Passing Data between Components

This is a **simple example of Parent-Child Component Communication in Angular**.

Explanation of the Setup

1. Child Component (child.component.ts + child.component.html)

---

child.component.html (template)

```
<h2>Product ID: <span style="color: red;">{{p_id}}</span></h2>
<h2>Product Name: <span style="color: red;">{{p_name}}</span></h2>
<h2>Product Cost: <span style="color: red;">{{p_cost}}</span></h2>
<button (click)="clickMe()">Send</button>
<hr>
```

This is a simple display of **product details** (**id, name, cost**), along with a button labeled "**Send**". When you click the button, it triggers the **clickMe()** function.

---

child.component.ts (class)

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

@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styles: []
})
export class ChildComponent {
```

```
  @Input() p_id: any;
  @Input() p_name: any;
  @Input() p_cost: any;

  @Output() send: EventEmitter<any> = new EventEmitter();

  clickMe(): any {
    this.send.emit(this.p_id + "...." + this.p_name + "...." + this.p_cost);
  }
}
```

Key points:

@Input() allows data to flow **from Parent to Child** (each product's details).
@Output() along with **EventEmitter** allows sending data **from Child to Parent**.
clickMe() combines product data into a string and **emits it** using **send.emit().**

---

2. Parent Component (parent.component.ts + parent.component.html)

---

parent.component.html (template)

```
<app-child
  *ngFor="let x of products"
  [p_id]="x.p_id"
  [p_name]="x.p_name"
  [p_cost]="x.p_cost"
  (send)="myFun($event)">
</app-child>
```

Here, the parent loops over products array and renders a <app-child> for **each product**.

- **[p_id], [p_name], [p_cost]** bind parent's product data to child's **@Input** properties.
- **(send)** listens for the send event emitted by the child and calls **myFun()** in the parent, passing the emitted data as **$event.**

---

parent.component.ts (class)

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

@Component({
  selector: 'app-parent',
  templateUrl: './parent.component.html',
  styles: []
})
export class ParentComponent {
  public products: Array<any> = [
    { p_id: 111, p_name: "TV", p_cost: 20000 },
    { p_id: 222, p_name: "Fridge", p_cost: 40000 },
    { p_id: 333, p_name: "Iphone", p_cost: 150000 },
    { p_id: 444, p_name: "Fan", p_cost: 1200 },
    { p_id: 555, p_name: "Dish Washer", p_cost: 50000 }
  ];

  public myFun(data: any) {
    alert(data);
  }
}
```

Key points:

Parent owns the **products** array.
Parent renders multiple children dynamically using ***ngFor.**
Parent listens to **child events** using (send) and processes the data using **myFun().**

Overall Flow

Step	What Happens
1	Parent renders app-child for each product.
2	Each child receives a product's details via @Input().
3	Clicking "Send" in child triggers clickMe(), emitting the product data back to parent.
4	Parent catches the event and handles it using myFun(), showing the data in an alert.

Why This Matters?

This is the **foundation of Parent-Child communication** in Angular.

- **@Input()** handles **Data Flow Down** (Parent → Child)
- **@Output()** handles **Data Flow Up** (Child → Parent)
