



Florida
Universitària

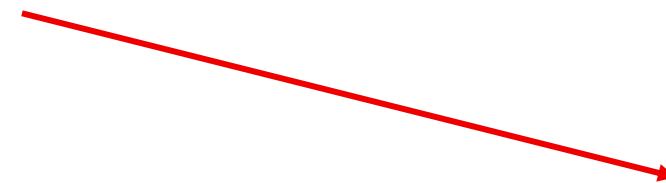
Comunicación entre Componentes

Curso 2025/2026

Paco Segura

Comunicación entre Componentes

- Hasta ahora hemos visto componentes ‘simples’.
- Ejemplo: Componente ‘Card’



Comunicación entre Componentes

- Ahora veremos cómo ‘parametrizarlo’ para poder reutilizarlo a lo largo de la aplicación y pasárselle distintos datos:



Fresas

2,85€/kg



Plátanos

1,95€/kg

Comunicación entre Componentes

- Cuando tenemos un componente que contiene otros componentes, automáticamente se establece una jerarquía entre ellos.
 - Dicha jerarquía sigue las mismas reglas que los elementos HTML del DOM.
 - Ejemplo: si instanciamos el componente ‘Card’ en una vista, la vista automáticamente es ‘padre’ del componente ‘Card’. Y viceversa: el componente ‘Card’ es hijo de la vista en la que ha sido instanciado.
-

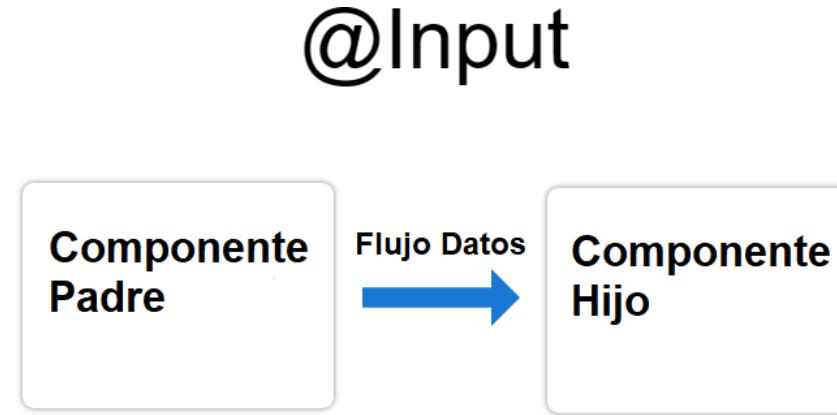
Comunicación entre Componentes

- Es imprescindible que los componentes puedan ‘comunicarse’ para poder pasarse información *-datos-* entre ellos.
 - Tenemos dos formas de comunicar datos entre componentes:
 - 1 - De Componente Padre a Componente Hijo.
 - 2 - De Componente Hijo a Componente Padre.
-

Comunicación entre Componentes

1 - Comunicación de Componente Padre a Componente Hijo:

Utilizamos el decorador @Input:



Comunicación entre Componentes

- Vamos a ver como se realizaría la comunicación entre la vista ViewOne (Padre) y el componente 'Card' (hijo).
- Pasos:
 - Creamos la vista ViewOne y el componente Card.
 - En el Componente Hijo (Card), importamos Input de la librería 'core' de Angular:

```
import { Component, Input } from '@angular/core';
```

Comunicación entre Componentes

- Definimos en el componente Hijo (Card) las propiedades que queramos '*parametrizar*' provenientes del Componente Padre.
 - En este caso la foto, el nombre de la fruta y el precio:
-

Comunicación entre Componentes

```
<div class="photo">
  <div class="image-container">
    
  </div>
  <span class="caption">Fresas</span>
  <p class="title">2,85€/kg</p>
</div>
```

Comunicación entre Componentes

- Por ello, definimos las variables ‘photo’, ‘fruit’ y ‘price’ en la clase TypeScript del componente Title precedida por el decorador @Input:

```
@Input() photo: string = '';
@Input() fruit: string = '';
@Input() price: string = '';
```

- A continuación, definimos en la parte del HTML del componente Card una variable ‘title’ con el contenido del título:



Comunicación entre Componentes

```
<div class="photo">
  <div class="image-container">
    {{fruit}}</span>
  <p class="title">{{price}} €/kg</p>

</div>
```

Comunicación entre Componentes

- Ahora ya podemos utilizar en nuestro Componente Padre (ViewOne) el Componente Hijo (Card). Para ello, primero importamos el Componente Card en la vista ViewOne:

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-card',
  imports: [],
  templateUrl: './card.html',
  styleUrls: ['./card.css'],
})
```



Comunicación entre Componentes

- Instanciamos el componente Card en el archivo HTML de la vista ViewOne.

```
<div id="page">
    <app-card></app-card>
</div>
```



Comunicación entre Componentes

- Para poder cambiar la información del Componente Hijo (Card) desde el Componente Padre (ViewOne), definimos en el Componente Padre variables cuyo nombre sea igual a las variables que hemos definido con el decorador @Input:

```
export class ViewOne {  
  public photo: string = 'https://encrypted-  
tbn0.gstatic.com/images?q=tbn:ANd9GcTm3jFWGKHucgSVFKi_hv9422Ee8zGKkQ0dsA  
&s';  
  public fruit: string = 'Fresas';  
  public price: string = '1.99';  
}
```

Comunicación entre Componentes

- Para que este contenido se muestre, en la parte HTML del componente Padre (ViewOne), definimos lo siguiente:

```
<div id="page">
  <app-card [photo]="photo" [fruit]="fruit" [price]="price"></app-card>
</div>
```



Comunicación entre Componentes

- RESUMEN:

Componente Hijo (Card):

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-card',
  imports: [],
  templateUrl: './card.html',
  styleUrls: ['./card.css'],
})
export class Card {
  @Input() photo: string = '';
  @Input() fruit: string = '';
  @Input() price: string = '';
}
```

Componente Padre:

```
import { Component } from '@angular/core';
import { Card } from "../../components/card/card";

@Component({
  selector: 'app-view-one',
  imports: [Card],
  templateUrl: './view-one.html',
  styleUrls: ['./view-one.css'],
})
export class ViewOne {
  public photo: string = 'https://encrypted-tbn0.goo...';
  public fruit: string = 'Fresas';
  public price: string = '1.99';
}
```

Comunicación entre Componentes

Componente Hijo (Card):

```
<div class="photo">
  <div class="image-container">
    {{fruit}}</span>
  <p class="title">{{price}} €/kg</p>
</div>
```

Componente Padre:

```
<div id="page">
  <app-card [photo]="photo" [fruit]="fruit" [price]="price"></app-card>
</div>
```

Comunicación entre Componentes

- 2 - **Comunicación de Componente Hijo a Componente Padre:**
- Utilizamos los decoradores **@Output** y **EventEmitter**:

@Output



Comunicación entre Componentes

- Vamos a ver como se realizaría el ‘envío’ de un evento entre el Componente Card (Hijo) y la vista ViewOne (Padre).



Comunicación entre Componentes

- Pasos:

- Creamos el Componente Card y la vista ViewOne.
- En el Componente Hijo (Card), importamos Output y EventEmitter de la librería 'core' de Angular:

```
import { Component, Output, EventEmitter } from  
  '@angular/core';
```

Comunicación entre Componentes

- Definimos en el Componente Hijo (Card) el evento que queremos ‘enviar’ al Componente Padre. Utilizamos @Output y EventEmitter:

```
@Output() message = new EventEmitter<string>();
```

Comunicación entre Componentes

- En este caso vamos a enviar un string y lo vamos a ubicar dentro de una función que maneje un evento click, de modo que cada vez que se haga click en el título del Componente Card, se envíe un string a la vista ViewOne:

En HTML: al realizar click se llama al manejador de evento onClick()

```
<div class="photo">
  <div class="image-container">
    {{fruit}}</span>
  <p class="title">{{price}} €/kg</p>
</div>
```

En TS: al realizar click, la función onClick emite un string al componente Padre mediante .emit() aplicado a la variable definida con el decorador @Output y EventEmitter

```
@Output() message = new EventEmitter<string>();

public onClick(): void {
  this.message.emit('String enviado...')
}
```

Comunicación entre Componentes

- En el HTML de la vista ViewOne (padre), instanciamos el componente hijo (Card), y le implementamos un evento con el nombre que hemos definido con @Output y EventEmitter en el Componente Hijo (Card). El manejador del evento debe pasar por parámetro \$event para poder recibir la información pasada desde el hijo (en este caso un string):

```
<app-card (message)="onClick($event)"></app-card>
```

Comunicación entre Componentes

- En la clase TypeScript creamos el manejador del evento - onClick- e implementamos una lógica con la información que se recibe por parámetro proveniente del componente hijo. En este caso una alerta mostrando el contenido del string:

```
public onClick(message: string): void {  
    alert(message);  
}
```



Comunicación entre Componentes

- RESUMEN :

Componente Hijo:

```
import { Component, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-card',
  imports: [],
  templateUrl: './card.html',
  styleUrls: ['./card.css'],
})
export class Card {
  public photo: string = '';
  public fruit: string = 'fresas';
  public price: string = '1.99';
  @Output() message = new EventEmitter<string>();

  public onClick(): void {
    this.message.emit('String enviado...')
  }
}
```

Componente Padre:

```
import { Component } from '@angular/core';
import { Card } from '../../components/card/card';

@Component({
  selector: 'app-view-one',
  imports: [Card],
  templateUrl: './view-one.html',
  styleUrls: ['./view-one.css'],
})
export class ViewOne {

  public onClick(message: string): void {
    alert(message);
  }
}
```

Comunicación entre Componentes

Componente Hijo:

```
<div class="photo">
  <div class="image-container">
    <img src={{photo}}>
  </div>
  <span (click)="onClick()" class="caption">{{fruit}}{{price}} €/kg</p>
</div>
```

Componente Padre:

```
<div id="page">
  <app-card (message)="onClick($event)"></app-card>
</div>
```

Comunicación entre Componentes

- Trabajando con **@Input** y **@Output** a la vez.
 - Podemos combinar ambos decoradores partiendo del ejemplo anterior para poder implementar una lógica más compleja. Ejemplo: que cuando se haga click sobre el Componente Card se cambie su contenido.
-

Comunicación entre Componentes

- Partiendo del ejemplo anterior, al componente Card le añadimos el decorador @Input a las variables para poder acceder a sus valores desde la vista ViewOne:

```
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-card',
  imports: [],
  templateUrl: './card.html',
  styleUrls: ['./card.css'],
})
export class Card {
  @Input() photo: string = '';
  @Input() fruit: string = '';
  @Input() price: string = '';

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

  public onClick(): void {
    this.message.emit('https://images-prod.healthline.com/hlcmsresource/images/AN_images/strawberries-1296x728-feature.jpg')
  }
}
```

Comunicación entre Componentes

- En la vista ViewOne: implementamos la lógica para poder mostrar el nuevo contenido enviado desde Card. Utilizamos @Input para mostrarlo:



Comunicación entre Componentes

HTML

```
<div id="page">
    <app-card (message)="onClick($event)" [fruit]="fruit" [photo]="photo" [price]="price"></app-card>
</div>
```

Comunicación entre Componentes

TypeScript

```
import { Component } from '@angular/core';
import { Card } from "../../components/card/card";

@Component({
  selector: 'app-view-one',
  imports: [Card],
  templateUrl: './view-one.html',
  styleUrls: ['./view-one.css'],
})
export class ViewOne {
  public photo: string = 'https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcTm3jFWGKHucgSVFKi_hv9422Ee8zGKkQ0dsA&s';
  public fruit: string = 'fresas';
  public price: string = '1.99';

  public onClick(message: string): void {
    this.photo = message;
  }
}
```