**CURSO DE ANGULAR**

**Curso realizado en:**

**Udemy:** <https://www.udemy.com/course/angular-fernando-herrera/>

Contenido

[Sección 3 – TYPESCRIPT 3](#_Toc66378092)

[Tipos de datos 4](#_Toc66378093)

[Arreglos 4](#_Toc66378094)

[Objeto 4](#_Toc66378095)

[Interfaz 4](#_Toc66378096)

[Funciones 4](#_Toc66378097)

[Funciones de flecha 4](#_Toc66378098)

[Funciones con objeto de parámetro 4](#_Toc66378099)

[Desestructuración 4](#_Toc66378100)

[Objeto 4](#_Toc66378101)

[Arreglo 4](#_Toc66378102)

[Argumentos 4](#_Toc66378103)

[Importación y exportación 4](#_Toc66378104)

[Clases 4](#_Toc66378105)

[Constructor 4](#_Toc66378106)

[Extender una clase 4](#_Toc66378107)

[Genéricos 4](#_Toc66378108)

[Decoradores 4](#_Toc66378109)

[Encadenamiento 4](#_Toc66378110)

[Sección 4 – ANGULAR 4](#_Toc66378111)

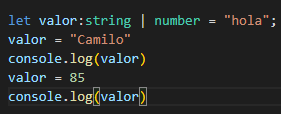
[Generar proyecto 4](#_Toc66378112)

[Explicación de cada archivo 4](#_Toc66378113)

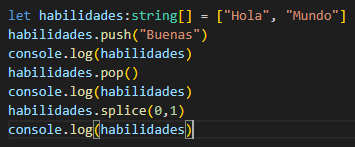
# Sección 3 – TYPESCRIPT

## Tipos de datos

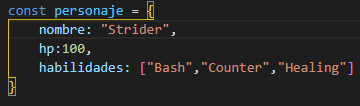
Una variable puede tener diferentes tipos de datos.



### Arreglos

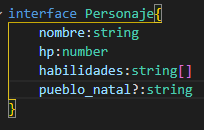


### Objeto



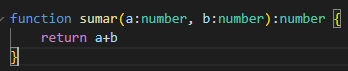
### Interfaz

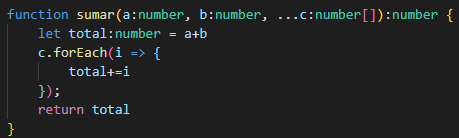
El ? se utiliza para decir que una variable puede ser opcional.



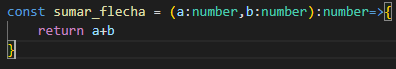
## Funciones

El orden de los parámetros es obligatorio, opcional y por último por defecto.

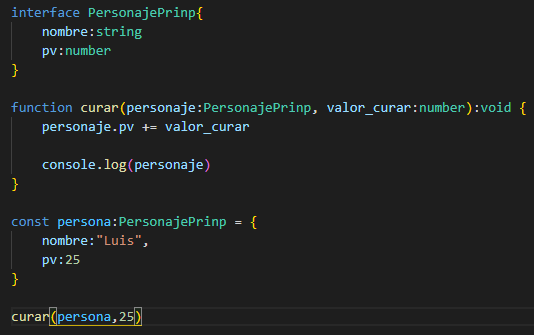




### Funciones de flecha



### Funciones con objeto de parámetro

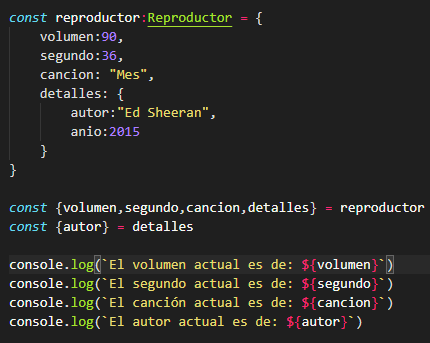


## Desestructuración

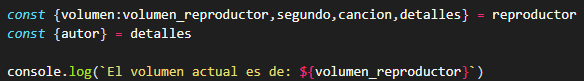
Extraer ciertas propiedades de un objeto/arreglo

### Objeto

Const {atributo1, atributo2…} = objeto “Esto quiere decir que se tomarán las propiedades del objeto que se especifiquen dentro de los { }”

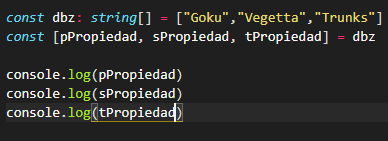


Si las variables chocan con otras variables por tener el mismo nombre entonces se puede cambiar el nombre de las variables desestructuradas. {variable:nueva\_variable}

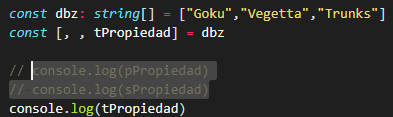


### Arreglo

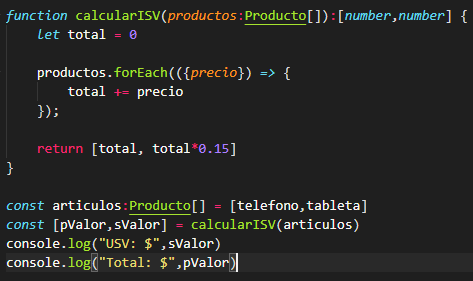
* Si necesito todos los valores



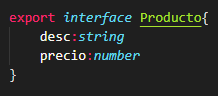
* Si necesito un valor en específico.



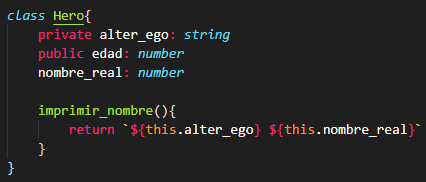
### Argumentos



## Importación y exportación

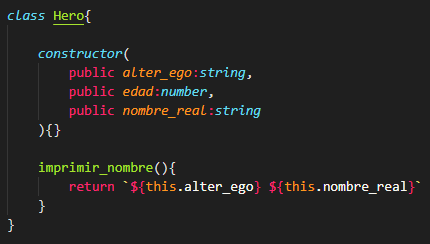


## Clases

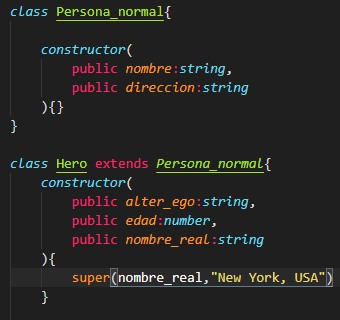


### Constructor

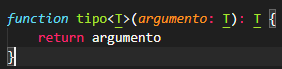
La verdadera forma de crear una clase



### Extender una clase

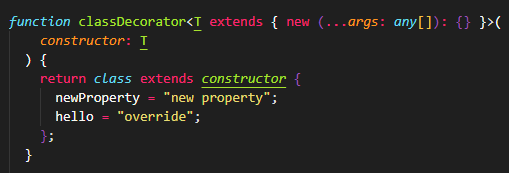


### Genéricos

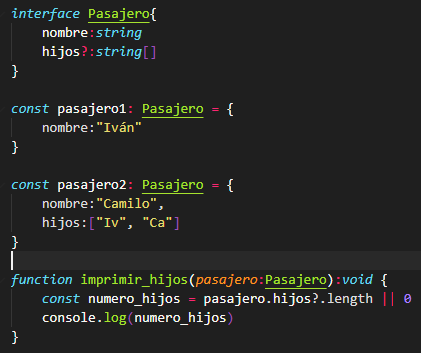


### Decoradores

Es una función que expande sus funcionales a clases, propiedades, etc.



## Encadenamiento



# Sección 4 – ANGULAR

Framework:

* Marco de trabajo estandarizado.
* Viene con todo lo que necesitas para trabajar.
* Es modular.
* Google es quien le da mantenimiento día a día.

Bloques fundamentales:



Componentes: Bloque de código que tienen fragmentos de html y typescript (usualmente una clase).

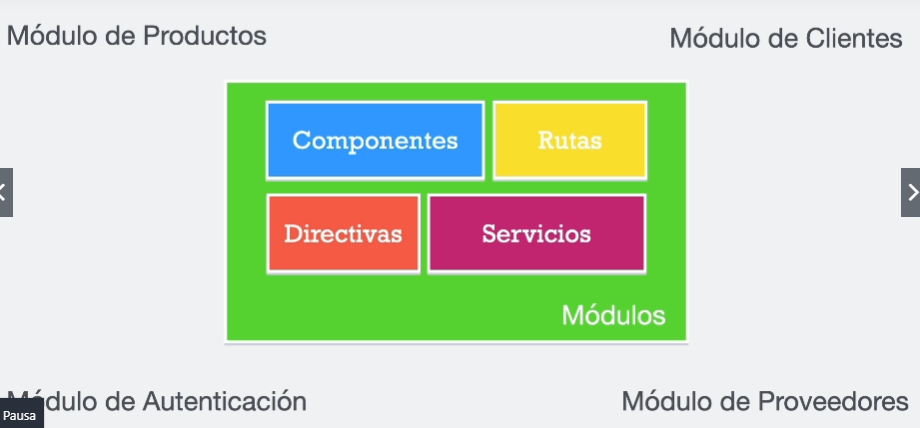
Servicios: Lugares centralizados de información.

Directivas:

* Directivas de componentes: Tiene un segmento de código html reutilizable.
* Directivas Estructurales: Modifican el html.
* Directivas de Atributos: Cambian la apariencia/comportamiento de un elemento.

Rutas: Mostrar diferentes componentes basados en el URL del navegador web.

Módulos: Permiten agrupar todo lo mencionado anteriormente.



## Generar proyecto

Estar ubicado donde queremos crear el proyecto de angular en el cmd y ejecutar

* ng new bases

Después ir a la carpeta donde se creó el proyecto (donde aparezca el package.json) y ejecutar el siguiente comando

* ng serve -o (Transcribe todo nuestro proyecto en TypeScript para arrancar el servidor y apenas esté listo nos lo muestra)

## Explicación de cada archivo

Los archivos .json son archivos de configuración

Tslint.json:

* Son reglas que obligan a programar de cierta manera. El ts es de TypeScript y quiere decir son las reglas para el ts que estamos trabajando.

Tsconfig.json:

* Archivo de configuración de TypeScript. Le dice a ts como quiere que se traduzca a JavaScript. O como funciona ts en el proyecto.

Tsconfig.spec.json:

* Toma todo lo que tiene el archivo tsconfig.json y le añade nuevas características.
* El spec está relacionado con las pruebas.

Tsconfig.app.json:

* Extiende de tsconfig.json y es enfocado en nuestra aplicación.

Package.json:

* Delicado y no se debería modificar manualmente.
* Contiene metadatos relevantes para el proyecto y se utiliza para gestionar las dependencias del proyecto, los scripts, la versión y mucho más.

Package.lock.json:

* Nos dice como se construyó los módulos de Node y como se construyó todo.
* Tampoco se debería modificar.

Karma.conf.json:

* Es la configuración de las pruebas unitarias e integración basadas en karma.

Angular.json:

* Contiene aplicaciones importantes de nuestra aplicación.

.gitignore:

* Son los archivos que quiero que git no le de un seguimiento.

.editoconfig:

* Reglas que vienen para algunos archivos. Rara vez se utiliza

.browserslistrc:

* Puede servir para incrementar la compatibilidad a internet Explorer 11

node\_modules:

* Contiene paquetes que nos ayudan a traducir de TS a JS, nos avisan de errores en una línea, ver errores en el html, ver errores desde la consola. Muchas cosas orientadas al desarrollo.

E2e:

* Ahí se harán las configuraciones de nuestras pruebas.

## Contenido de la carpeta src

App.component.css:

* Son los estilos que se aplican a nuestro html.

App.component.html:

* Es el archivo que el usuario puede ver.

App.component.spect.ts:

* Está relacionando con pruebas.

App.component.ts:

* Contiene una clase, pero con un decorador para convertir la clase en un componente propio de angular. Archivo ts de nuestra aplicación.

App.module.ts

* Contiene una clase común y corriente, pero con un decorador especial.

Assets:

* Se utiliza para colocar recursos estáticos (imágenes, videos, etc)

Environment:

* Se utilizan para colocar nuestras variables de entorno. Ya sean de desarrollo o de producción.

Main.ts:

* Casi nunca se toca, le dice a angular el ambiente en el que está corriendo.

Polyfills.ts:

* Mayor compatibilidad con otros navegadores. Casi nunca se toca.

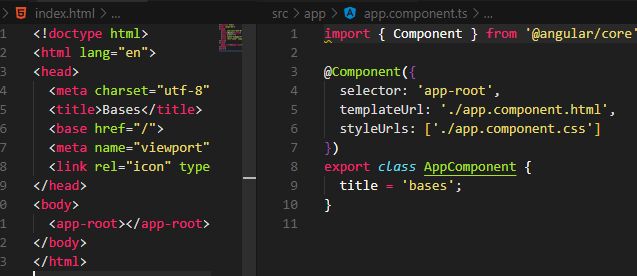
Style.css:

* Hoja de estilos general que se aplica a toda la aplicación.

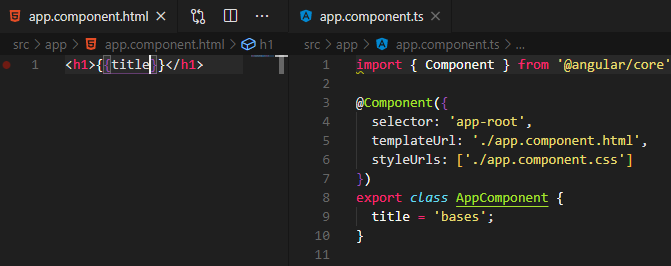
Test.ts:

* La configuración del entorno de pruebas.

## App Component

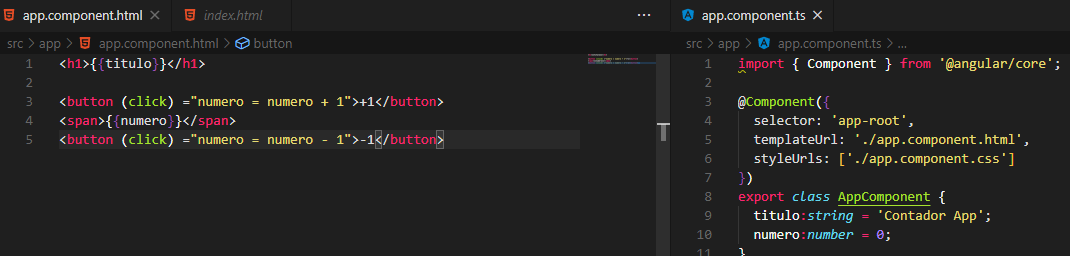


* En el app.component.ts los parámetros que utiliza el decorador son necesarios para mostrar nuestra app. El selector es el nombre de nuestra app y la utiliza el index.html general. Y el template url es la dirección de nuestro app.component.html para mostrar al usuario. También existe el parámetro Template que permite agregar ahí el código html y mostrarlo directamente en la página.



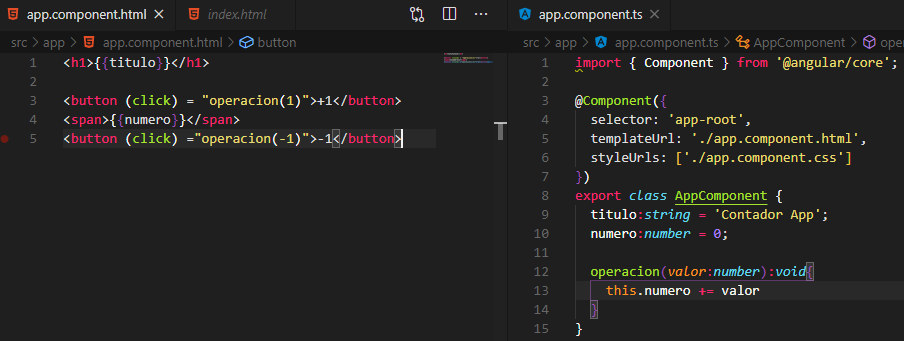
* Podemos utilizar las propiedades de nuestro componente para mostrarlo en el html mediante la expresión {{ }}, podemos imprimir cualquier expresión que coloquemos ahí, por ejemplo: {{ 1 + 1 }}

## Contador App



* No es recomendable colocar lógica en el template, pero en este caso es pasable porque es una instrucción sencilla.

## Métodos en el componente



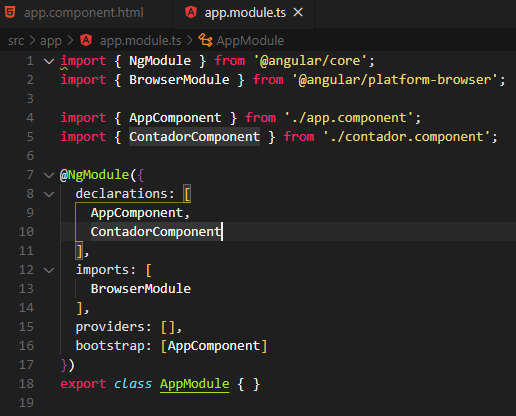
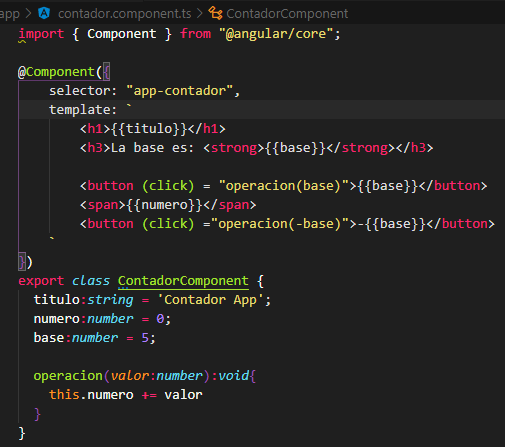
## Crear un componente manualmente.

Creación del nombre del archivo:

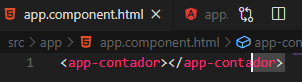
* Nombre-archivo.tipo\_bloque.ts ejemplo: contador-original.component.ts

Se agrega la clase y su respectivo decorador para convertir la clase en un componente propio de Angular.

Y se agrega el nombre de la clase a app.module.ts a las declaraciones de los componentes que se están utilizando

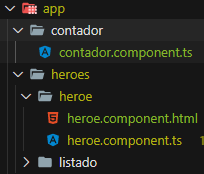


Y se agrega el nombre (selector del @Component) de nuestro componente en el app.component.html para mostrarlo al usuario.



## Componente de Héroe y separación de directorios

No hay que trabajar directamente en el app component, hay que colocar los componentes en un nuevo directorio.



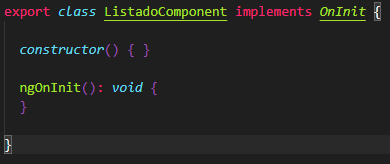
## Crear componente automáticamente

Para crear un componente automáticamente se utiliza el comando:

* Ng g c ruta/nombre\_archivo
* Donde g significa generate y c component

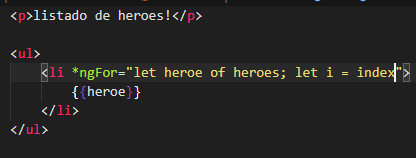
El OnInit es un ciclo de vida y que con su constructor son ciertos pasos/métodos que dispara angular de manera automática.

El ngOnInit se utiliza para inicializar cosas como por ejemplo traer la información al hacer la petición de un servicio.



## Directiva \*ngFor

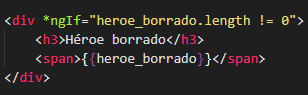
Funciona igual que un for





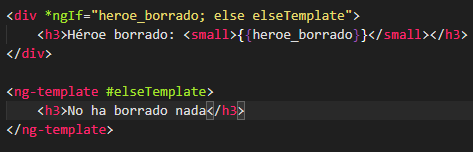
## Directiva -nglf

Muestra o no el bloque html si se cumple la condición



## Directiva ng-Template y ngIf-else

# es una referencia local



## Módulos

Agrupar componentes y piensas de la aplicación que tienen sentido entre sí.

* Declaraciones: que componentes, archivos, etc. Contiene la aplicación.
* Export: Que cosas quiero que sean visibles afuera de este módulo (públicas).
* Import: solamente van módulos. CommunModule ofrece dieferentes cosas, entre ellas las directivas

