Decorador @Input en Angular

Uso de `@Input` en Angular

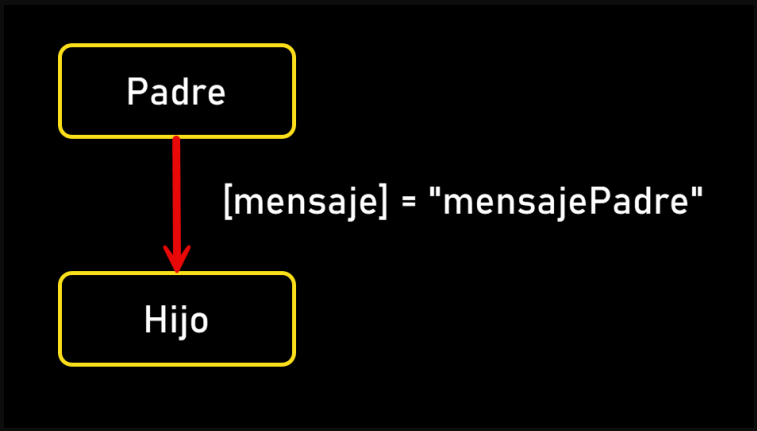
En esta lección, aprenderás a utilizar el decorador `@Input` en Angular para pasar datos de un componente padre a un componente hijo. Entenderás la sintaxis básica, cómo funciona, y veremos un ejemplo práctico para aplicar este concepto.

1. ¿Qué es `@Input` en Angular?

Un **decorador** en Angular es una función especial que se utiliza para añadir metadatos a clases, métodos, propiedades, o parámetros. Estos metadatos son esenciales para que Angular pueda entender cómo debe manejar y comportarse un elemento específico en la aplicación.

El decorador `@Input` en Angular permite que un componente hijo reciba datos desde su componente padre. Es una de las formas más comunes de comunicación entre componentes en Angular.

- **Componente Padre:** El componente que contiene o envuelve a otro componente (componente hijo).
- **Componente Hijo:** El componente que recibe datos desde el componente padre.



2. Sintaxis Básica de `@Input`

Para utilizar `@Input`, debes seguir estos pasos:

1. **Importar el Decorador `@Input`:**

   En el componente hijo, importa `@Input` desde el paquete `@angular/core`.

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

2. **Declarar la Propiedad Decorada con `@Input`:**

   En el componente hijo, usa `@Input` para declarar una propiedad que recibirá el valor desde el componente padre.

```
export class HijoComponent {
  @Input() mensaje!: string;
}
```

3. **Pasar el Valor desde el Componente Padre:**

   En la plantilla del componente padre, pasa el valor al componente hijo utilizando la vinculación de propiedades.

```
<app-hijo [mensaje]="mensajePadre"></app-hijo>
```

   En el componente padre:

```
export class PadreComponent {
      mensajePadre: string = 'Hola desde el Padre';
}
```

3. Ejemplo Completo: Comunicación Padre-Hijo

Vamos a crear un ejemplo completo donde un componente padre pase un mensaje a un componente hijo.

Paso 1: Crear el Componente Hijo

Primero, crea un componente hijo que reciba un mensaje desde su componente padre.

Código del Componente Hijo (`hijo.component.ts`)

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

@Component({
  selector: 'app-hijo',
  standalone: true,
  imports: [],
  templateUrl: './hijo.component.html',
  styleUrl: './hijo.component.css'
})
export class HijoComponent {
  // !:operador non-null assertion operator en TypeScript.
  // confíe en que esta propiedad será inicializada
  // Se inicializa desde el componente Padre
  @Input() mensaje!: string;

}
```

Código del Componente Hijo (`hijo.component.html`)

```html
<div class="alert alert-info w-50 mx-auto">
    <p>{{ mensaje }}</p>
</div>
```

Paso 2: Crear el Componente Padre

Ahora, crea un componente padre que pasará un mensaje al componente hijo.

Código del Componente Padre (`padre.component.ts`)

```ts
import { Component } from '@angular/core';
import { HijoComponent } from "./hijo/hijo.component";

@Component({
  selector: 'app-padre',
  standalone: true,
```

```
  imports: [HijoComponent],
  templateUrl: './padre.component.html',
  styleUrl: './padre.component.css'
})
export class PadreComponent {
  mensajePadre: string = 'Mensaje desde el Componente Padre';
}
```

Código del Componente Padre (`padre.component.html`)

```
<p>Componente Padre</p>
<app-hijo [mensaje]="mensajePadre"/>
```

Paso 3. Resultado Esperado

Cuando el componente padre se renderiza, pasará el texto `Mensaje desde el Componente Padre` al componente hijo, y el componente hijo mostrará este mensaje dentro de una alerta estilizada.

Paso 4. Puntos Clave para Recordar

- **Tipos de Datos en `@Input`:** Puedes pasar cualquier tipo de datos a través de `@Input`, incluyendo strings, números, objetos, arrays, y más.
- **Cambios en los Datos:** El componente hijo reaccionará automáticamente a los cambios en los datos que se le pasen desde el componente padre. Si el valor en el componente padre cambia, el componente hijo se actualizará automáticamente para reflejar ese cambio.
- **Nombre del `@Input`:** Puedes personalizar el nombre del `@Input` si deseas que el nombre de la propiedad en la plantilla sea diferente al de la clase:

```
@Input('nombreEnPlantilla') mensaje!: string;
```

Conclusión

El decorador `@Input` es esencial para comunicar datos desde un componente padre a un componente hijo en Angular. Comprender este concepto te permitirá construir aplicaciones más modulares y reutilizables.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx