Micro-Services

❖ What is Decorators in Angular?

Decorators are function that are invoked with a prefix @ symbol , and immediately followed by class, function (method) , or property (data member) ,

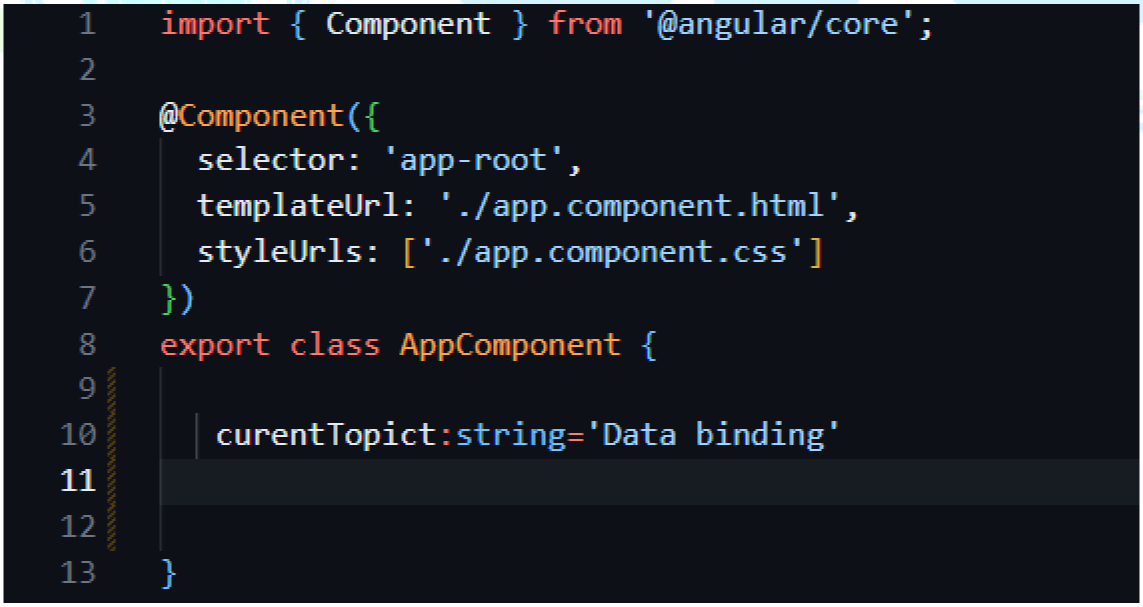
Decorators add extra functionality in our code. Basically decorators informs to angular compiler how that piece of code should be treated while executing.

❖ Class Decorators : -

A class decorator tells Angular , why that particular class is all about. Is decorated class is angular component class ? or it is angular module class ?

There are various class decorators in Angular, and among them, @Component and @NgModule are widely used.

Below is the snippet to represent @ component directive.



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

   curentTopict:string='Data binding'


}
```

Micro-Services

❖ Property Decorator : -

The property decorator is use to decorate specific properties inside a class. Which adds some extra behaviour to our class properties.

@Input , @Output and @ViewChild are some property decorators widely used decorators in angular to achieve component to component communication.(parent to child and vice versa).

1. @Input( ) :-

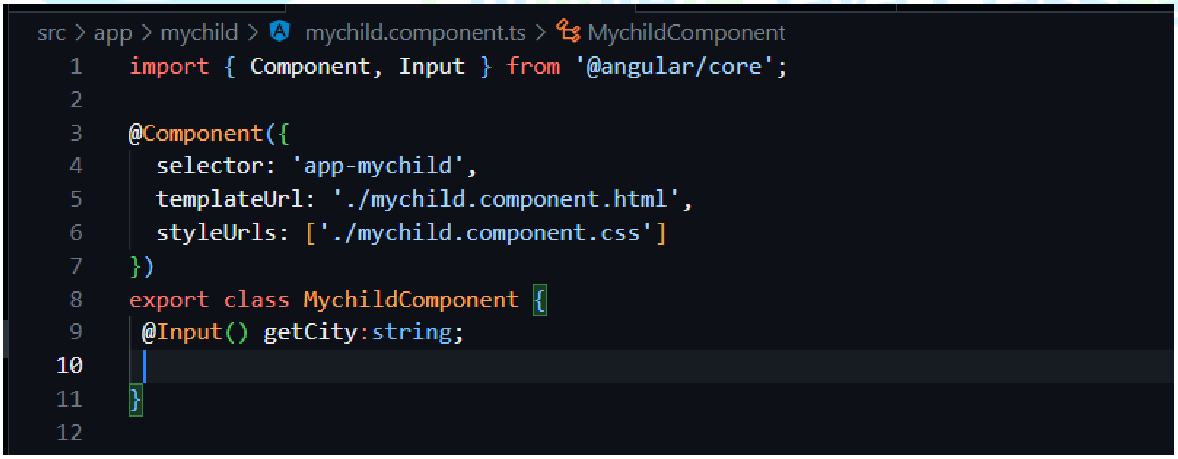
@Input decorator is use to receive the data from parent component to child component.

The property which is decorated with @Input decorator can receive data from parent component. Also any how ,if data get change in parent component ,Angular update the value in child component.

➢ **Below example demonstrate use of @Input decorator.**

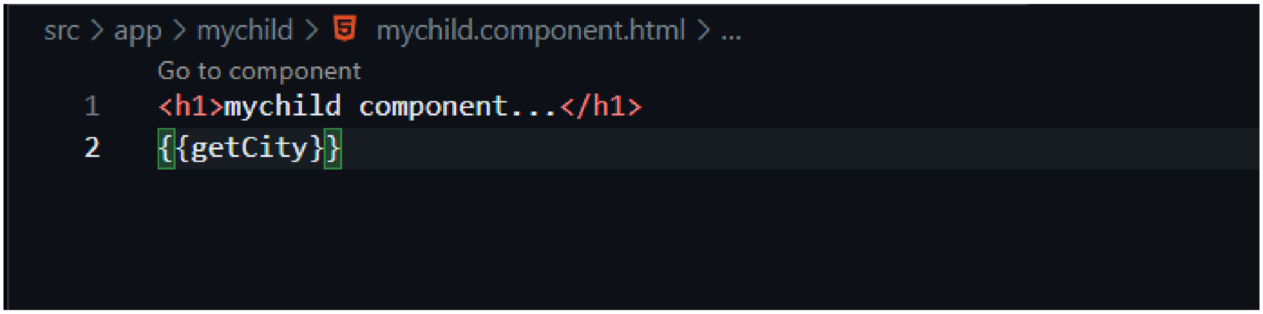
1. Create a custom angular component , in this component decorate one property with @Input decorator.

◆ **Snippet of mychild.component.ts**


src > app > mychild > mychild.component.ts > MychildComponent

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

@Component({
  selector: 'app-mychild',
  templateUrl: './mychild.component.html',
  styleUrls: ['./mychild.component.css']
})
export class MychildComponent {
 @Input() getCity:string;

}

```

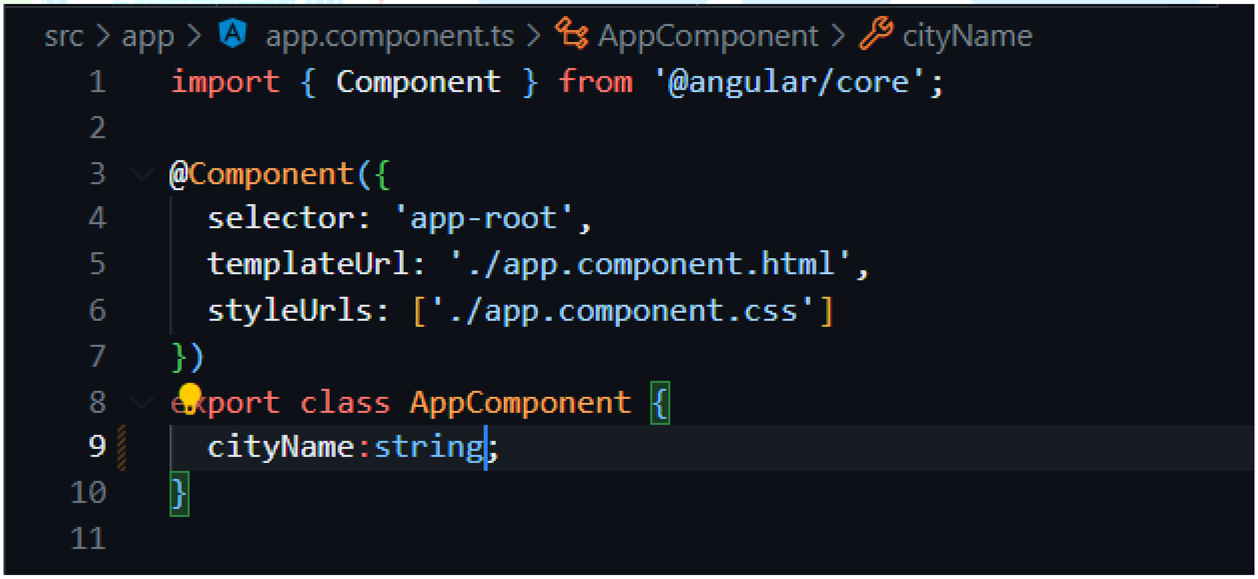
- **Snippet of mychild.component.html**



```html
<h1>mychild component...</h1>
{{getCity}}
```

2. In AppComponet declare a variable to get city name from user . while adding the selector of a MychildComponent bind input property which is declare in MychildComponent , and assign the value of city name.

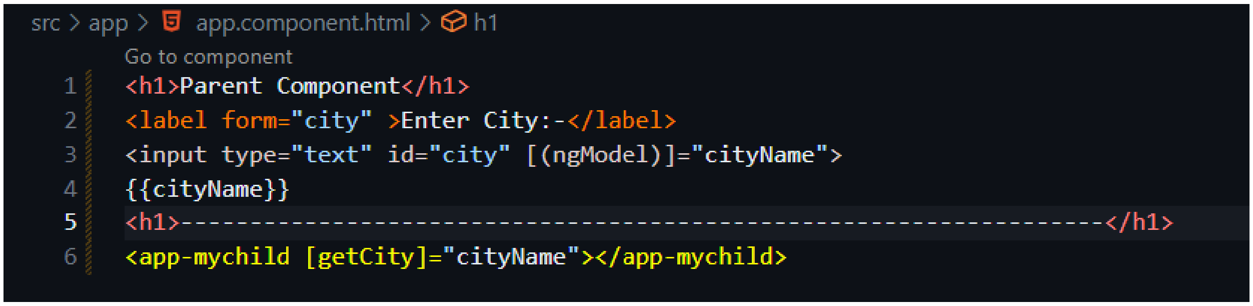
- **Snippet of app.component.ts**



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  cityName:string;
}

```

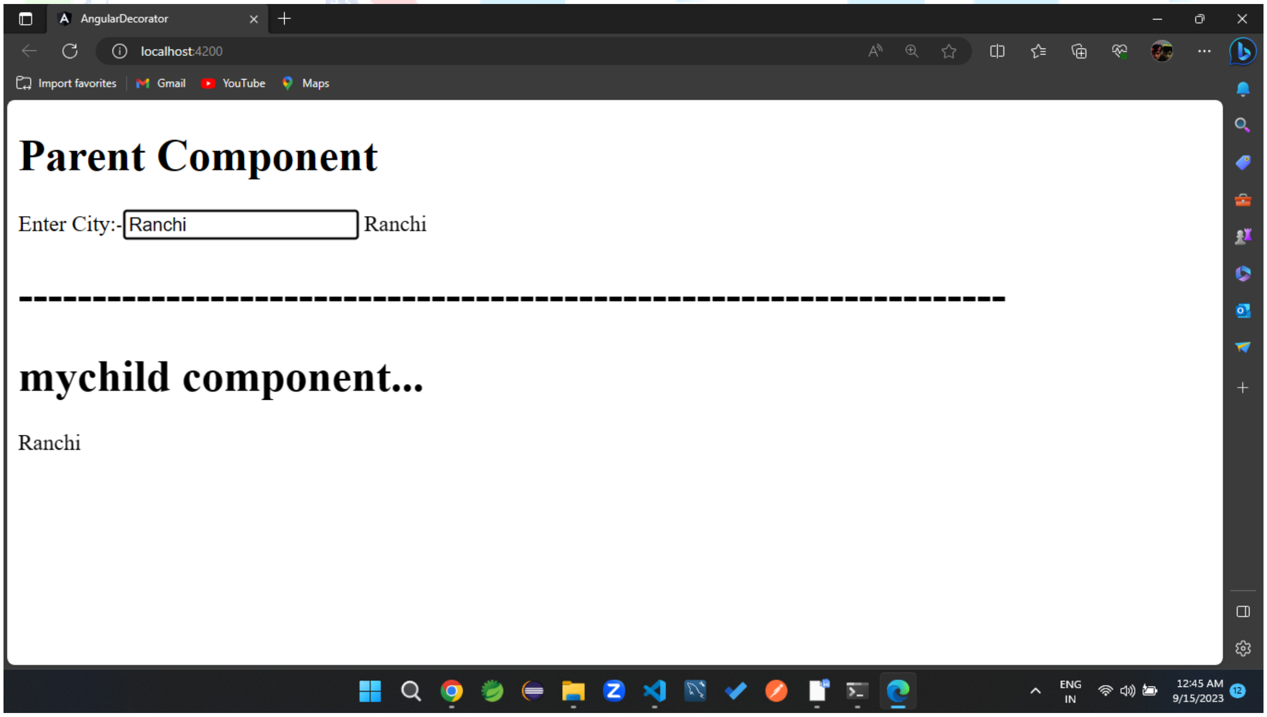
Micro-Services

- **Snippet of app.component.html**



```
<h1>Parent Component</h1>
<label form="city" >Enter City:-</label>
<input type="text" id="city" [(ngModel)]="cityName">
{{cityName}}
<h1>------------------------------------------------------------------</h1>
<app-mychild [getCity]="cityName"></app-mychild>
```

- **Output :-**



Parent Component

Enter City:- Ranchi Ranchi

--

mychild component...

Ranchi

Micro-Services

2. @Output( ) Decorator : -

@Output decorator is use to transfer (send) the data from child component to parent component.

@Output use EventEmitter object to emit data on event.

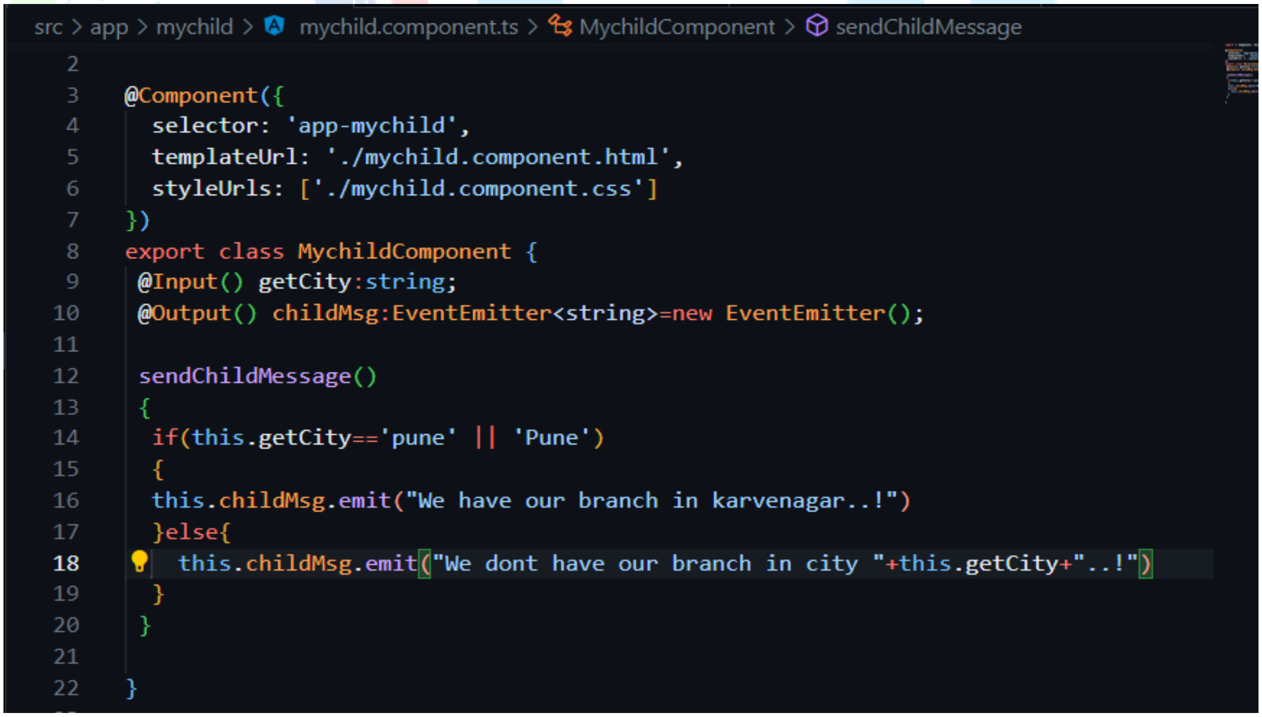
new EventEmitter() :- tells Angular to create a new event emitter and that the data it emits is of type string. The type could be any type, such as number, boolean, and so on.

➢ **Below example will demonstrate use of @Output decorator.**

1. In MychildComponet declare property and initialize it with EventEmitter Object.

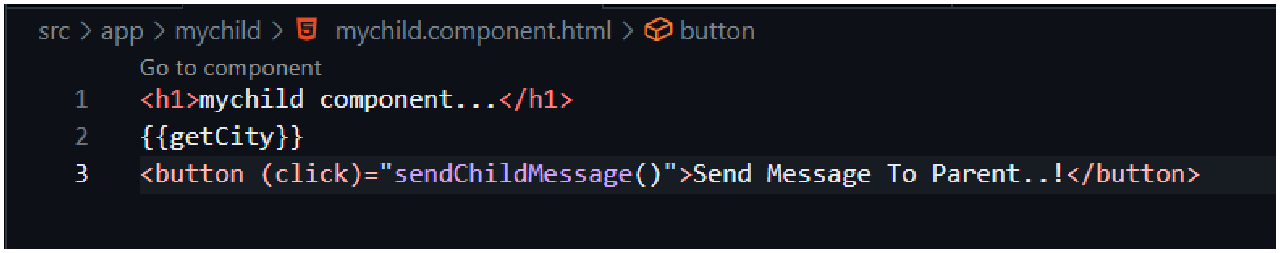
Also declare a function and in function body, emit the data to the parent component , call this function on click event of a button in html of a MychildComponent.

◆ **Snippet of mychild.component.ts**


src > app > mychild > mychild.component.ts > MychildComponent > sendChildMessage

```

@Component({
  selector: 'app-mychild',
  templateUrl: './mychild.component.html',
  styleUrls: ['./mychild.component.css']
})
export class MychildComponent {
 @Input() getCity:string;
 @Output() childMsg:EventEmitter<string>=new EventEmitter();

 sendChildMessage()
 {
  if(this.getCity=='pune' || 'Pune')
  {
  this.childMsg.emit("We have our branch in karvenagar..!")
  }else{
    this.childMsg.emit("We dont have our branch in city "+this.getCity+"..!")
  }
 }

}
```

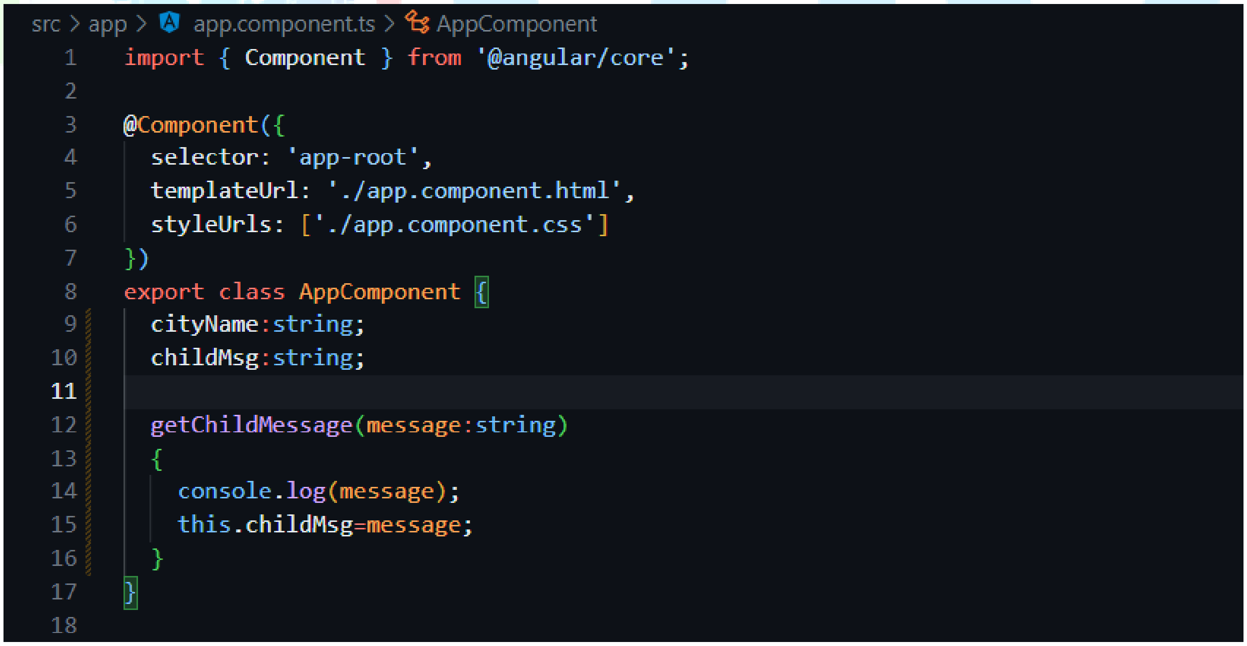
- **Snippet of mychild.component.html**



```html
<h1>mychild component...</h1>
{{getCity}}
<button (click)="sendChildMessage()">Send Message To Parent..!</button>
```

2. **In app component bind the declare function which accepts on argument as a parameter, bind this function on output event on a child component. Also print received data from event on the html of parent component.**

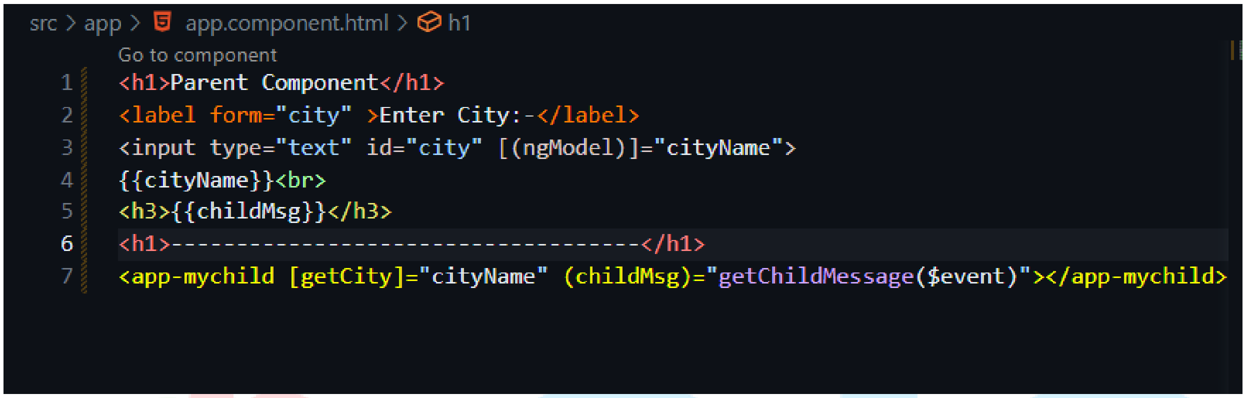
- **Snippet of app.component.ts**



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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  cityName:string;
  childMsg:string;

  getChildMessage(message:string)
  {
    console.log(message);
    this.childMsg=message;
  }
}

```

Micro-Services

- **Snippet of app.component.html**

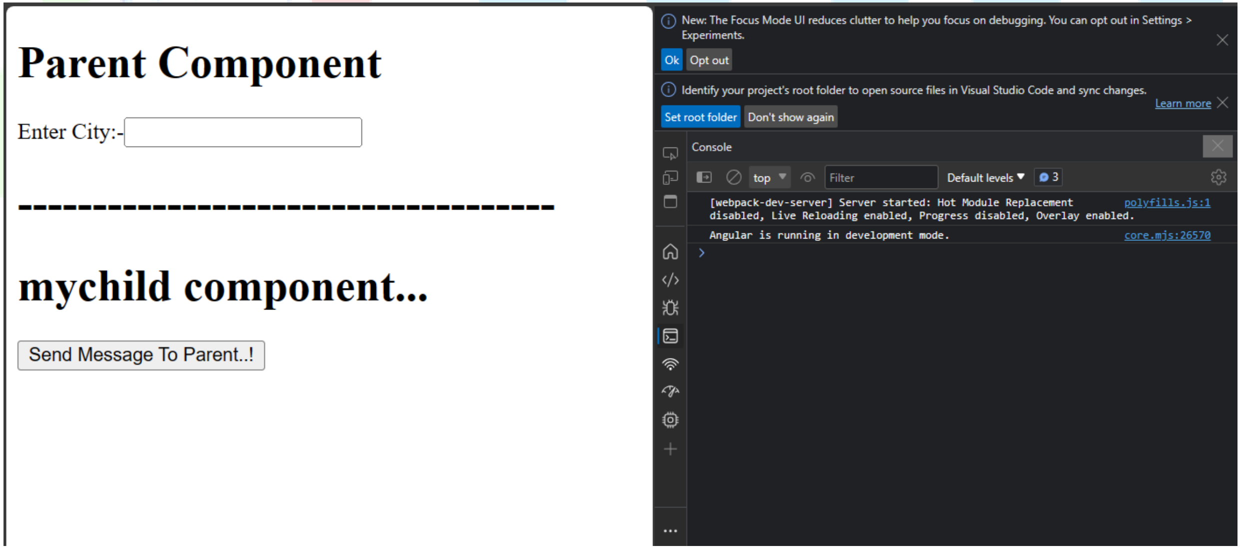


src > app > app.component.html > h1

Go to component

```html
<h1>Parent Component</h1>
<label form="city" >Enter City:-</label>
<input type="text" id="city" [(ngModel)]="cityName">
{{cityName}}<br>
<h3>{{childMsg}}</h3>
<h1>------------------------------------</h1>
<app-mychild [getCity]="cityName" (childMsg)="getChildMessage($event)"></app-mychild>
```

Output :-



Parent Component

Enter City:-

----

mychild component...

Send Message To Parent..!

New: The Focus Mode UI reduces clutter to help you focus on debugging. You can opt out in Settings > Experiments.

Identify your project's root folder to open source files in Visual Studio Code and sync changes.

```
[webpack-dev-server] Server started: Hot Module Replacement        polyfills.js:1
disabled, Live Reloading enabled, Progress disabled, Overlay enabled.
Angular is running in development mode.                            core.mjs:26570
```