ANGULAR

¿Como se envía información entre componentes?

Input/Output Properties:

Los componentes pueden comunicarse a través de las propiedades de entrada y salida. Un componente padre puede pasar datos a un componente hijo mediante la vinculación de propiedades de entrada, y un componente hijo puede emitir eventos que el componente padre puede escuchar a través de propiedades de salida.

ANGULAR

COMUNICACIÓN ENTRE COMPONENTES

INPUT

1)En el componente hijo, puedes definir propiedades de entrada utilizando el decorador @Input(). Estas propiedades representarán los datos que se esperan recibir del componente padre.

```
// Componente hijo
@Input() datoEntrada: string;
```

2) En el componente padre, puedes vincular datos a la propiedad de entrada del componente hijo utilizando la sintaxis de corchetes [] en el marcado del template.

```
<!-- Componente padre -->
<app-hijo [datoEntrada]="valorDesdePadre"></app-hijo>
```

3) Cuando el valor de la propiedad en el componente padre cambia, Angular automáticamente actualiza la propiedad de entrada en el componente hijo. Esto proporciona una forma eficiente y automática de mantener sincronizados los datos entre componentes.

```
// Componente padre
valorDesdePadre = "Hola, mundo!";
```

4) En el componente hijo, puedes utilizar la propiedad de entrada (datoEntrada en este caso) como cualquier otra propiedad local. Puedes mostrarla en el template, realizar lógica basada en ese valor, etc.

```
<!-- Componente hijo template -->
<p>{{ datoEntrada }}</p>
```

ANGULAR

OUTPUT

COMUNICACIÓN ENTRE COMPONENTES

1)Se usa @Output y EventEmitter para lograr la comunicación entre un componente hijo y su componente padre. Declaras una propiedad con @Output en el componente hijo y emites eventos con EventEmitter.

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

sendMessage() {
  this.messageEvent.emit(this.message);
}
```

2)Este archivo HTML contiene la interfaz de usuario del componente hijo. Incluye un input para que el usuario ingrese un mensaje y un botón para enviarlo. Utiliza ngModel para vincular el input con la propiedad message del componente TypeScript.

```
<div>
  <label for="childInput">Mensaje:</label>
  <input id="childInput" [(ngModel)]="message" />
  <button (click)="sendMessage()">Enviar Mensaje</button>
</div>
```

3)El archivo TypeScript define el componente ParentComponent, que tiene una propiedad (receivedMessage) para almacenar mensajes recibidos del componente hijo. Incluye un método (receiveMessage) que actualiza esta propiedad cuando se emite el evento desde el componente hijo.

```
receivedMessage: string = '';

receiveMessage(message: string) {
  this.receivedMessage = message;
}
```

4)La plantilla HTML del componente padre incluye el componente hijo (<app-child>) y utiliza el evento de salida messageEvent para llamar al método receiveMessage cuando se emite un mensaje desde el componente hijo. Muestra el mensaje recibido en la interfaz del componente padre.

```
<div>
  <app-child (messageEvent)="receiveMessage($event)"></app-child>
  <p>Mensaje recibido en el padre: {{ receivedMessage }}</p>
</div>
```