports de NgModule en app.module.ts

Directivas de Atributo

src/app/app.module.ts (FormsModule import)

```
import { NgModule } from '@angular/core';
import { BrowserModule }  from '@angular/platform-
browser';
import { FormsModule } from '@angular/forms'; // <
JavaScript import from Angular

/* Other imports */

@NgModule({
  imports: [
    BrowserModule,
    FormsModule // <--- import into the NgModule
  ],
  /* Other module metadata */
```