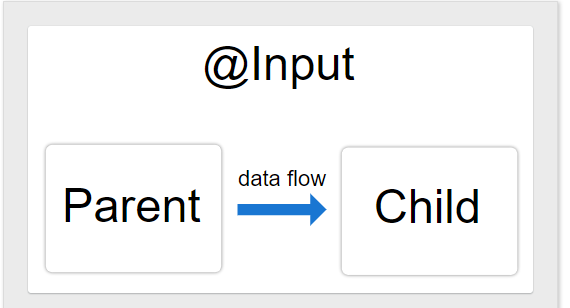
# Interacción de componentes

## Este libro contiene escenarios de comunicación de componentes comunes en los que dos o más componentes comparten información.



## Pasar datos de padre a hijo con enlace de entrada

HeroChildComponent tiene dos propiedades de entrada , típicamente adornadas con [el decorador @Input()](https://angular.io/guide/inputs-outputs#input) .



El segundo crea un alias para el nombre de la propiedad del componente secundario como .

@[Input](https://angular.io/api/core/Input) masterName'master'

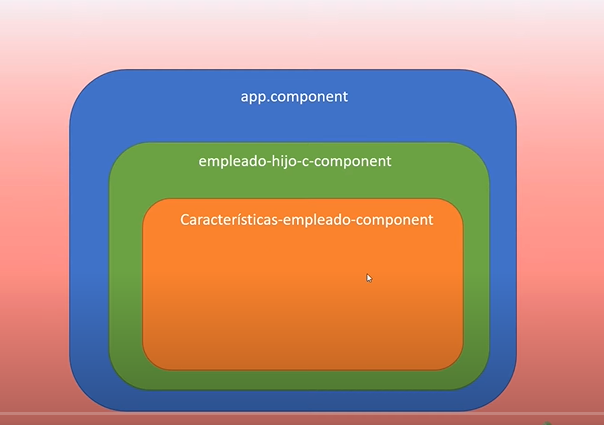
HeroParentComponentAnida al hijo dentro HeroChildComponentde un repetidor, vinculando su propiedad de cadena al alias del hijo , y la instancia de cada iteración a la propiedad del hijo.

\*[ngFor](https://angular.io/api/common/NgForOf)mastermasterherohero

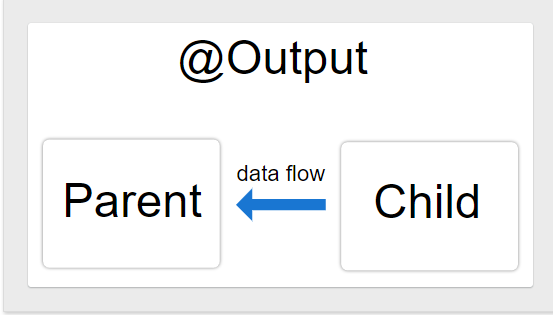
|  |  |
| --- | --- |
|  |  |

# Envío de datos a un componente principal

***(Nota: En mi ejemplo creamos otro componente mas , llamado empleado-nieto lo cual este será el hijo del componente empleado-hijo ) ver código ¡!!***

******

El decorador en un componente o directiva hijo permite que los datos fluyan del hijo al padre.@[Output](https://angular.io/api/core/Output)()



@[Output](https://angular.io/api/core/Output)() marca una propiedad en un componente secundario como una puerta a través de la cual los datos pueden viajar del componente secundario al principal.

El componente secundario usa la propiedad para generar un evento para notificar al elemento principal sobre el cambio. Para generar un evento, debe tener el tipo de , que es una clase que se utiliza para emitir eventos personalizados.@[Output](https://angular.io/api/core/Output)()@[Output](https://angular.io/api/core/Output)()[EventEmitter](https://angular.io/api/core/EventEmitter)@angular/core

El siguiente ejemplo muestra cómo configurar un componente secundario que envía datos de un HTML a una matriz en el componente principal.@[Output](https://angular.io/api/core/Output)()<input>

Para usar , debe configurar el padre y el hijo.@[Output](https://angular.io/api/core/Output)()

1. Importar [Output](https://angular.io/api/core/Output)y [EventEmitter](https://angular.io/api/core/EventEmitter)en la clase de componente secundario:



1. En la clase de componentes, decora una propiedad con . El siguiente ejemplo tiene un tipo de , lo que significa que es un evento.@[Output](https://angular.io/api/core/Output)()newItemEvent @[Output](https://angular.io/api/core/Output)()[EventEmitter](https://angular.io/api/core/EventEmitter)

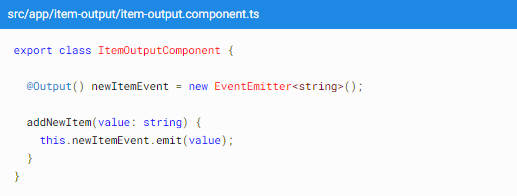


Las distintas partes de la declaración anterior son las siguientes:

* @[Output](https://angular.io/api/core/Output)()—una función de decorador que marca la propiedad como una forma de que los datos pasen del elemento secundario al elemento principal
* newItemEvent-El nombre de@[Output](https://angular.io/api/core/Output)()
* [EventEmitter](https://angular.io/api/core/EventEmitter)<string>—el tipo de@[Output](https://angular.io/api/core/Output)()
* new [EventEmitter](https://angular.io/api/core/EventEmitter)<string>(): le dice a Angular que cree un nuevo emisor de eventos y que los datos que emite son de tipo cadena.

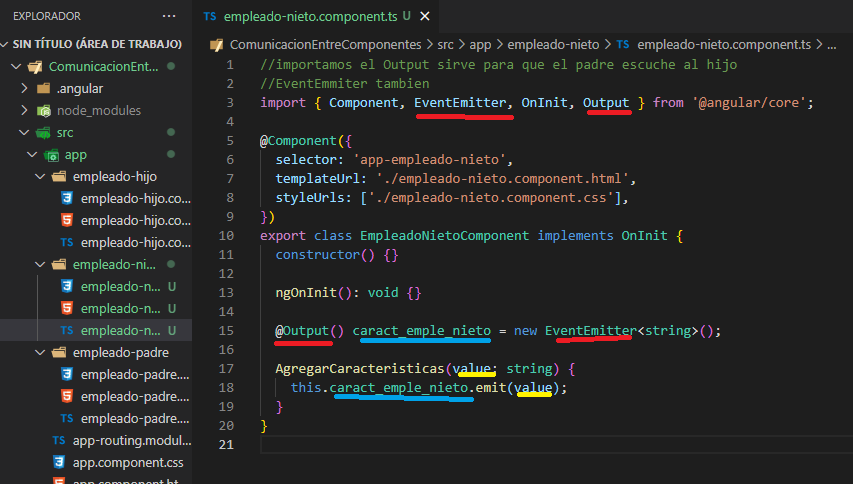
Para obtener más información sobre [EventEmitter](https://angular.io/api/core/EventEmitter), consulte la documentación de la [API de EventEmitter](https://angular.io/api/core/EventEmitter) .

1. Cree un addNewItem()método en la misma clase de componente:



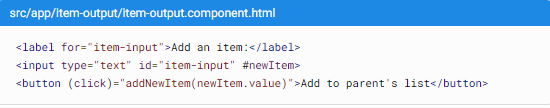
La addNewItem()función usa el , para generar un evento con el valor que el usuario ingresa en el .@[Output](https://angular.io/api/core/Output)()newItemEvent<input>

Ejemplo de mi código



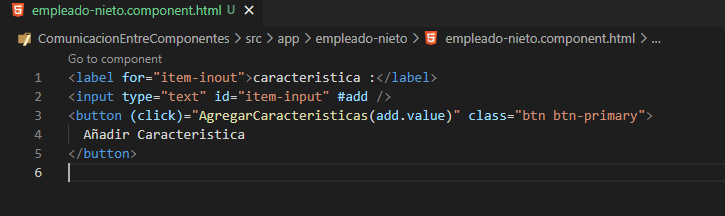
## Configurando la plantilla del niño

La plantilla del niño tiene dos controles. El primero es un HTML <input>con una variable de [referencia de plantilla](https://angular.io/guide/template-reference-variables)#newItem , donde el usuario escribe el nombre de un elemento. La valuepropiedad de la #newItemvariable almacena lo que el usuario escribe en el archivo <input>.



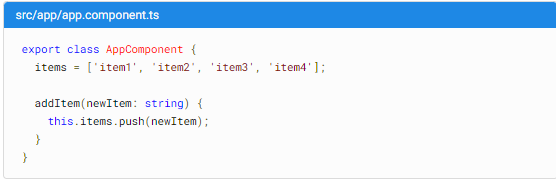
El segundo elemento es un <button>con un click [enlace de evento](https://angular.io/guide/event-binding) .

El (click)evento está vinculado al addNewItem()método en la clase del componente secundario. El addNewItem()método toma como argumento el valor de la #newItem.valuepropiedad.

Ejemplo de mi código

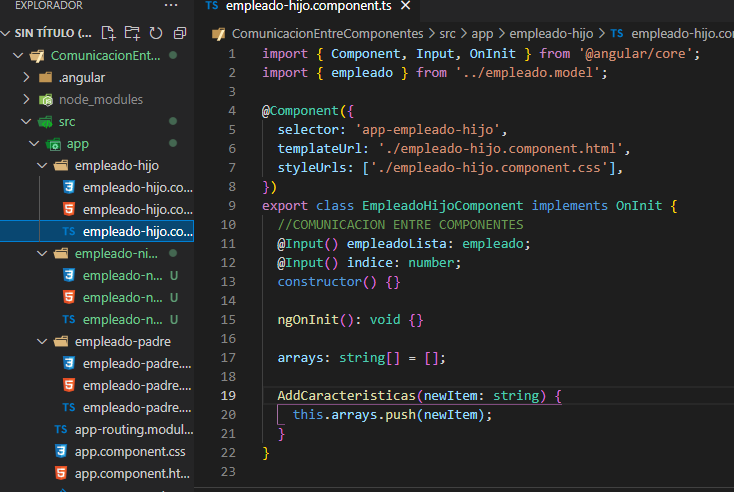
## Configuración del componente principal

En AppComponenteste ejemplo, se presenta una lista de elementos ítems en una matriz y un método para agregar más elementos a la matriz.



El addItem()método toma un argumento en forma de cadena y luego agrega esa cadena a la ítems matriz.

Ejemplo de mi código



## Configuración de la plantilla de los padres

En la plantilla del padre, vincule el método del padre al evento del hijo.

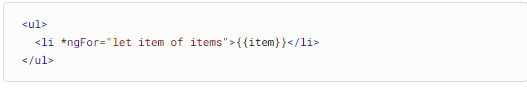
Coloque el selector secundario, aquí <app-item-output>, dentro de la plantilla del componente principal, app.component.html.



El enlace de eventos (newItemEvent)='addItem($event)', conecta el evento en el elemento secundario, newItemEventcon el método en el elemento principal, addItem().

contiene los $eventdatos que el usuario escribe en la <input>interfaz de usuario de la plantilla secundaria.

Para ver el funcionamiento, agregue lo siguiente a la plantilla de los padres:@[Output](https://angular.io/api/core/Output)()



Itera sobre los elementos de la matriz . Cuando ingresa un valor en el elemento secundario y hace clic en el botón, el elemento secundario emite el evento y el método del elemento principal empuja el valor a la matriz y el nuevo elemento aparece en la lista.\*[ngFor](https://angular.io/api/common/NgForOf) items<input>addItem()items

Ejemplo de mi código

