**A P U N T E S**

**COMANDOS ANGULAR**

**Para crear un nuevo proyecto ->** ng n nombre-del-proyecto

**Para crear un nuevo componente ->** ng g c nombre-del-componente

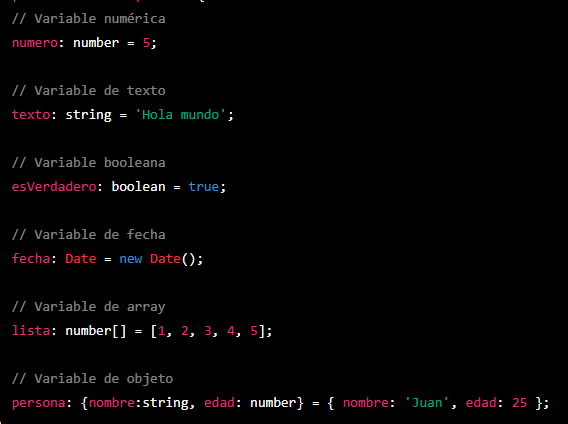
**Para crear una nueva directiva ->** ng g d nombre-de-la-directiva

**Para crear un nuevo servicio ->** ng g s nombre-del-servicio

**Para visualizar tu proyecto ->** ng serve -o

**Para recuperar los node-modules ->** npm i

**TIPOS DE DATOS EN ANGULAR (TypeScript)**



Es importante declarar el tipo de dato de una variable incluso cuando esta se declara e inicializa, ya que ayuda a prevenir errores de programación y mejora la legibilidad del código.

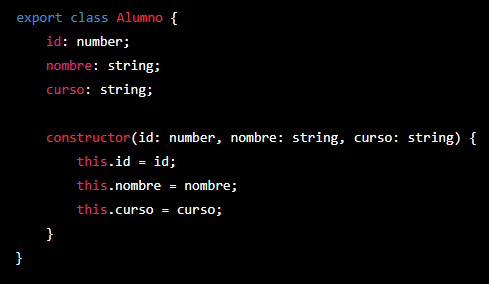
TypeScript puede realizar una comprobación de tipos en tiempo de compilación y avisar de cualquier posible error. Esto puede ayudar a detectar errores de programación antes de que ocurran en tiempo de ejecución y facilitar la depuración y mantenimiento del código.

**FICHEROS MODELOS**

Hablamos de **Ficheros Modelo** cuando almacenamos clases de objetos en un fichero .ts para más tarde utilizarlas en cualquier componente de la aplicación.

Normalmente, los ficheros modelos se crean en una carpeta llamada Models.

1. Creación de la clase en un fichero modelo (alumno.model.ts)



1. Importar la clase en el componente donde se desea usar e instanciar un objeto de dicha clase.





**DATA BINDING – INTERPOLATION**

**Interpolation** es una técnica en Angular que permite la inclusión de expresiones dentro de las etiquetas HTML. Permite mostrar información de manera dinámica en la vista. Se representa con doble llaves **{{ }}**



**DATA BINDING - PROPERTY BINDING**

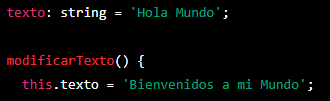
**Property Binding** nos permite modificar de manera dinámica el valor de los atributos de una etiqueta HTML, la cual debemos poner entre corchetes [ ], el dato que va entre comillas “ “, sería el nombre de la variable que se encuentra en el fichero TS





**DATA BINDING - EVENT BINDING**

**Event Binding** nos permite enlazar un evento de un elemento HTML con una expresión/método definido en el fichero TS.

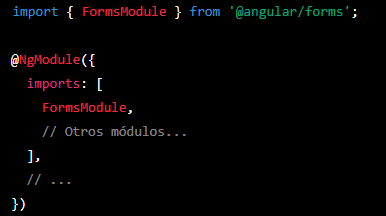
1. En el TS tenemos definida la variable texto y el método modificarTexto(), el cual estará asociado a un botón en el HTML, el método es llamado cuando se hace click en el botón
2. En el HTML tenemos un botón con un evento (click) asociado al método modificarTexto() y la variable texto interpolada. El valor inicial de la variable es “Hola Mundo”, esta cambiará a “Bienvenidos a mi Mundo” una vez se haga click en el botón.



**DIRECTIVA DE ATRIBUTO - [(ngModel)]**

Para realizar un **Two-Way Binding** o **Binding Bidireccional** usamos la directiva **[(ngModel)]**

Para hacer uso de esta directiva debemos importarla en el app.module.ts



**1.** En el fichero TS tenemos la variable miNombre declarada e inicializada con el valor “Victor”



**2.** En el fichero HTML tenemos dicha variable interpolada, y una etiqueta input con un **Two-Way Binding** a la variable "miNombre" usando **[(ngModel)]**.  
Cualquier cambio en el input será reflejado en el componente y viceversa, además se tiene un párrafo que muestra el valor de la variable "miNombre" para poder ver el cambio.

**DIRECTIVA DE ATRIBUTO – [ngClass]**

**[ngClass]** es una directiva de atributo de Angular que permite aplicar una o varias clases CSS a un elemento HTML en función de una expresión booleana, se evalua una condición y en base a esta aplica una clase CSS.

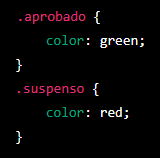
**1.** Declaramos e inicializamos la variable a evaluar en el fichero TS



**2.** En el fichero HTML



**3.** En el fichero CSS tenemos las clases .aprobado y .suspenso

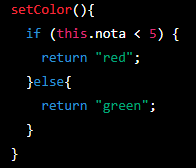


De esta forma estamos agregando una clase CSS a la etiqueta que contiene la directiva **[ngClass]** en base al valor de la variable “nota”.

**DIRECTIVA DE ATRIBUTO – [ngStyle]**

**[ngStyle]** nos permite modificar de manera dinámica los estilos de un elemento HTML en base a una expresión (normalmente un método).

**1.** Creamos el método/función y una variable en el fichero TS.

  
  
  
  
  
  
  
  
  
  
  
  
**2.** En el HTML del componente  
  
  
  
  
Con esto se está llamando al método **setColor()** y se está retornando el color en base a la condición de la nota.

**DIRECTIVA DE ESTRUCTURA - \*ngFor**

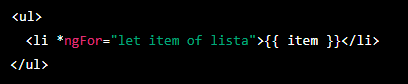
La directiva **\*ngFor** permite iterar sobre un array y/o objeto para poder mostrar sus atributos/valor/datos en etiquetas HTML.

**Ejemplo 1:**

La variable “lista” que es un Array de Strings, está declarada e inicializada en el fichero TS del componente.

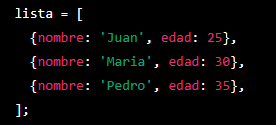


En este ejemplo se está utilizando \*ngFor para iterar sobre una variable(Array) llamada "lista" y mostrando cada elemento dentro de una etiqueta <li>.

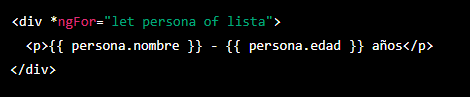


**Ejemplo 2:**

La variable “lista” que es un Array de Objetos, está declarada e inicializada en el fichero TS del componente.



En este ejemplo se está utilizando la directiva **\*ngFor** para iterar sobre el array "lista" y mostrar cada propiedad de cada objeto en una etiqueta <p> (nombre y la edad de cada persona).

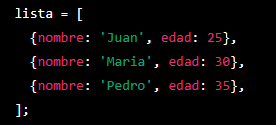


**DIRECTIVA DE ESTRUCTURA - \*ngIf**

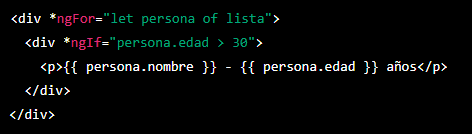
La directiva **\*ngIf** nos permite evaluar una condición en el fichero HTML, para hacer que un elemento se muestre o no.

**Ejemplo:**

La variable “lista” que es un Array de Objetos, esta declarada e inicializada en el fichero TS del componente.



En este ejemplo se está utilizando la directiva **\*ngFor** para iterar sobre el Array de Objetos "lista", y para cada objeto se está utilizando la directiva **\*ngIf** para comprobar si la edad es mayor a 30 años, si es así, se muestra el nombre y la edad de cada persona, si no se oculta.

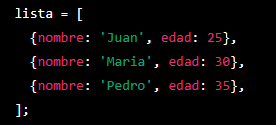


**DIRECTIVA DE ESTRUCTURA - \*ngSwitch**

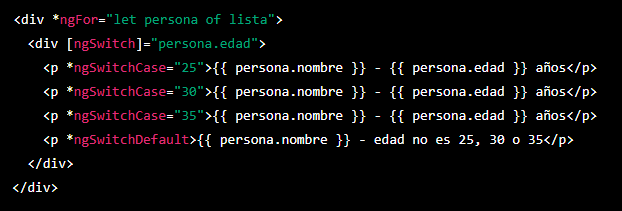
La directiva **\*ngSwitch** permite seleccionar un conjunto de elementos HTML para mostrar en función de un valor específico. Es similar al uso de un **switch** en programación, se especifica un valor para comparar y se seleccionan los elementos a mostrar dependiendo del valor. Se pueden especificar varios casos con **\*ngSwitchCase** y un caso default con **\*ngSwitchDefault**.  
  
El elemento HTML donde se coloque la directiva ngSwitch, será el contenedor donde se coloquen los elementos a mostrar dependiendo del valor especificado.

**Ejemplo:**

La variable “lista” que es un Array de Objetos, esta declarada e inicializada en el fichero TS del componente.



En este ejemplo se está utilizando la directiva **\*ngFor** para iterar sobre el array "lista" y para cada objeto se esta utilizando la directiva **\*ngSwitch** para comparar el valor de la edad con los casos 25, 30 y 35. Si la edad coincide con alguno de esos valores se muestra el nombre y la edad de cada persona, si no se usa el caso default para mostrar un mensaje.



**DECORADOR @Input – PADRE -> HIJO**

El decorador **@Input** en Angular se utiliza para pasar datos desde un componente padre a un componente hijo.

Por ejemplo, si un componente padre tiene una propiedad (Variable) llamada "mensajePadre" y desea pasar ese valor a un componente hijo, puede hacerlo de la siguiente manera:

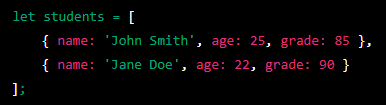
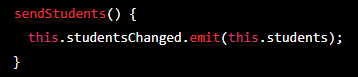
1. En el fichero HTML del componente padre “enviamos” el valor de la propiedad/variable incluyéndolo en el selector del componente hijo.  
    **[mensaje]** hace referencia al nuevo nombre del dato que estamos enviando.  
   **“mensajePadre”** es el nombre de la variable/propiedad que contiene el el dato a enviar.
2. En el fichero TS del componente hijo “recibimos” el dato.  
   Importamos **{ Input }**Creamos un **@Input()** **mensaje: string;**  
     
   El nombre de la variable asociada al decorador @Input ha de ser el mismo que usamos en el selector a la hora de enviar los datos   
    **Es importante hacer coincidir el tipo de dato que recibe con el que se envía.**

**DECORADOR @Output – HIJO -> PADRE**

El decorador **@Output** en Angular se utiliza para emitir eventos desde un componente hijo a un componente padre.

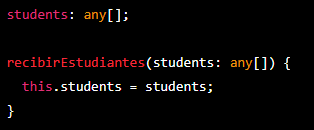
Por ejemplo, tenemos un Array de Objetos en la variable **“students”** declarada e inicializada en el fichero TS del componente hijo y queremos enviarlo al componente padre.

**EN EL COMPONENTE HIJO:**

1. Importar los módulos necesarios en el fichero TS.
2. Tener la variable declarada e inicializada en el fichero TS.  
   
3. Declarar una propiedad/variable decorada con **@Output** para emitir el evento.
4. Crear una función que emita el evento cuando sea necesario, en este caso cuando se quiera enviar el array **“students”** al componente padre.
5. En el HTML utilizar la función **sendStudents()** para emitir el evento cuando sea necesario. Por ejemplo, cuando se presione un botón.

**DECORADOR @Output – HIJO -> PADRE**

**EN EL COMPONENTE PADRE:**

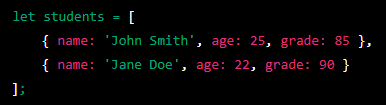
1. En el HTML, utilizamos la sintaxis **(nombreEvento)="handler"** para recibir el evento emitido por el componente hijo.  
     
     
     
   Estamos llamando al **@Output studentsChanged** el cual se encarga de emitir el evento, igualado a la función encargada de enviar el dato (array **“students”**) con **$event** como parámetro.
2. En el TS, declaramos una función para manejar el evento recibido y asignar el valor del array **“students”** a una variable.

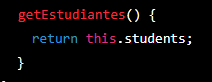
De esta forma, cuando se presione el botón en el componente hijo, se emitirá el evento **"studentsChanged"** y el componente padre recibirá el evento y asignará el valor del array **“students”** a la variable **"students".**

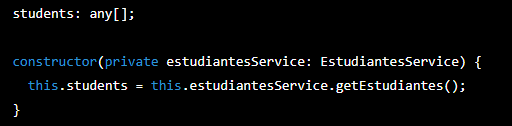
**SERVICIOS**

En Angular, un **servicio** es una clase que proporciona funcionalidades específicas que pueden ser reutilizadas a través de varios componentes.

Para crear un servicio en Angular, se debe utilizar el comando:  
**ng generate service nombre-del-servicio**

1. En la terminal, escribimos el comando **ng generate service servicios/alumnos** para crear un servicio llamado **"AlumnosService"** dentro de una carpeta llamada **"servicios"**. Este comando generará un archivo llamado **"alumnos.service.ts"** dentro de la carpeta "servicios" con la estructura básica del servicio.
2. En el archivo **"alumnos.service.ts"** declaramos una variable llamada **"students"** con el array de objetos de alumnos y una función que retorna dicho array.



1. Importamos el servicio en el TS del componente donde queremos utilizarlo.
2. Inyectamos el servicio en el constructor del componente e inicializamos una variable  
   donde vamos a guardar los datos.