## Trabajar con formularios

En el archivo *app.module.ts* importamos *FormsModule* de la librería *@angular/forms*



## Agregar Bootstrap

1. Instalamos vía npm
2. Incluimos el archivo CSS de Bootstrap dentro de angular.json (en versiones mas antiguas puede ser que se llame .angular-cli.json dentro de la sección *styles.*Por defecto npm instala en la carpeta *node\_modules* que se encuentra en el mismo nivel que la carpeta *src*



Si lanzamos nuestro proyecto e hicimos bien los pasos las tipográficas van haber cambiado a las por defecto de Bootstrap.

## Crear una clase desde Angular-CLI



Donde:

* models: es el nombre de la carpeta donde se va a crear la clase
* Empleado: es el nombre de la clase
* --skipTests (opcional): omite crear el archivo *specs* de test

### Poder usar la clase desde otro lugar (importarla)



Donde:

* { Empleado } : es el nombre de la clase
* ‘src/app/models/empleado’: es la ruta donde se encuentra el archivo .ts que contiene la clase.

## Crear método a una clase en TypeScript



## Tipos de datos en TypeScript

1. boolean
2. number (es valido para: decimales, hexadecimales, binarios y octales)
3. string
4. null
5. any (puede valer cualquier cosa)

any

array

tuple

enum

## Regiones en Visual Studio Code

*Doble barra (la de comentarios)* **+** *espacio* **+** *numeral* **+** *region* **+** nombreDeLaRegion  
*Doble barra (la de comentarios)* **+** *espacio* **+** *numeral* **+** *endregion*



## Arrays en TypeScript

Existen dos formas de declarar un array en TypeScript

1. Nombre del Array **+** : **+** Tipo del Array **+** []



1. Nombre del Array **+** : + *Array<tipo del Array>*



## Crear un componente con la CSS inline

ng generate component hero-app --inline-style

## Crear una interfaz



## Crear cualquier cosa dentro de un módulo en particular



Por ejemplo:



## Pasos para crear un servicio – Resumen

1. Crear el servicio mediante Angular-CLI



1. Agregamos la declaración del servicio en algún módulo, para lo cual debemos importar el archivo del donde se encuentra la clase del servicio (que creamos en el paso anterior) y luego agregar la clase en el array de *providers* (con esto ya lo podemos utilizar en cualquier componente del módulo donde lo estamos declarando)

**NO OLVIDARSE DE ESTE PUNTO**





1. Utilizamos nuestro servicio dentro de un componente: para utilizar el servicio lo que podemos hacer es inyectarlo como dependencia del componente (el componente debe pertenecer el módulo donde declaramos el servicio), también hay que importarlo dentro del componente donde lo necesitemos utilizar, luego a partir de ese momento podemos comenzar a utilizarlo dentro de la clase del componente.



## Pasos para utilizar *@Output* – Resumen

1. Dentro del archivo TS del componente hijo importamos la clase *EventEmitter* y el decorador *@Output*



1. Dentro del archivo TS del componente hijo creamos una propiedad del tipo *EventEmitter<tipo\_de\_datos\_que\_vamos\_a\_enviar>* y la vamos a decorar con @*Output()*



1. Dentro del archivo TS del componente hijo tenemos que definir los lugares desde donde se va emitir. Lo mas prolijo es definir un método para emitir y desde todos los lugares que se quiera emitir se llama a ese método:



## Pasos para utilizar ngFor - Resumen

1. Primero definimos un array en la clase del componente



1. Luego en la vista del componente agregamos lo siguiente:



## Agregar un color de fondo a toda la app

Editamos el archivo *src/styles.css* y pegamos lo siguiente:



Podemos encontrar otros diseños de colores en:  
<https://uigradients.com/#TalkingToMiceElf>

## Definir un texto en el html en función de una condición

