[Fecha]

Frank

[Nombre de la empresa]

[Dirección de la compañía]

[Título del documento]

[Subtítulo del documento]

Contenido

[1. COMPILAR CODIGO TYPESCRIPT. 2](#_Toc62150253)

[1.1 COMPILACION AUTO. 2](#_Toc62150254)

[2. VARIABLES Y TIPOS DE DATOS. 2](#_Toc62150255)

[2.1 VARIABLES PERSONALIZADAS. 3](#_Toc62150256)

[3. FUNCIONES CON TIPADO FUERTE. 3](#_Toc62150257)

[4. POO. 4](#_Toc62150258)

[4.1 CLASES. 4](#_Toc62150259)

[4.2 CONSTRUCTOR. 5](#_Toc62150260)

[4.3 IMPORTS/EXPORTS. 5](#_Toc62150261)

[4.3.1 EXPORT. 5](#_Toc62150262)

[4.4 INTERFACES. 6](#_Toc62150263)

[4.5 HERENCIA. 7](#_Toc62150264)

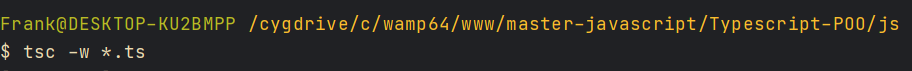
[4.6 DECORADORES. 8](#_Toc62150265)

# 1. COMPILAR CODIGO TYPESCRIPT.

Ejecutamos el comando



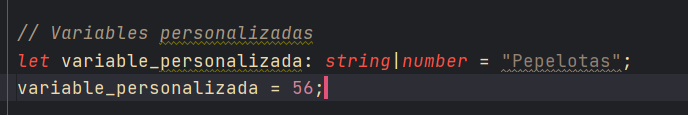
## COMPILACION AUTO.



# 2. VARIABLES Y TIPOS DE DATOS.



## 2.1 VARIABLES PERSONALIZADAS.



# 3. FUNCIONES CON TIPADO FUERTE.

// Named function

*function* add1(*x*, *y*) {

    return x + y;

}

// Funcion con tipado

*function* add(*x*: *number*, *y*: *number*): *number* {

    return x + y;

}

// Anonymous function

*let* myAdd1 = *function* (*x*, *y*) {

    return x + y;

};

// Funcion con tipado

*let* myAdd = *function* (*x*: *number*, *y*: *number*): *number* {

    return x + y;

};

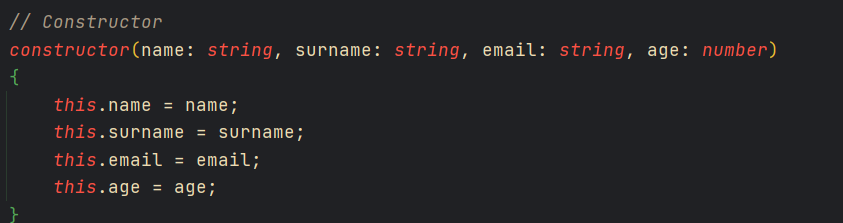
console.log(add1(2,3));

# 4. POO.

## 4.1 CLASES.



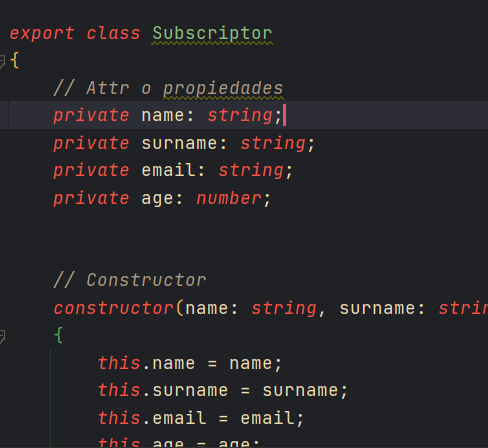
## 4.2 CONSTRUCTOR.



## 4.3 IMPORTS/EXPORTS.

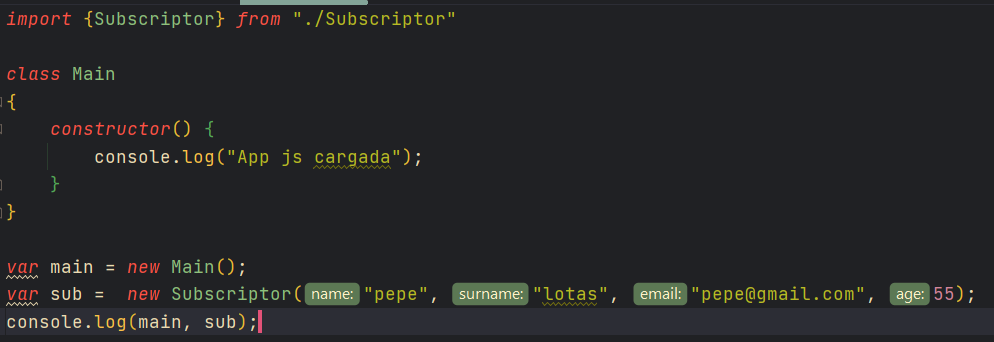
### 4.3.1 EXPORT.

Imaginemos que necesitamos usar una clase para instanciar un objeto en otro fichero por ejemplo en main.js para ello hemos de exportarla e importarla en el fichero deseado.



Ahora en el fichero main.js realizamos el import



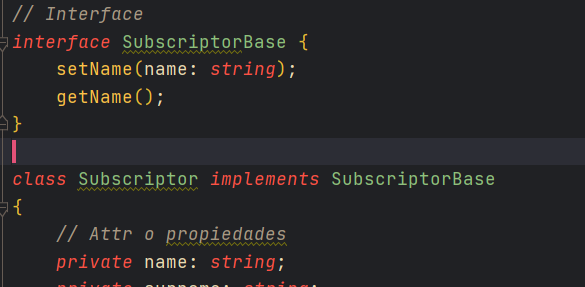


Cuando trabajemos con frameworks este tema lo controla dicho frameworks.

## 4.4 INTERFACES.

Es un contrato en el cual definimos que propiedades y métodos **obligatorios** va a utilizar dicha clase, al heredar otra de esta interface obligatoriamente tendremos que redifinir dichas propiedades y atributos.

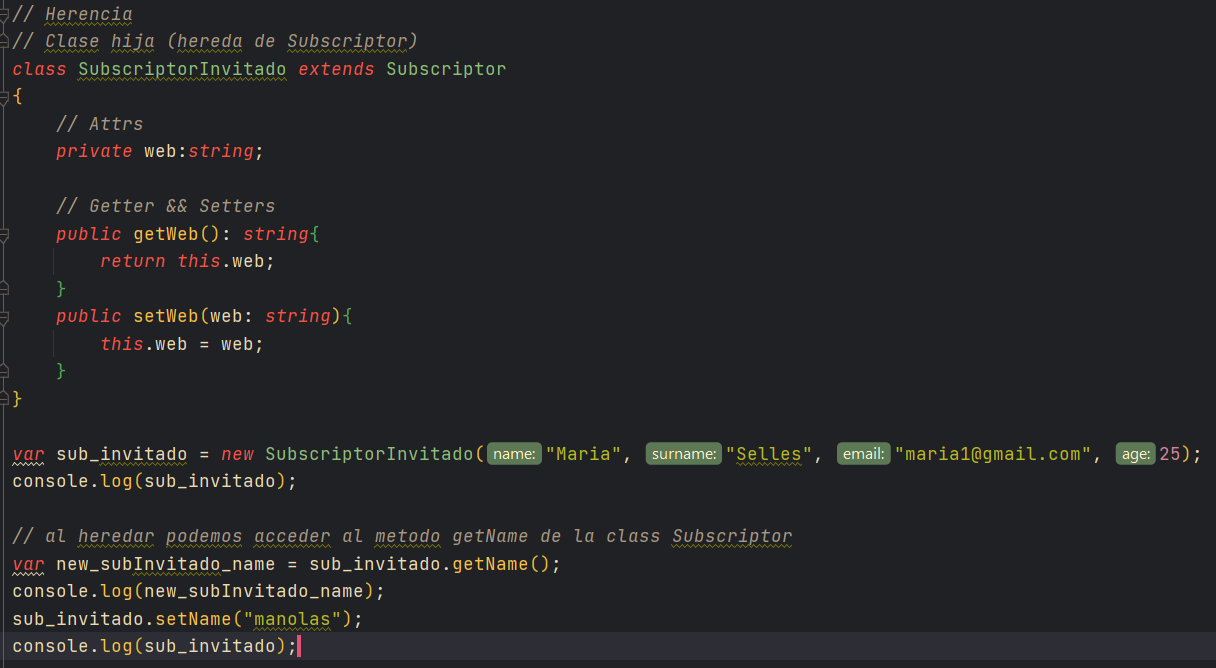
Usaremos la palabra reservada **implements**



## 4.5 HERENCIA.

Mecanismo en la cual una clase hija puede heredar las características de la clase padre y utilizar sus métodos y propiedades.

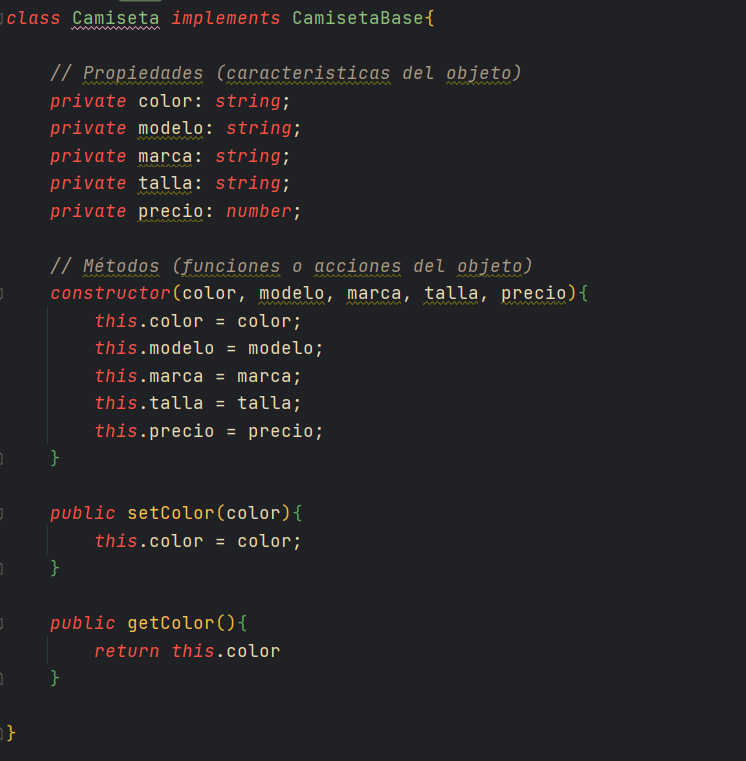
Para heredar usamos la palabra reservada **extends**



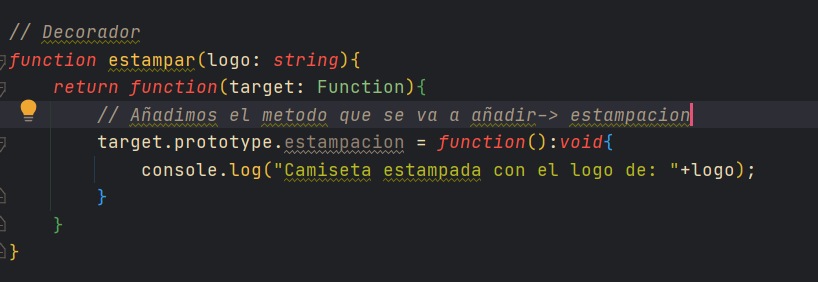
## 4.6 DECORADORES.

Es un patrón de diseño en el cual definimos unos determinados parámetros o metadatos, a una clase, y en función de dichos parámetros realice una acción u otra.

Imaginemos un ejemplo donde tenemos la clase camiseta :

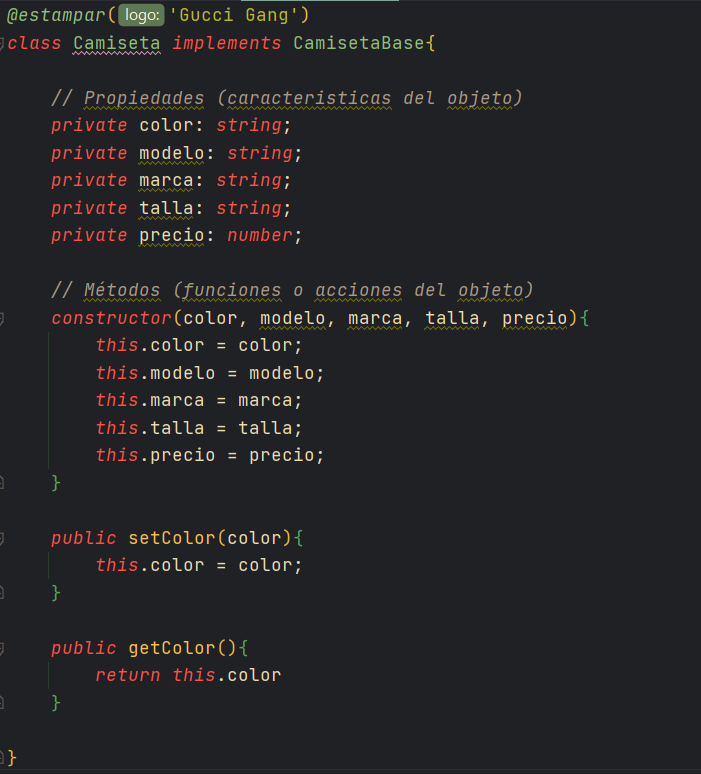


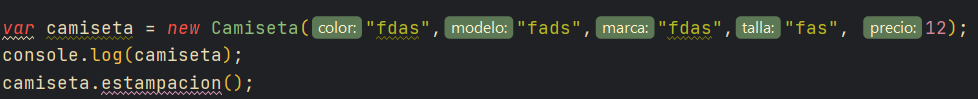
Vamos a definir un decorador y al aplicar este a una clase se va a añadir el método que hemos implementado en el decorador. En este caso vamos a definir el decorador **estampar**



Ahora mediante @estampar

NOTA :NUNCA SE CIERRA CON ;





# 5. AÑADIR JQUERY A TYPESCRIPT.

https://stackoverflow.com/questions/32050645/how-to-use-jquery-with-typescript

Creamos un fichero packages.json

