Communication between the Components in Angular will help you pass data from child components to parent components and vice-versa (Goudar, 2019).

Use input binding with @Input decorations to pass data from the parent component to the child component. In order to pass data, there are a few steps involved. In the parent component, add an attribute to the tag inside of the parent’s template, named after the selector in the child component. In the child component, add a variable with the name of the attribute called in the parent tag from the previous step, and use it to bind the element in the template. Let’s do an example to bring clarity to these steps:

We’ll have two components: parent and child for the sake of simplicity.

**Parent Component**

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

*@Component({*

*selector: 'my-app',*

*template: `*

*<child-comp [parentCount]="count"></child-comp>*

*`*

*})*

*export class App {*

*count: number = 10;*

*}*

**Child Component**

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

*@Component({*

*selector: 'child-comp',*

*template: `*

*<div> {{ parentCount }} </div>*

*`*

*})*

*export class ChildComponent {*

*@Input() parentCount: number;*

*}*

In order to pass data back from child to parent, you must use EventEmitter along with @Output so the parent can capture the exposed event and retrieve data.

*//import Output, EventEmitter from angular core*

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

*//define the event emitter*

*@Output() change: EventEmitter<number> = new EventEmitter<number>();*

*//emit the event with the above change event this.change.emit(11); //11 will be emitted to the parent*

Using the example for @Input, let’s push back the updated content to the parent. We will then declare a change event with the type of EventEmitter. The event emitter will use emit() to emit to the parent.

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

*//import the child componentimport*

*{ChildComponent} from 'src/child.component'*

*//add (change) event to the child component selector.*

*//(change) is the name of the event that is emitted from the child component*

*@Component({*

*selector: 'parent',*

*template: `*

*<child-comp [parentCount]="count" (change)="updateFromChild($event)"></child-comp>*

*`*

*});*

*export class ParentComponent {*

*count: number = 0;*

*//initial value of count //This is the event that we subscribe to the child's selector*

*//We will receive the data from child in this function*

*updateFromChild($event){*

*this.count++;*

*}*

*}*

It takes quite a few steps to make this happen, but it’s quite powerful, and makes for clean code.

**REFERENCES**

@Input and @Output in Angular. (2017, August 10). Retrieved from https://angularman.wordpress.com/2017/08/10/input-and-output-in-angular/

Goudar, Y. (2019, January 29). Yallaling Goudar. Retrieved from https://www.pluralsight.com/guides/angular-communication-between-components-input-output-properties