## @Output – Comunicación componente hijo-padre

Mediante *@Output* vamos a poder comunicar datos desde los componentes hijos hacia los padres (lo contrario de *@Input*)

La idea es crear eventos personalizados que nos van a permitir avisar a los componentes padres de alguna situación que ocurra en los componentes hijos

El uso de @*Output* no es trivial e implica el trabajo de varios actores:

## En el componente Hijo

### Clase EventEmitter

Es una clase de Angular que implementa objetos capaces de emitir un evento.  
Para utilizarla debemos importarla en el componente hijo, se encuentra en la librería *@angular/core*Lo hacemos de la siguiente manera:



Luego vamos a declarar una propiedad del tipo *EventEmitter* dentro del componente. Las clases *EventEmitter* espera que le especifiquemos un tipo de datos mediante genéricos de TypeScript.

Este tipo de datos va a ser el que necesitemos comunicarle al padre. La vamos a declarar de la siguiente manera:



En el caso de nuestro ejemplo estamos especificando que nuestro emisor de eventos va a ser del tipo *string,* pero podría haber sido alguna clase nuestra o interfaz

### Decorador @Output

Mediante el decorador *@Output* le indicamos al framework que la propiedad que acabamos de declarar la vamos a comunicar el hijo hacia el padre (de igual manera que hicimos con *@Input*)

Para poder utilizarlo primero debemos importarlo, al igual que *EventEmitter*, *Output* se encuentra en *@angular/core*



Una vez importado vamos a agregarlo delante de la propiedad que queremos decorar @Output()



### Método emit()

El método *emit()* pertenece al objeto *EventEmitter* y lo vamos a llamar cada que necesitemos enviar datos hacia el padre.

El método recibe como parámetro el dato que queremos enviar. Obviamente el tipo de datos tiene que coincidir con el que utilizamos como genérico para declarar la propiedad del tipo *EventEmitter.*



En nuestro ejemplo creamos el método *OnPropagar* para emitir

## Recibir eventos personalizados en el componente padre

Por último en la template del componente padre cuando agreguemos el componente hijo vamos a escribir entre paréntesis el nombre de la propiedad del tipo *EventEmitter* en el componente hijo (como si se tratase de un evento del componente hijo, como por ejemplo hacemos con *(click)* con un botón) y lo vamos a igualar a un método en el componente padre que va ser el que se va a llamar cuando desde el componente hijo se propague el evento.

Como parámetro del método vamos a colocar *$event* que son los datos que nos envió el hijo

Template del componente padre:



TS del componente padre:



## Resumen de pasos para utilizar *@Output*

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:

