# Sharing data between child and parent directives and components

A common pattern in Angular is sharing data between a parent component and one or more child components. You can implement this pattern by using the @[Input](https://angular.io/api/core/Input)() and @[Output](https://angular.io/api/core/Output)() directives.

<https://angular.io/guide/inputs-outputs>

*@Input and @Output* decorators.  Both are use to transform the data from one component to another component.  Or, you can say pass the different types of data form parent to child component and child to parent component.

Or

In a simple way, transform/exchange data between two components.

Introduction

In this article, I am exploring two very important points, related to Angular 2 + version, which the part of Parameter Decorator and these points are called *@Input and @Output decorators*.  Both are used to transform the data from one component to another component.  Or, you can say pass the different types of data from parent to child component and child to parent component.  Or, in a simple way transform/exchange data between two-component.

Let's explore each, one by one.

@Input Decorator

@Input is a decorator to mark a property as an input.  @Input is used to define an input property, to achieve component property binding.  @Inoput decorator is used to pass data (property binding) from parent to child component.  The component property should be annotated with @Input decorator to act as input property.

Let's explore it practically.

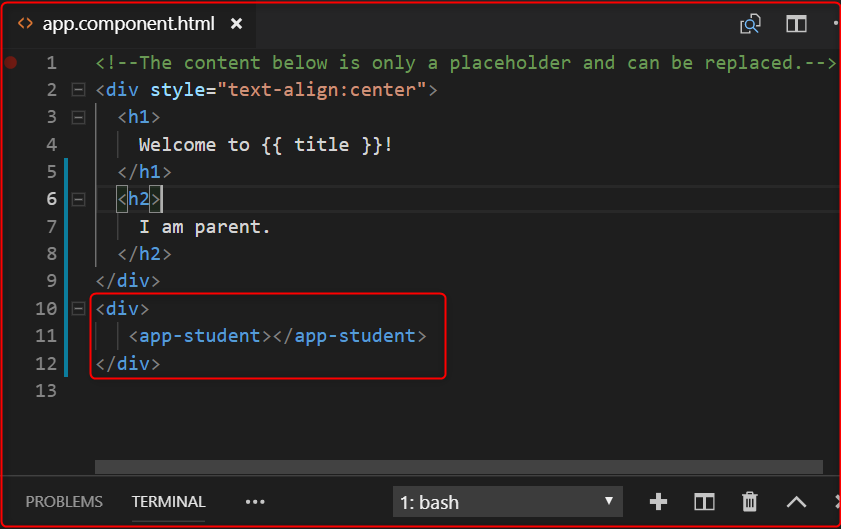
I have created an angular application which is AngApp.  I have created two components.  They are app components and student components.  I will transfer the data from parent to child component, using @Input decorator.  I am assuming my, app-component is the parent component and student-component is the child component.

To make the parent-child relation, keep the instance (selector of student component) of student component inside the template URL (app.component.html) of app component.

Open app.component.html:  Inside this file, we keep an instance of student component.

**Example**

1. <div> <app-student></app-student></div>



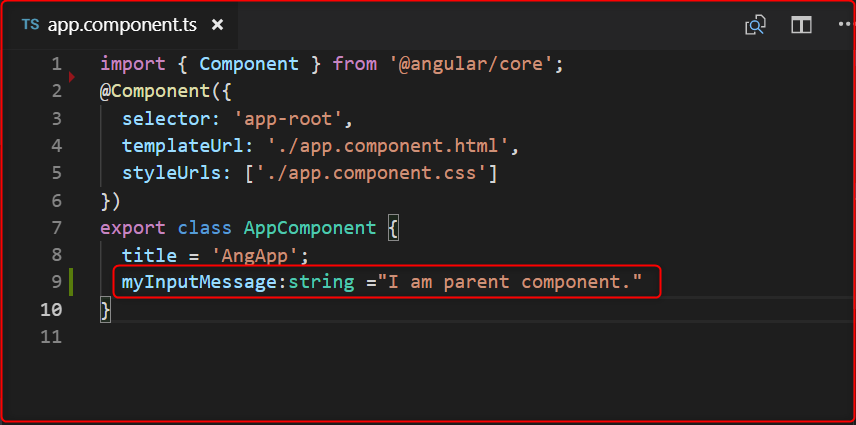
Listing 1.0

In the above image, the selected area is the child component.

Now, we want to send the message from parent to child component.

Let's open the parent component's .ts file (app.component.ts) and declare a variable inside AppComponent class, to store the message.  This message is received by the child component.

1. myInputMessage:string ="I am the parent comppnent"

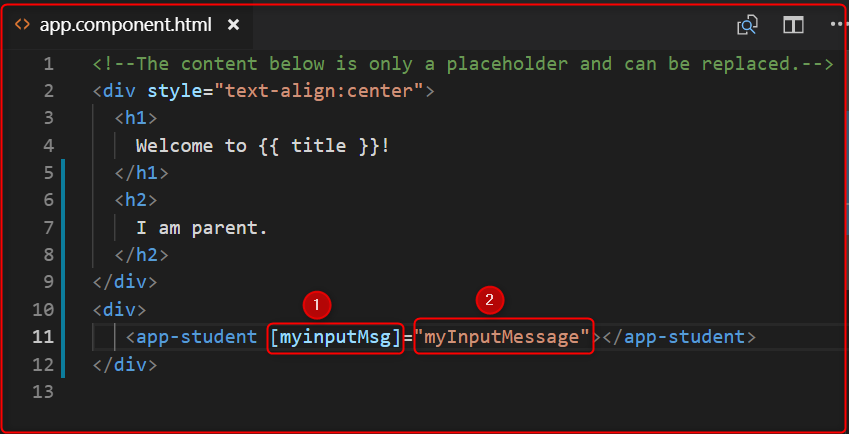


1. **import** { Component, Input, OnInit } from '@angular/core';
3. @Component({
4. selector: 'app-student',
5. templateUrl: './student.component.html',
6. styleUrls: ['./student.component.css']
7. })
9. **export** **class** StudentComponent **implements** OnInit {
10. @Input() myinputMsg:string;
12. constructor() { }
14. ngOnInit() {
15. console.log(**this**.myinputMsg);
16. }
18. }

In the above image, we have declared a variable( myInputMessage) shown in the selected area.

Now, let's open parent component views (app.component.html) pass this variable inside child component instance, which is passed inside parent component.

1. <div>
2. <app-student [myinputMsg]="myInputMessage"></app-student>
3. </div>



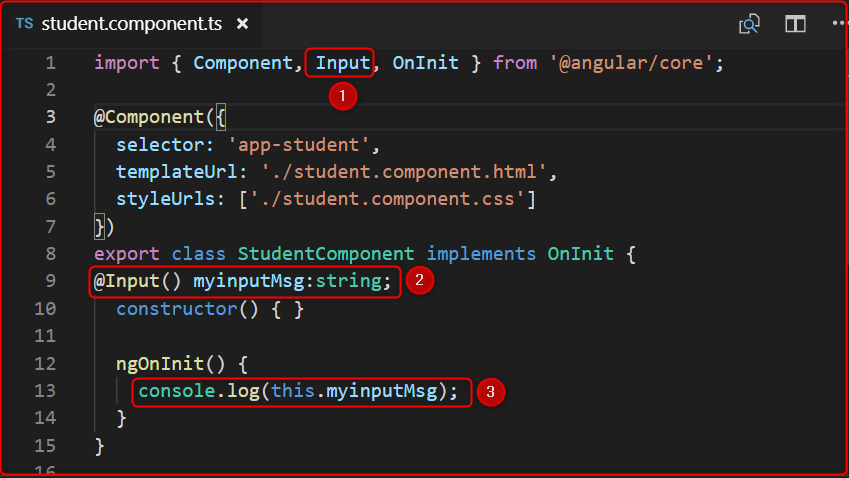
The above image, represents 2 points.  Let's explain each of the points.

1. Denotes those variables which will be used by child component (student component) with @Input decorator to fetch the message from parent component and,
2. denotes those variables which are passed the parent component message to child component.

Now, open the child component's .ts file (student.component.ts) and import Input decorator, using the myinputMsg variable with @Input decorator and print it inside constructor or ngOnInit().

1. **import** { Component, Input, OnInit } from '@angular/core';
3. @Component({
4. selector: 'app-student',
5. templateUrl: './student.component.html',
6. styleUrls: ['./student.component.css']
7. })
9. **export** **class** StudentComponent **implements** OnInit {
10. @Input() myinputMsg:string;
12. constructor() { }
14. ngOnInit() {
15. console.log(**this**.myinputMsg);
16. }
18. }

**Output**



**Output**

In image 4, represent 3 points. Let's explain each of the points.

1. First import the Input decorator, which is provided by angular and full path is @anuglar/core.
2. Use @Input decorator and declare those variables which are passed by parent component Html(app.component.html) file's point 1.  When we declare those variable (myinputMsg) with @Input decorator it automatically fetches the value of the parent component variable with the help of @Input decorator.
3. Print the values of this variable inside constructor or ngOnInit(). We have used inside ngOnInit().

Let's output,



@Output Decorator

@Output decorator is used to pass the data from child to parent component.  @Output decorator binds a property of a component, to send data from one component to the calling component.  @Output binds a property of the type of angular EventEmitter class.

To transfer the data from child to parent component, we use @Output decorator.

Lets's Open the child component' .ts file (student.component.ts).

For use the @Output decorator we have to import, two important decorators, they are Output and EventEmitter.

EventEmitter

Use in components with the @Output directive to emit custom events synchronously or asynchronously, and register handlers for those events by subscribing to an instance.

1. **import** { Component, Input, Output,EventEmitter, OnInit } from '@angular/core';

Now, create any variable with @Output decorator.

1. @Output() myOutput:EventEmitter<string>= **new** EventEmitter();

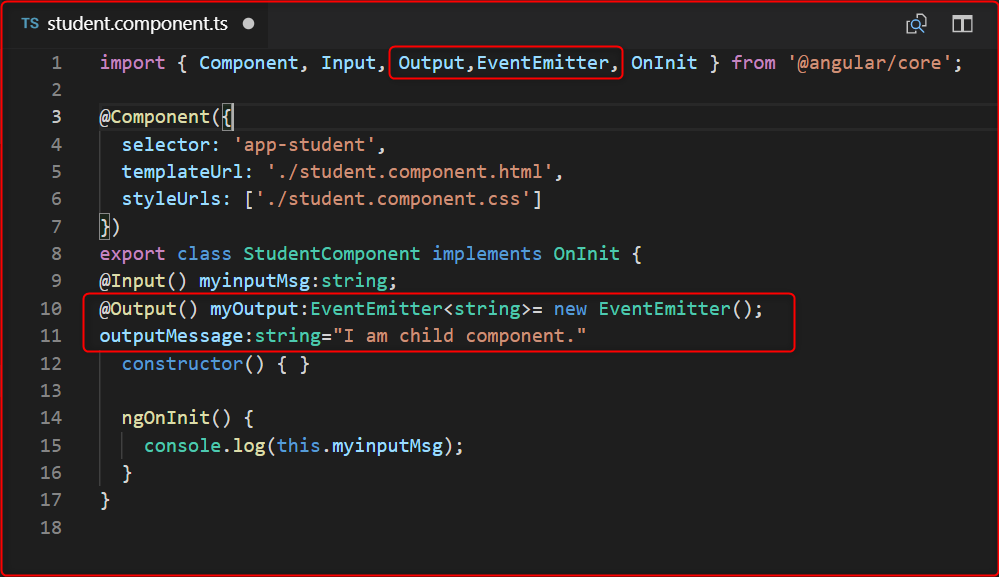
Here in the place of string, we can pass different types of DataTypes.

After that create a variable to store and pass the message to the parent component.

1. outputMessage:string="I am child component."

**Code**

1. **import** { Component, Input, Output,EventEmitter, OnInit } from '@angular/core';
3. @Component({
4. selector: 'app-student',
5. templateUrl: './student.component.html',
6. styleUrls: ['./student.component.css']
7. })
8. **export** **class** StudentComponent **implements** OnInit {
9. @Input() myinputMsg:string;
10. @Output() myOutput:EventEmitter<string>= **new** EventEmitter();
11. outputMessage:string="I am child component."
12. constructor() { }
14. ngOnInit() {
15. console.log(**this**.myinputMsg);
16. }
17. }



Send the value of output message, to the parent component.  Then we create a button and click on this button.  We will send the values to the parent component.

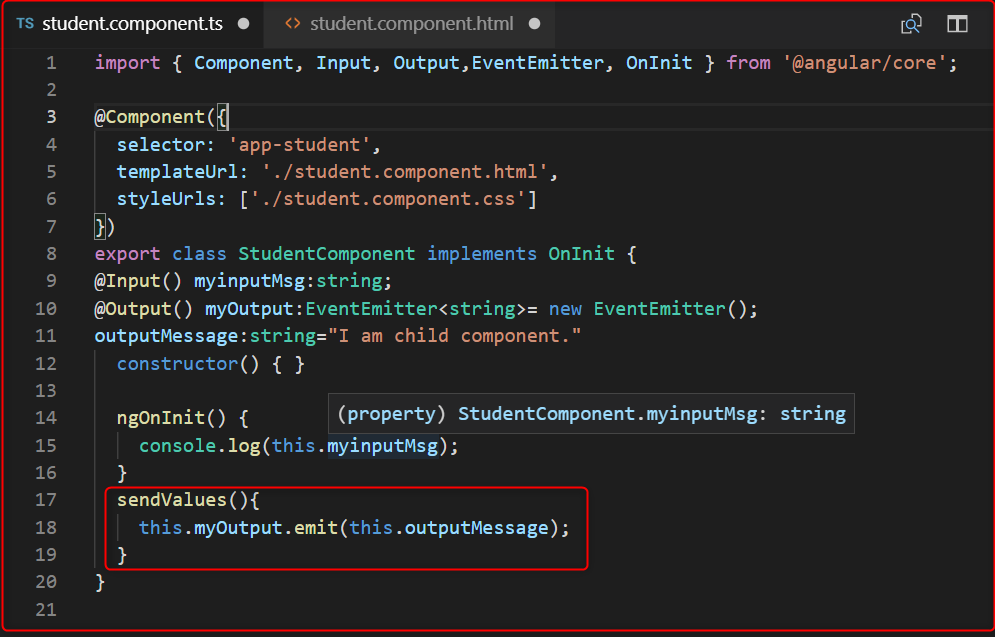
Let's open the child component Html page and create a button and click event of this button.  We then send the values.

**student.component.html**

1. <button (click)="sendValues"> Send Data </button>

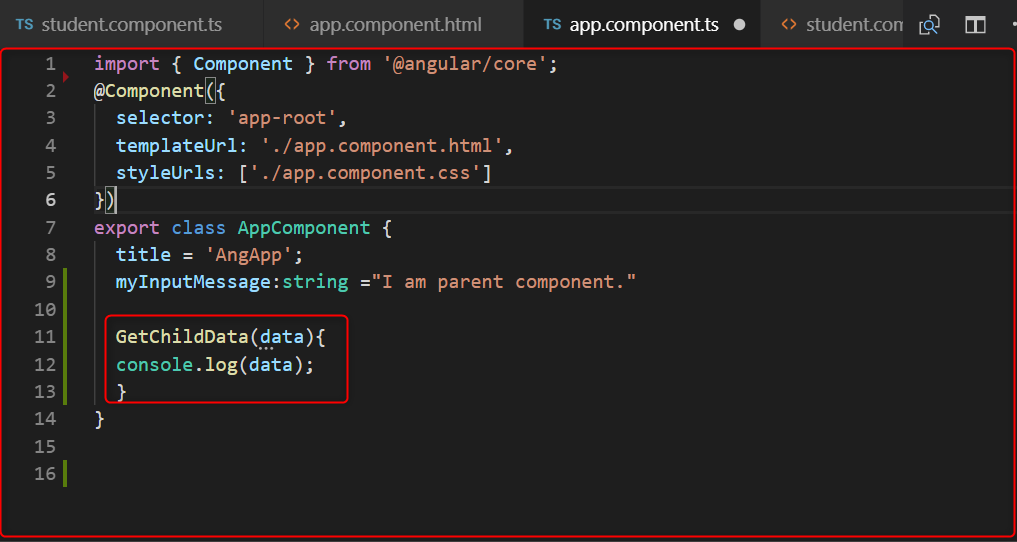
 Now fire the click on student.component.ts.

1. sendValues(){
2. **this**.myOutput.emit(**this**.outputMessage);
3. }



Now, to fetch the value we have to go *app.component.html* file and use the below code.

1. <app-student [myinputMsg]="myInputMessage" (myOutput) ="GetChildData($event)"></app-student>



1. **function** which is GetChildData() on parent component' .ts file, **for** fetch the data from child component.
2. Open the app.component.ts:
4. Code:
6. GetChildData(data){
7. console.log(data);
8. }

