Curso de Desarrollo Web con Angular

Artículo

Entendiendo Input y Output a fondo

Jorge Cano 23 PlatziRank Mar. 7, 2017

Entendiendo Input y Output a fondo

Al tener componentes dentro de otros, necesitamos que tengan una comunicación mas fluida que las comunicaciones que podríamos llegar a conseguir a travez de un servicio.

Para eso tenemos los inputs y outputs, los nombres ya por si solos nos dan a entender que tenemos forma de ingresar y egresar datos.

Haciendo un poco de seguimiento sobre el curso, un componente se genera a travez de un tag:

```
<my-component></my-component>
```

De esta forma estaríamos llamando a un componente que su selector seria my-component

```
@Component({
    selector: 'my-component',
    ...
})
```

Ahora que pasa con ese componente cuando tiene otro componente adentro

```
@Component({
    selector: 'my-component',
    template: '<my-component-hijo></my-component-hijo>'
})
```

Ahora tenemos un componente dentro de otro, pero lo que pasa ahora, es que cuando se genera my-component tambien se genera my-component-hijo, pero no tenemos ningun pasaje de parametros ni nada, para ello vamos a utilizar @Input.

Input es un annotation que nos deja enviar datos del componente padre para poder recibirlos con el hijo.

Entonces lo que vamos a hacer es crear una variable string, con el annotation input para poder recibir el parametro enviado desde el tag.

my.component.hijo.ts

```
import { Component, Input } from '@angular/core';
@Component({
    selector: 'my-component-hijo',
    template: ' {{ variableRecibida }} '
})

export class MyComponentHijo{

    @Input() variableRecibida: string;

    constructor(){

    }

}
```

De esta forma vamos a tener nuestra variableRecibida, que vamos a obtenerla del componente padre... para eso vamos a modificar el componente padre:

my.component.ts

```
import { Component } from '@angular/core';
import { InputComponent } from './input/input.component';

@component({
    selector: 'my-component',
    template: ' <my-component-hijo variableRecibida="{{variableAenviar}}"></my-component-
hijo> '
})

export class MyComponent{

    variableAenviar: string;

    constructor (){}

    ngOnInit(){
        this.variableAenviar = 'Soy un string, enviado al componente hijo';
    }
```

```
}
```

Y de esta forma vamos a enviarle la variable al input… para revisar un poco mas el codigo… cuando escribo “variableRecibida” en el template dentro del tag del componente hijo lo que va a pasar es “bindear” esa variable con la variable string que recibe el componente hijo.

Ahora que ya podemos recibir parametros dentro de nuestro componente hijo, nos faltaria que le envie variables al componente padre, para ello vamos a tener que realizar una serie de pasos para poder realizarlo

El primero de todo es crear una variable dentro de nuestro componente padre, para poder recibir el valor del componente hijo

my.component.ts

```
import { Component } from '@angular/core';
import { InputComponent } from './input/input.component';

@component({
    selector: 'my-component',
    template: ' <my-component-hijo variableRecibida="{{variableAenviar}}"></my-component-
hijo> '
})

export class MyComponent{

    variableAenviar: string;

    variableRecibidaDelHijo: string;

    constructor (){}

    ngOnInit(){
        this.variableAenviar = 'Soy un string, enviado al componente hijo';
    }



}
```

Debido a que Output es un “eventEmitter” vamos a tener que crear una funcion para poder recibir el parametro

my.component.ts

```
import { Component } from '@angular/core';
import { InputComponent } from './input/input.component';

@component({
    selector: 'my-component',
    template: ' <my-component-hijo variableRecibida="{{variableAenviar}}"></my-component-
hijo> '
})

export class MyComponent{

    variableAenviar: string;

    variableRecibidaDelHijo: string;

    constructor (){}

    ngOnInit(){
        this.variableAenviar = 'Soy un string, enviado al componente hijo';
    }

    funcionParaRecibirVariable(_variableRecibida:string){
        this.variableRecibidaDelHijo = _variableRecibida;
    }


}
```

Ahora vamos a agregar a nuestro HTML del componente padre, la función que creamos para poder recibir el evento del hijo, para eso vamos a utilizar la función que creamos arriba

my.component.ts

```
import { Component } from '@angular/core';
import { InputComponent } from './input/input.component';

@component({
    selector: 'my-component',
    template: ' <my-component-hijo
(funcionParaRecibirVariable)="funcionParaRecibirVariable($event)" variableRecibida="
{{variableAenviar}}"></my-component-hijo> '
})

export class MyComponent{

    variableAenviar: string;
```

```
    variableRecibidaDelHijo: string;

    constructor (){}

    ngOnInit(){
        this.variableAenviar = 'Soy un string, enviado al componente hijo';
    }

    funcionParaRecibirVariable(_variableRecibida:string){
        this.variableRecibidaDelHijo = _variableRecibida;
    }


}
```

Ahora lo que tenemos que hacer es modificar el componente hijo para enviar datos, para ello vamos a tener que importar el Output y el EventEmitter ya que son los que van a enviar los datos y emitirlo.

Ademas vamos a agregar una función que llame al annotation para enviar el mensaje y por ultimo agregar una función a nuestro HTML para poder llamar a la función que envía el mensaje

my.component.hijo.ts

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
@Component({
    selector: 'my-component-hijo',
    template: ' <span (click)="enviarMensajeAlPadre()"> {{ variableRecibida }}  </span>'
})

export class MyComponentHijo{

    @Input() variableRecibida: string;

    @Output() funcionParaRecibirVariable = new EventEmitter<string>();

    constructor(){

    }

    enviarMensajeAlPadre():void{
        this.funcionParaRecibirVariable.emit('Soy un mensaje para el componente padre');
    }

}
```

Entonces de esta forma vamos a poder enviar y recibir datos entre ambos componentes, siempre y cuando sean padre e hijo.

B I U | Insertar código | Enlace | Imagen

Deja tu comentario

Suma tu comentario

Top Nuevas

5 **andresnator** 28 Puntos — 2 meses

Buena la aclaración, cuesta un poco cambiar la manera de pensar entre la comunicación de componentes pero con la practica todo se alcanza

Responder

4 **eleanatorresortiz** 0 Puntos — 2 meses

Excelente el contenido y las explicaciones hasta ahora!! Sería interesante que se pudiese descargar estos documentos resumen en PDF. 😊

Responder

1 **jeissonesteban1** 13 Puntos — 2 meses

Gracias, mucho más claro! 😊, lo ideal es crear componentes lo más sencillos posibles para que su responsabilidad sea única.

Responder

0 **natancho89** 0 Puntos — 3 meses

Gracias por el material, esto refuerza el video anterior!. Tengo una pregunta que me surgió viendo el material, ¿existe alguna diferencia entre inicializar una variable en el constructor o en en método ngOnInit?
😊

2 **razpeitia_** — 3 meses

Si, tal vez quieras ver esto.
http://stackoverflow.com/questions/35763730/difference-between-constructor-and-ngoninit/35763811#35763811

Responder

0 **pabloge** 0 Puntos — 2 meses

Se agradece la ampliación. 😊

Por favor, cuiden las faltas de ortografía.

Responder