The @Output decorator in Angular is used to emit custom events from a child component to its parent component. This allows for one-way communication from the child to the parent, enabling you to notify the parent component of events or changes that occur within the child component.

**Key Concepts:**

* **Child-to-Parent Communication:** The @Output decorator facilitates the emission of custom events from the child component to its parent component.
* **EventEmitter:** The @Output decorator is typically used in conjunction with the EventEmitter class from @angular/core.
* **Event Binding:** The parent component uses event binding in its template to listen for the events emitted by the child component.

**Example:**

**Child Component (child.component.ts)**

TypeScript

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

@Component({

selector: 'app-child',

template: `

<button (click)="onButtonClicked()">Click Me</button>

`

})

export class ChildComponent {

@Output() childEvent = new EventEmitter<any>();

onButtonClicked() {

this.childEvent.emit('Child Component Button Clicked!');

}

}

**Parent Component (parent.component.ts)**

TypeScript

import { Component } from '@angular/core';

@Component({

selector: 'app-parent',

template: `

<app-child (childEvent)="onChildEvent($event)"></app-child>

<p>Parent Component: {{ parentMessage }}</p>

`

})

export class ParentComponent {

parentMessage: string = '';

onChildEvent(message: string) {

this.parentMessage = message;

}

}

**Explanation:**

1. **Child Component:**
   * @Output() childEvent = new EventEmitter<any>(); declares an EventEmitter named childEvent that will be used to emit events.
   * The onButtonClicked() method emits the message "Child Component Button Clicked!" using this.childEvent.emit().
2. **Parent Component:**
   * (childEvent)="onChildEvent($event)" binds the childEvent emitted by the child component to the onChildEvent() method in the parent component.
   * The onChildEvent() method receives the emitted message and updates the parentMessage property.

**Benefits:**

* **Component Interaction:** Enables effective communication between child and parent components.
* **Modular Design:** Promotes a more modular and decoupled component architecture.
* **Flexibility:** Allows for various types of events to be emitted from child components.

By effectively using the @Output decorator, you can create well-structured Angular applications with flexible and efficient communication between components.