@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. @Input 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.

They are app components and student components. I will transfer the data from the parent to the 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 the student component inside the template URL (app.component.html) of the app component.

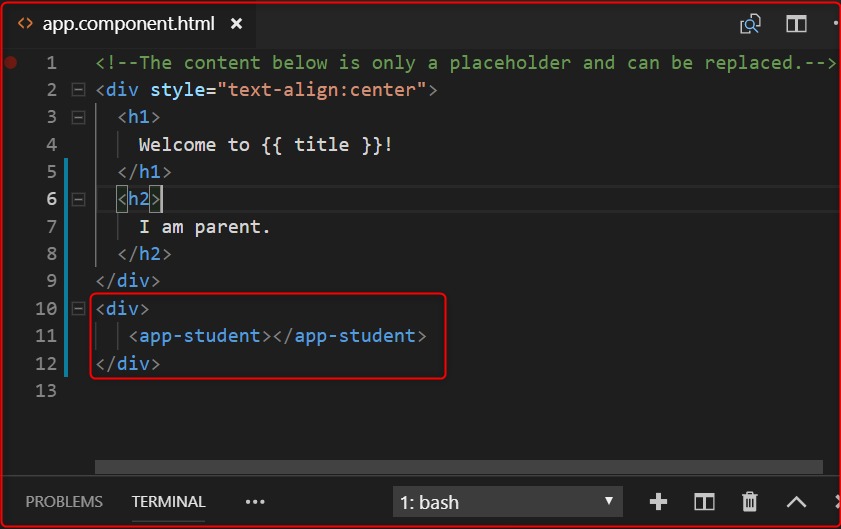
Open app.component.html: Inside this file, we keep an instance of the student component.

Example

<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 the AppComponent class, to store the message. This message is received by the child component.

myInputMessage: string = "I am the parent component";



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

@Component({

selector: 'app-student',

templateUrl: './student.component.html',

styleUrls: ['./student.component.css']

})

export class StudentComponent implements OnInit {

@Input() myinputMsg: string;

constructor() { }

ngOnInit() {

console.log(this.myinputMsg);

}

}

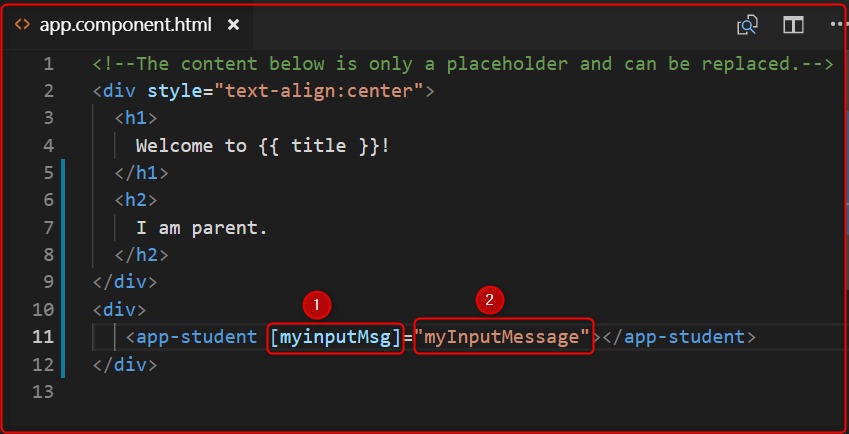
In the above image, we have declared a variable( myInputMessage) shown in the selected area.

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

<div>

<app-student [myinputMsg]="myInputMessage"></app-student>

</div>



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

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

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

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

@Component({

selector: 'app-student',

templateUrl: './student.component.html',

styleUrls: ['./student.component.css']

})

export class StudentComponent implements OnInit {

@Input() myinputMsg: string;

constructor() { }

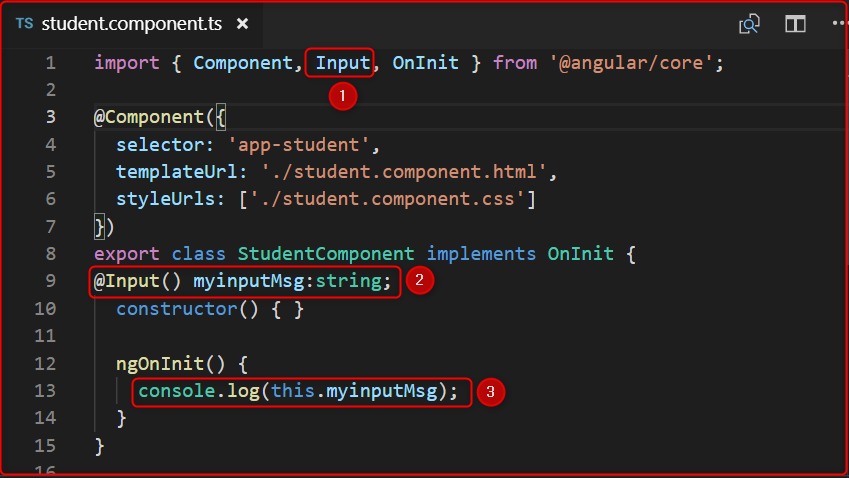
ngOnInit() {

console.log(this.myinputMsg);

}

}

**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 the full path is @anuglar/core.
2. Use @Input decorator and declare those variables which are passed by the parent component HTML (app.component.html) file's point 1. When we declare that 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 the constructor or ngOnInit(). We have used inside ngOnInit().

Let's output,



@Output Decorator

@Output decorator is used to pass the data from the child to the 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 the child to the parent component, we use the @Output decorator.

Let'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.

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

Now, create any variable with the @Output decorator.

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

Co

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.

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

Code

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

@Component({

selector: 'app-student',

templateUrl: './student.component.html',

styleUrls: ['./student.component.css']

})

export class StudentComponent implements OnInit {

@Input() myinputMsg: string;

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

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

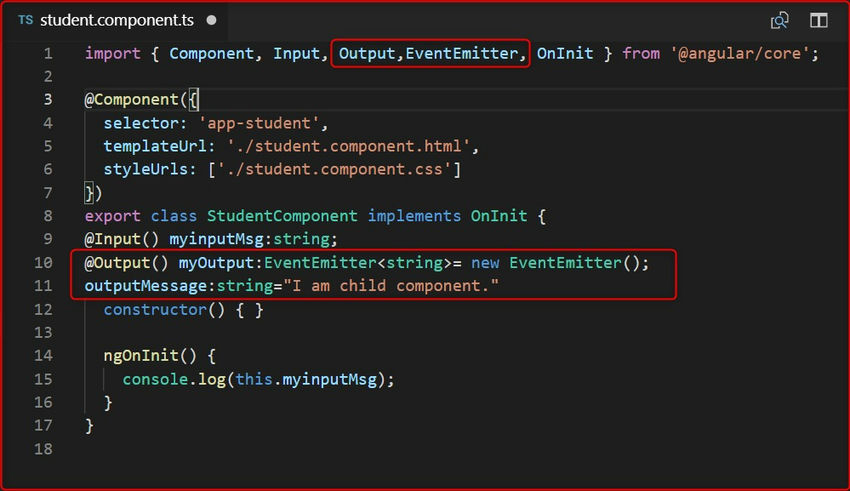
constructor() { }

ngOnInit() {

console.log(this.myinputMsg);

}

}



Send the value of the 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 the event of this button. We then send the values.

student.component.html.

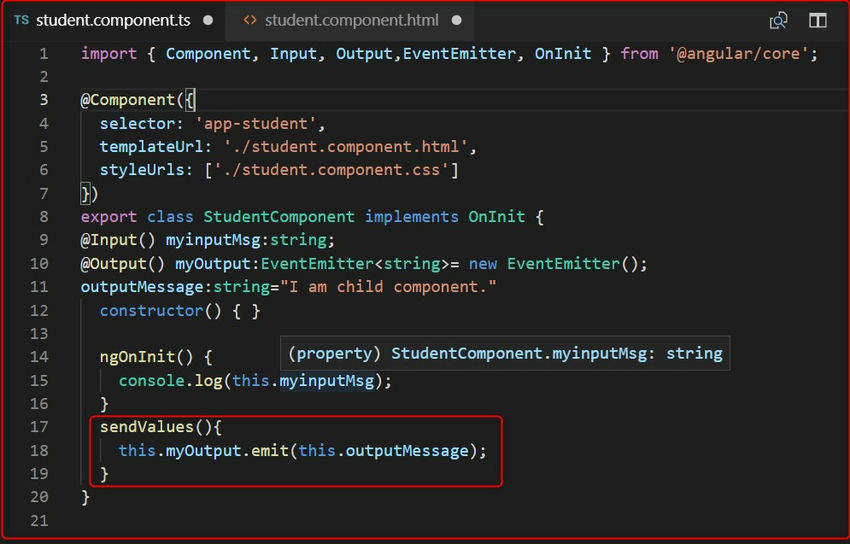
<button (click)="sendValues"> Send Data </button>

Now fire the click on student.component.ts.

sendValues() {

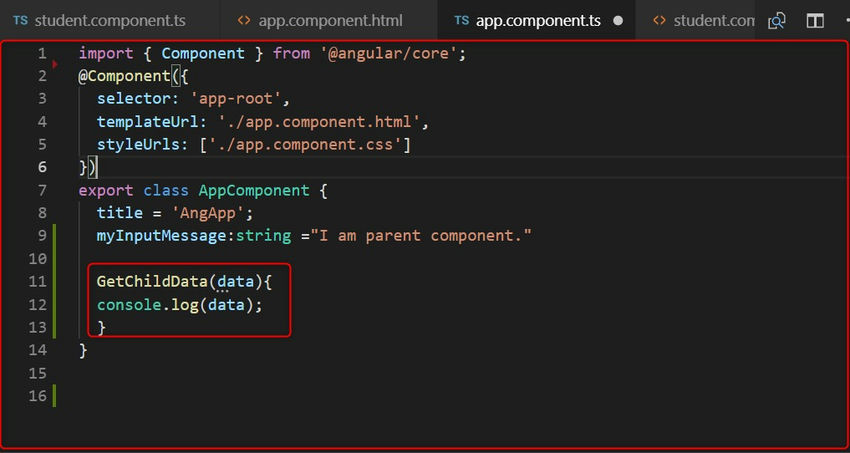
this.myOutput.emit(this.outputMessage);

}



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

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



function which is GetChildData() on parent component' .ts file, for fetch the data from child component.

Open the app.component.ts:

Code:

GetChildData(data){

console.log(data);

}

