## 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:

