DESARROLLO DE APLICACIONES CON ANGULAR

Angular

ANGULAR

- Directivas
  - Son los añadidos al HTML (lo amplían) propios de Angular.
    - De atributo:
      - NgStyle
      - NgModel
      - NgClass
    - Estructurales:
      - NgIf
      - NgFor
      - NgSwitch
        - NgSwitchCase
        - NgSwitchDefault
    - Otras directivas de @angular/common:
      - <ng-template>, <ng-container>, NgTemplateOutlet
      - NgComponentOutlet
      - NgPlural y NgPluralCase
      - NgForOf

ANGULAR

ngStyle

ANGULAR

- Directivas
  - De atributo: **[ngStyle]**.
    - Permite asignar estilo a un elemento.
    - Usando la sintaxis JavaScript de los atributos de estilo.
    - Directamente en el template:
      - <h1 class="titulo" **[ngStyle]**="{color:'orange'}">{{titulo}}</h1>
    - Como referencia a un atributo del componente:
      - <h1 class="titulo" [**ngStyle**]="**estilo**">{{titulo}}</h1>

```
this.estilo = {
        color:'blue',
        backgroundColor:'orange'
}
```

ANGULAR

- Directivas
  - De atributo: **[ngStyle]**.
    - Ejemplo: asignación de color dinámico utilizando ngModel:
    - En el ts:

```
this.estilo = {
    color:'blue',
    backgroundColor:'orange'
  }
```

    - En el template:

```
<h1 class="titulo" [ngStyle]="estilo">{{titulo}}</h1>
<input type="color" [(ngModel)]="estilo.color">
```

    - En app.module.ts:

```
import { FormsModule } from '@angular/forms';
imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
```

ANGULAR

ngModel

ANGULAR

- Directivas
  - De atributo: **[ngModel]**.
    - NOTA: Requiere haber importado el módulo FormsModule en app.module.ts.

```
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    GamesListComponent,
    GameComponentComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
```

ANGULAR

- Directivas
  - De atributo: **[ngModel]**.
    - Permite enlazar en los dos sentidos un campo de un formulario HTML con un atributo de una clase.
    - Válido para cualquier tipo de input o textareas.
    - En el componente, un atributo o atributo de instancia de objeto.
    - En el template HTML:
      - `<input type="number" [(ngModel)]="nombre"/>`
      - `<input type="color" [(ngModel)]="nombre"/>`
      - `<input type="checkbox" [(ngModel)]="nombre"/>`
      - `<textarea [(ngModel)]="nombre"></textarea>`

      - `<input type="radio" [(ngModel)]="nombre" value="OK" name="Disponible">`
      - `<input type="radio" [(ngModel)]="nombre" value="KO" name="Disponible">`

Nota: el atributo name en los input de type radio no tiene efecto si los ngModel hacen referencia al mismo atributo. Si no es así, el mismo valor de name hará que los radios sean excluyentes entre sí.

ANGULAR

ngClass

ANGULAR

- Directivas
  - De atributo: **[ngClass]**.
    - Permite asignar una clase o un conjunto de clases css a un elemento.
    - Pueden ser clases css simples, varias clases en un único string o arrays.
    - En el ts:

```
estilong:string[]=[];
this.estilong.push("redondo");
this.estilong.push("rojo");
```

    - En el ts (alternativa):

```
estilong:string="";
this.estilong="redondo rojo";
```

    - En el HTML:

```
<div [ngClass]="estilong">
    EstiloNg
</div>
```

    - En el CSS:
      - Declaración convencional de estilos css.

ANGULAR

- Directivas
  - De atributo: **[ngClass]**.
    - **Se puede condicionar la aplicación de las clases a valores booleanos:**
    - **En el TS:**

```
redondoactivo:boolean=true;
rojoactivo:boolean=true;
```

    - **En el HTML:**

```
<div [ngClass]="{redondo: redondoactivo, rojo: rojoactivo}">
    EstiloNg
</div>
```

    - **Ejemplo de uso: conmutación entre dos clases.**

```
<div [ngClass]="{redondo: redondoactivo, cuadrado: !redondoactivo}">
    EstiloNg
</div>
```

ANGULAR

- Directivas
  - De atributo: **[ngClass]**.
    - **Se puede condicionar la aplicación de las clases a expresiones:**
    - **En el TS:**

```
redondoactivo:boolean=true;
rojoactivo:boolean=true;
valor:number=10;
```

    - **En el HTML:**

```
<div [ngClass]="{redondo: redondoactivo, rojo: (valor>8)}">
    EstiloNg
</div>
```

ANGULAR

ngIf

ANGULAR

- Directivas
  - Estructurales: ***ngIf**.
    - **Permite condicionar la representación de un componente y su contenido.**

```
<div *ngIf="atributoTSlogico">
```

    - **La condición puede ser un atributo o una expresión.**

```
<h1 *ngIf="valor>11)">
```

ANGULAR

- Directivas
  - Estructurales: ***ngIf**.
    - Permite su uso junto con then-else. Se deben definir los bloques con ng-template.

```
<div *ngIf="(valor>10); then bloque1 else bloque2"></div>
<ng-template #bloque1>
    Bloque 1**************************
</ng-template>
<ng-template #bloque2>
    Bloque 2**************************
</ng-template>
```

ANGULAR

ngFor

ANGULAR

- Directivas
  - Estructurales: ***ngFor**.
    - Permite iterar por los elementos de una estructura.
    - El ámbito es el elemento HTML en el que se definen.

```
<div *ngFor="let item of coleccion">
<!-- AMBITO DEL FOR -->
</div>
```

ANGULAR

- Directivas
  - Estructurales: ***ngFor**.
    - El atributo trackBy permite indicar por qué valor se quiere determinar si un elemento del for ha cambiado para evitar que se modifique todo el DOM cuando se modifica un elemento de la lista, mejorando al eficiencia.

HTML:

```
<ul>
<li *ngFor="let dia of dias; index as i; trackBy: trackByDias">
    {{dia}}
</li>
```

TS:

```
dias=["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"]
trackByDias(index:number, dia:string):string{
    return dia;
}
```

ANGULAR

ngSwitch

ANGULAR

- Directivas
  - Atributo: **[ngSwitch]**.
  - Estructural: *ngSwitchCase *ngSwitchDefault
    - Permite mostrar contenido alternativo en función de una condición.
      - Requiere una variable de control declarada en el TS.
      - Si el valor es numérico → "1"
      - Si el valor es alfanumérico →"'uno'" (comillas simples dentro de comillas dobles)

```
<div [ngSwitch]="variable_control">
   <div *ngSwitchCase="valor1">Contenido 1</div>
   <div *ngSwitchCase="valor2">Contenido 2</div>
   <div *ngSwitchDefault>Contenido por defecto</div>
</div>
```

ANGULAR

<ng-template>, <ng-container> & *ngTemplateOutlet

ANGULAR

- Directivas
  - <ng-template #nombre_template>
    - Permite declarar una plantilla para su posterior uso (no se renderiza por defecto).
  - En el HTML:
    - `<ng-template #saludo><h3 style="color:red;">Hola</h3></ng-template>`
  - <ng-container> →Permite utilizar directivas estructurales sin añadir nuevos elementos al DOM.
  - *ngTemplateOutlet →Inserta una vista a partir de un template.
    - <ng-container *ngTemplateOutlet="'nombre_template'">

ANGULAR

ngComponentOutlet

ANGULAR

- Directivas
  - *ngComponentOutlet permite instanciar e insertar un componente de manera declarativa.
    - En el TS:
      - (Alternativa) Declarar un atributo de tipo any asignado a null si inicialmente no se quiere instanciar ningún componente.
        - `componenteDinamico:any=null;`
      - (Alternativa) Declarar un atributo asignando la clase del componente que se quiere instanciar.
        - `componenteDinamico=Componente1;`
    - En el template (HTML):
      - `<ng-container *ngComponentOutlet="componenteDinamico"></ng-container>`
    - La asignación del componente y/o cambio se realiza asignando al atributo el nombre de clase:
      - `componenteDinamico=Componente2;`

ANGULAR

ngPlural y ngPluralCase

ANGULAR

- Directivas
  - [ngPlural] permite mostrar un template de una lista a partir de un valor numérico.
    - En el TS:
      - `nombreplural=1 //El valor determina qué opción se va a mostrar`
    - En el template (HTML), se define el contenedor y las opciones:

```
<div [ngPlural]="nombreplural">
    <ng-template ngPluralCase="=0"><h1>Zulema</h1></ng-template>
    <ng-template ngPluralCase="=1"><ng-container
*ngTemplateOutlet="saludo"></ng-container></ng-template>
    <ng-template ngPluralCase="=2">Isabel</ng-template>
    <ng-template ngPluralCase="other">Otra persona</ng-template>
</div>
```

ANGULAR

nfForOf

ANGULAR

- Directivas
  - [ngForOf] permite crear una iteración. Se suele utilizar la forma abreviada *ngFor.

HTML:

```
<ul>
<li *ngFor="let dia of dias; index as i; trackBy: trackByDias">
    {{dia}}
</li>


</ul>
<ul>
    <ng-template ngFor let-item [ngForOf]="dias" let-i="index" [ngForTrackBy]="trackByDias">
        <li style="width: 200px;background-color: #CCC;">{{item}}</li>
    </ng-template>
</ul>
```

TS:

```
dias=["Lunes","Martes","Miércoles","Jueves","Viernes","Sábado","Domingo"]
trackByDias(index:number, dia:any):string{
    return dia;
}
```

ANGULAR

Directivas personalizadas

ANGULAR

- Directivas personalizadas
  - Permite generar directivas propias
  - ng generate directive *nombre_directiva*
  - Modifica **app.module.ts** importando la directiva y agregando la declaración.
  - Ejemplo:
    - ng generate directive directivas/mayúsculas

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

@Directive({
  selector: '[appMayusculas]'
})
export class MayusculasDirective {
  constructor() { }
}
```

Este selector indica que va a ser un atributo

  - Uso: <div appMayusculas>El texto…

ANGULAR

- Directivas personalizadas
  - Ejemplo:

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

@Directive({
  selector: '.appMayusculas'
})
export class MayusculasDirective {
  constructor() { }
}
```

Este selector indica que va a ser una clase

  - Uso: El texto…

ANGULAR

- Directivas personalizadas
  - Ejemplo:
  - Directiva:

```
import { Directive, ElementRef } from '@angular/core';

@Directive({
  selector: '[appMayusculas]'
})
export class MayusculasDirective {
  constructor(elemento:ElementRef) {
    elemento.nativeElement.style.textTransform="uppercase";
  }
}
```

  - Uso:

```
<p appMayusculas>Directiva:{{direccion}}</p>
```

ANGULAR

- Directivas personalizadas
  - Ejemplo utilizando renderer (abstrae del tipo de presentación):
  - Directiva:

```
import { Directive, ElementRef, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appMayusculas]'
})
export class MayusculasDirective {
  constructor(elemento:ElementRef, renderer:Renderer2) {
    renderer.setStyle(elemento.nativeElement, "text-transform", "lowercase");
    renderer.setStyle(elemento.nativeElement, "color", "#AA0000");
    renderer.setStyle(elemento.nativeElement, "width", "200px");
    renderer.setStyle(elemento.nativeElement, "border-style", "solid");
    renderer.setStyle(elemento.nativeElement, "border-width", "2px");

  }
}
```

  - Uso:

```
<p appMayusculas>Directiva:{{direccion}}</p>
```

ANGULAR

- Directivas personalizadas
  - Permiten incluir decoradores:

```
import { Directive, ElementRef, HostListener, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appMayusculas]'
})
export class MayusculasDirective {
  elemento:any;
  renderer:any;
  constructor(elemento:ElementRef, renderer:Renderer2) {
    this.elemento = elemento;
    this.renderer = renderer;
  }
  @HostListener('mouseover') entrando(){
    this.renderer.setStyle(this.elemento.nativeElement, "color", "blue");
  }
  @HostListener('mouseout') saliendo(){
    this.renderer.setStyle(this.elemento.nativeElement, "color", "green");
  }
}
```

ANGULAR

- Directivas personalizadas
  - Permiten incluir decoradores:

```
import { Directive, ElementRef, HostBinding, HostListener, Renderer2 } from '@angular/core';

@Directive({
  selector: '[appMayusculas]'
})
export class MayusculasDirective {
  elemento:any;
  renderer:any;
  @HostBinding('class') valor:any;
  constructor(elemento:ElementRef, renderer:Renderer2) {
    this.elemento = elemento;
    this.renderer = renderer;

  }
  @HostListener('mouseover') entrando(){
    this.renderer.setStyle(this.elemento.nativeElement, "color", "blue");
    this.valor="negrita";
  }
  @HostListener('mouseout') saliendo(){
    this.renderer.setStyle(this.elemento.nativeElement, "color", "green");
    this.valor="normal";
  }
}
```

ANGULAR

- Directivas personalizadas
  - Permiten incluir decoradores:
    - Decorador @Input
    - HTML:

```
(Alternativa) <p appMayusculas color="yellow">Directiva:{{direccion}}</p>
(Alternativa) <p appMayusculas [color]="elcolor">Directiva:{{direccion}}</p>
```

Literal

Atributo

    - TS:

```
@Input() color:string="";

this.renderer.setStyle(this.elemento.nativeElement, "color", this.color)
```