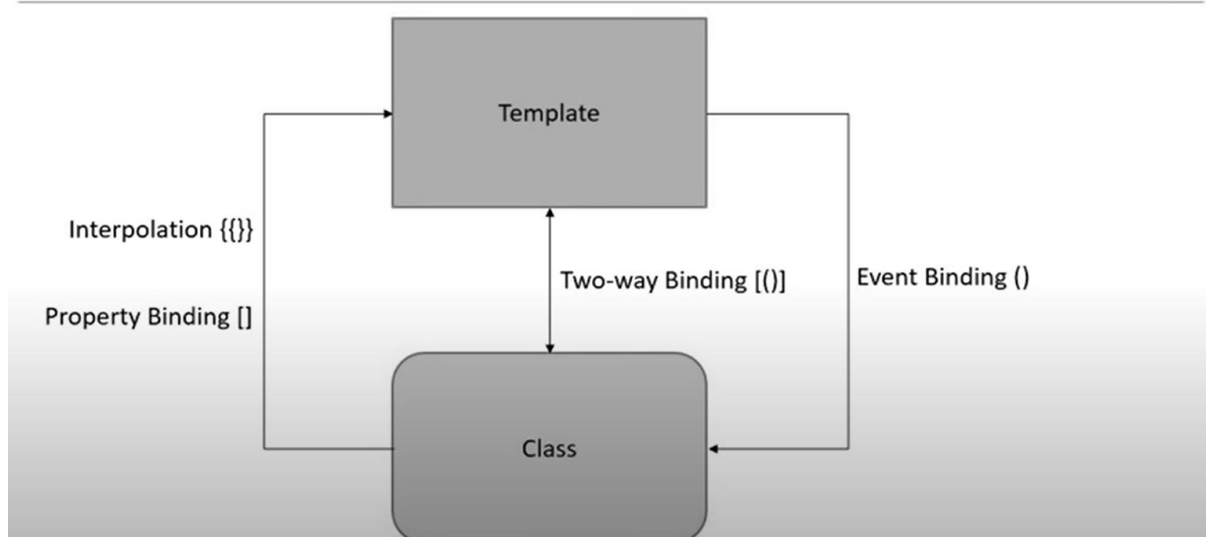


Component Communication:

Communication in Class and Templates.

Binding



As we have already seen component in angular has four files. One is class file, second is template that is html file, third one is css and fourth is testing ts file. So, as component communication we will see how data will be send from class to template and user action from template to class file. In order to perform such pull and push operation angular provides us four forms of binding. Each form has binding to DOM or from DOM or in both directions.

1. **Interpolation (One way Binding)** → We have already seen interpolation which is denoted by `{{}}` double curly braces. This we can say one way binding from component class to template that is HTML.

2. **Property Binding** → Next, we have property binding. Which is represented by square brackets []. This is also type of one-way data binding from component class to properties of template (HTML) elements.
3. **Event Binding** → In opposite to Property binding, in other direction we have event binding which is represented by parenthesis. This is one way binding from HTML template to component class. With Event binding we can execute handlers on users' interactions.
4. **Two-way Binding** → Finally we have two-way binding. Which combines property binding and event binding. Which denotes by [()] parenthesis in square brackets. Which means model and views are always in sync.

Let's start understanding them with examples.

Step 1. Create a new project.

Step 2. Run project with ng serve command

Step 3. Clear all data in app.component.html file and also delete all properties in app.component.ts file.

Step 4. Add a property as titleOfApp='My First Binding Understanding Demo' and using interpolation let's display it in app.component.html file.

App.component.ts

```
<> app.component.html TS app.component.ts X
src > app > TS app.component.ts > ...
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    titleOfApplication='My First Application'
10 }
11 |
```

App.component.html

```
<> app.component.html X TS app.component.ts
src > app > <> app.component.html > h1
1  <h1>{{titleOfApplication}}</h1>
```

So, value of this property will be displayed on our rendered html page.

Which is one-way binding using interpolation from class to template.

Step 5: There is also a one-way binding which is also called as property binding. With which we can bind to property of html element.

For understanding this we will add a new property in class.(Copy paste img folder in asset folder of project)

imgUrl='/assets/img/img1.png'

now let's go back to template and add an img tag

<div>

```
<img [src]="imgUrl" alt="image">
</div>
```

app.component.ts

```
<> app.component.html TS app.component.ts X
src > app > TS app.component.ts > AppComponent > imgUrl
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    titleOfApplication='My First Application';
10   imgUrl='/assets/img/img1.png'
11 }
```

app.component.html

```
<> app.component.html X TS app.component.ts
src > app > <> app.component.html > div
1  <h1>{{titleOfApplication}}</h1>
2  <div>
3    <img [src]="imgUrl" alt="image">
4  </div>
```

Some of you might have doubt that what is difference between interpolation (Data binding) and property binding.

Let's try to understand with example.

Add given below code in **app.component.html**

```
<> app.component.html X TS app.component.ts
test > src > app > <> app.component.html > ...
1 <!-- Difference between interpolation and property binding -->
2 <!-- 1--><input type="text" disabled [value]="placeholderValue">
3 <!-- 4--><input type="text" [disabled]="disableBox" [value]="placeholderValue">
4 <br/>
5 <br/>
6 <!-- 2--><input type="text" disabled value={{placeholderValue}}>
7 <!-- 5--><input type="text" disabled={{disableBox}} value={{placeholderValue}}>
8 <br/>
9 <!-- 3--><button (click)="makeEnable()">Enable Me</button>
10
```

And add given below code in **app.component.ts**

```
<> app.component.html X TS app.component.ts X
test > src > app > TS app.component.ts > AppComponent > makeEnable
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'test';
10  placeholderValue='Please Provide input';
11  disableBox=true;
12
13  makeEnable(){
14    this.disableBox=false;
15    this.placeholderValue=""
16  }
17 }
```

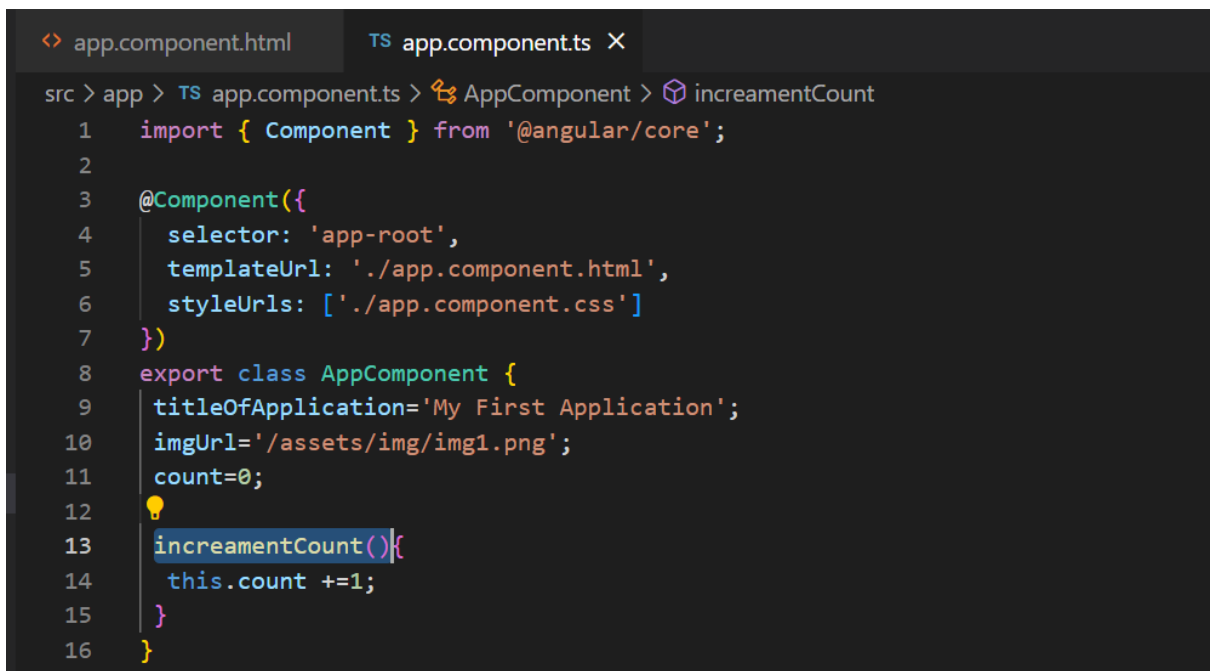
Step 6. One-way binding from template to html that is event binding. Means we will have to handle users' action. When user will perform any event then it should handle using class file. So, it's also one-way direction binding but from template to

class. For this purpose, we will add an element on template and bind event with it.

We will add a button on html page and whenever we will click button, count will increase by 1.

- a. Open app.component.ts file and create new property as count and initialize it with 0.
- b. Then create a function and simply do incrementation in count inside this function and then display this count again to template using interpolation.
- c. Observe given below code:

app.component.ts



```
src > app > TS app.component.ts > AppComponent > increamentCount
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    titleOfApplication='My First Application';
10   imgUrl='/assets/img/img1.png';
11   count=0;
12
13   increamentCount(){
14     this.count +=1;
15   }
16 }
```

app.component.html

```
app.component.html X TS app.component.ts
src > app > app.component.html > h2
1 <h1>{{titleOfApplication}}</h1>
2 <div>
3   <img [src]="imgUrl" alt="image">
4 </div>
5 <button (click)="incrementCount()">Increase Count</button>
6 <h2>You have clicked button {{count}} times</h2>
```

This is also one-way binding from template to class but for handling event that is users' action.

Step 7: Final form is two-way binding. In this binding template passes the data from template to class to update the model and model passes the data back to the template to keep the view in sync.

In order to start working with two-way binding, we first need to include forms module.

- a. So, open app.module.ts file and import forms module.
- b. Also, we have to add it in imports array.

```
src > app > TS app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { FormsModule } from '@angular/forms';
7
8  @NgModule({
9    declarations: [
10     AppComponent
11    ],
12    imports: [
13     BrowserModule,
14     AppRoutingModule,
15     FormsModule
16    ],
17    providers: [],
18    bootstrap: [AppComponent]
19  })
20  export class AppModule { }
21
```

c. Now. Come back to component class and create new property to use for two-way binding.

d. `name :string | undefined`

name :string | undefined

e. Now, open app.component.html and add an input element in template.

<div>

<input type="text" [(ngModel)]="name">

<h2>Welcome {{name}}</h2>

</div>

f. Final code will be as in picture
app.component.ts


```
<> app.component.html TS app.module.ts TS app.component.ts X
src > app > TS app.component.ts > AppComponent > name
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    titleOfApplication='My First Application';
10   imgUrl='/assets/img/img1.png';
11   count=0;
12   name:string | undefined
13
14   increamentCount(){
15     this.count +=1;
16   }
17 }
18
```

app.component.html

```
<> app.component.html X TS app.module.ts TS app.component.ts
src > app > <> app.component.html > div
1  <h1>{{titleOfApplication}}</h1>
2  <div>
3    <img [src]="imgUrl" alt="image">
4  </div>
5  <button (click)="increamentCount()">Increamse Count</button>
6  <h2>You have clicked button {{count}} times</h2>
7  <div>
8    <input type="text" [(ngModel)]="name">
9    <h2>Welcome {{name}}</h2>
10 </div>
```

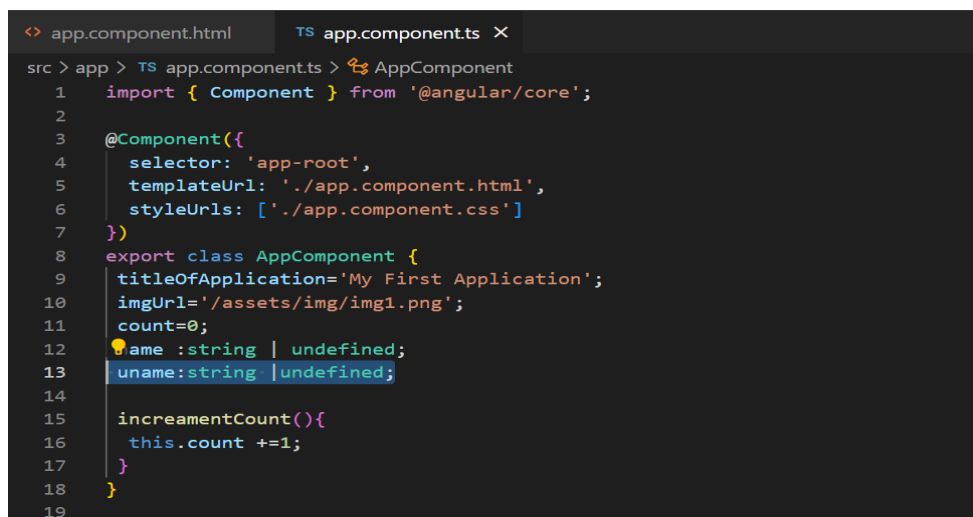
In this way two-way binding can be achieved. So, in the background what is happening? First event binding from view (html) to class property and then updated value passed back to template with property binding. This way class and templates are always in sync with regards to property values. Two-way binding is very important when you want to submit a form or load form with

prefilled data. We will see two-way binding when we will discuss about forms in other session.

Step 8. Here, we can also split two-way binding if sometimes we will have to add some additional functionalities while performing two-way binding. What it means try to understand it with simple example.

a. Add one more property in class file.

uname:string |undefined;



```
<> app.component.html TS app.component.ts X
src > app > TS app.component.ts > AppComponent
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    titleOfApplication='My First Application';
10   imgUrl='/assets/img/img1.png';
11   count=0;
12   ⚠ame :string | undefined;
13   uname:string |undefined;
14
15   incrementCount(){
16     this.count +=1;
17   }
18 }
19
```

b. We will copy same code from previous example and add just below it and do some basic changes as given below:

<div>

<input type="text" [ngModel]="uname"
(ngModelChange)="uname=\$event">

<h2>Welcome {{uname}}</h2>

</div>

```
<> app.component.html X TS app.component.ts
src > app > <> app.component.html > div
1 <h1>{{titleOfApplication}}</h1>
2 <div>
3   <img [src]="imgUrl" alt="image">
4 </div>
5 <button (click)="increamentCount()">Increamse Count</button>
6 <h2>You have clicked button {{count}} times</h2>
7 <div>
8   <input type="text" [(ngModel)]="name">
9   <h2>Welcome {{name}}</h2>
10 </div>
11
12 <div>
13   <input type="text" [ngModel]="uname" (ngModelChange)="uname=$event">
14   <h2>Welcome {{uname}}</h2>
15 </div>
```

Here, **\$event** refers to updated value.
It will work in same fashion.

- c. But now, we have to add some additional functionalities to it so, what we will do is we will add function for ngModelChange.
Follow given below code.

app.component.html

```
<div>
  <input type="text" [ngModel]="uname" (ngModelChange)="myFunction($event)">
  <h2>Welcome {{uname}}</h2>
</div>
```

app.component.ts

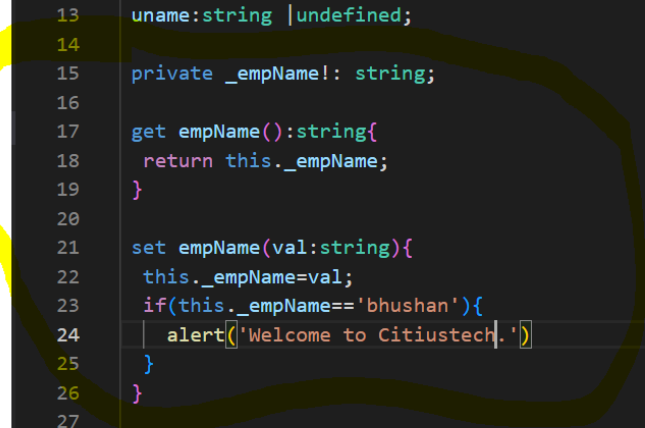
```
<> app.component.html TS app.component.ts X
src > app > TS app.component.ts > AppComponent > myFunction
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    titleOfApplication='My First Application';
10   imgUrl='/assets/img/img1.png';
11   count=0;
12   name :string | undefined;
13   uname:string |undefined;
14
15   increamentCount(){
16     this.count +=1;
17   }
18   myFunction(newVal:string){
19     this.uname=newVal;
20     if(this.uname=='bhushan'){
21       alert('You are authorized person..');
22     }
23   }
24 }
25
```

d. Same thing we can achieve using getter and setters.

app.component.html

```
<> app.component.html X TS app.component.ts
src > app > app.component.html > div > h2
1  <h1>{{titleOfApplication}}</h1>
2  <div>
3    <img [src]="imgUrl" alt="image">
4  </div>
5  <button (click)="increamentCount()">Increamse Count</button>
6  <h2>You have clicked button {{count}} times</h2>
7  <div>
8    <input type="text" [(ngModel)]="name">
9    <h2>Welcome {{name}}</h2>
10 </div>
11
12 <div>
13   <input type="text" [ngModel]="uname" (ngModelChange)="myFunction($event)">
14   <h2>Welcome {{uname}}</h2>
15 </div>
16
17 <div>
18   <input type="text" [(ngModel)]="empName">
19   <h2>Welcome {{empName}}</h2>
20 </div>
```

app.component.ts



```

src > app > TS app.component.ts > AppComponent > (set) empName
6 | styleUrls: ['./app.component.css']
7 | })
8 | export class AppComponent {
9 |   titleOfApplication='My First Application';
10 |  imgUrl='/assets/img/img1.png';
11 |  count=0;
12 |  name :string | undefined;
13 |  uname:string |undefined;
14 |
15 |  private _empName!: string;
16 |
17 |  get empName():string{
18 |    return this._empName;
19 |  }
20 |
21 |  set empName(val:string){
22 |    this._empName=val;
23 |    if(this._empName=='bhushan'){
24 |      alert('Welcome to Citiustech.')]
25 |    }
26 |  }
27 |
28 |  increamentCount(){
29 |    this.count +=1;
30 |  }
31 |  myFunction(newVal:string){
32 |    this.uname=newVal;
33 |    if(this.uname=='bhushan'){
34 |      alert('You are authorized person..')
35 |    }
36 |  }

```

So far, we have seen intra component communication it means just sending data withing components. From class to template and template to class but now we will see inter component communication.

Here we will see,

Data passing in parent and child component with Angular:

@Input, @Output, and EventEmitter

Before we start it, I would like to let know remember that we can also access members of a

class to another class by creating an object that we have already seen.

But for this purpose, we will have to import that component in our component where we have to use members of that components. But angular provided us easiest way to achieve this using concept of Parent-Child component communication.

In order to achieve this, we have to use some decorators

Our task to achieve for is.

- 1.Data exchange from parent to child
- 2.Data exchange from child to parent

@Input Decorator: This is used to define an input property and it is used to send data from parent component to child component.

@Output Decorator: This is used to bind a property of the type of Angular EventEmitter class and it is used to pass data from child component to parent component

EventEmitter: This is used to emit events in components. Use in components with the @Output directive to emit custom events

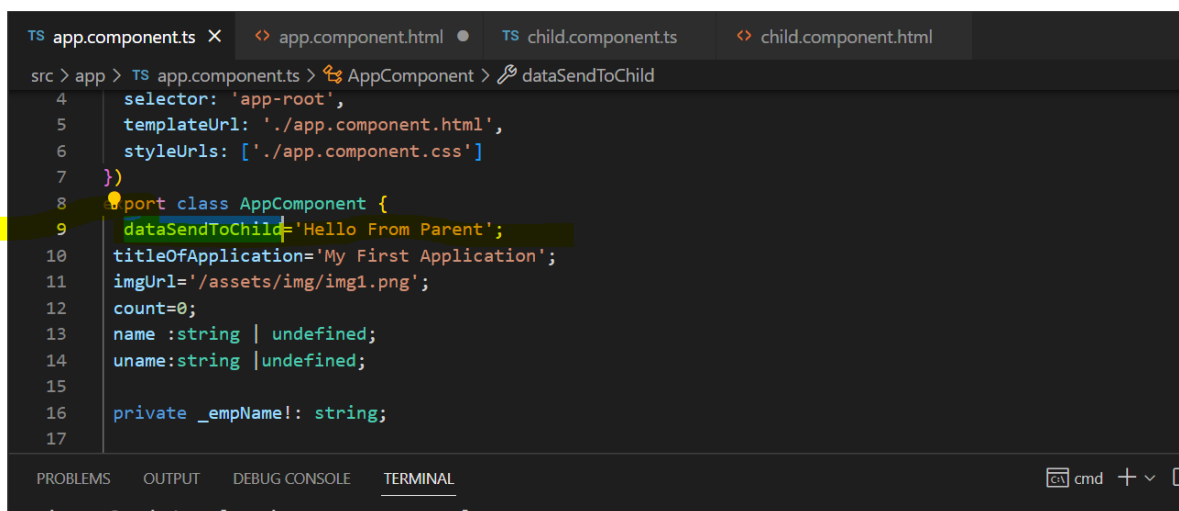
synchronously or asynchronously and register handlers for those events by subscribing to an instance.

Step By Step Implementation of Exchanging data from Parent to Child.

Step 1. Add new component 'Child'. Consider App component is our parent component.

Step 2: Add <app-child> in app.component.html. Check whether it's displaying on app.component.html or not.

Step 3. Open app.component.ts and add one property as dataToSendToChild='Hello From Parent'.



```
TS app.component.ts X  app.component.html  TS child.component.ts  child.component.html
src > app > TS app.component.ts > AppComponent > dataSendToChild
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8
9   port class AppComponent {
10    dataSendToChild='Hello From Parent';
11    titleOfApplication='My First Application';
12    imgUrl='/assets/img/img1.png';
13    count=0;
14    name :string | undefined;
15    unname:string |undefined;
16    private _empName!: string;
17  }
```

Step 4. Open app.component.html again and bind a property.

```
TS app.component.ts    <> app.component.html ●    TS child.component.ts    <> child.com
src > app > <> app.component.html > app-child
17 </div>
18 <input type="text" [(ngModel)]="empName">
19 <h2>Welcome {{empName}}</h2>
20 </div>
21 <app-child [sendDataToChild]="dataSendToChild"></app-child>
```

Step 5. Open child.component.ts file and at top, add input in import. And in class add @Input decorator with name which specified in app.component.html.

```
TS app.component.ts    <> app.component.html ●    TS child.component.ts X    <> child.component.html
src > app > child > TS child.component.ts > ChildComponent > sendDataToChild
1 import { Component, Input } from '@angular/core';
2
3
4 @Component({
5   selector: 'app-child',
6   templateUrl: './child.component.html',
7   styleUrls: ['./child.component.css']
8 })
9 export class ChildComponent {
10   @Input() sendDataToChild: any;
11
12
13
14 }
```

Step 6. Now in child.component.html file add given below code with property which we binded.

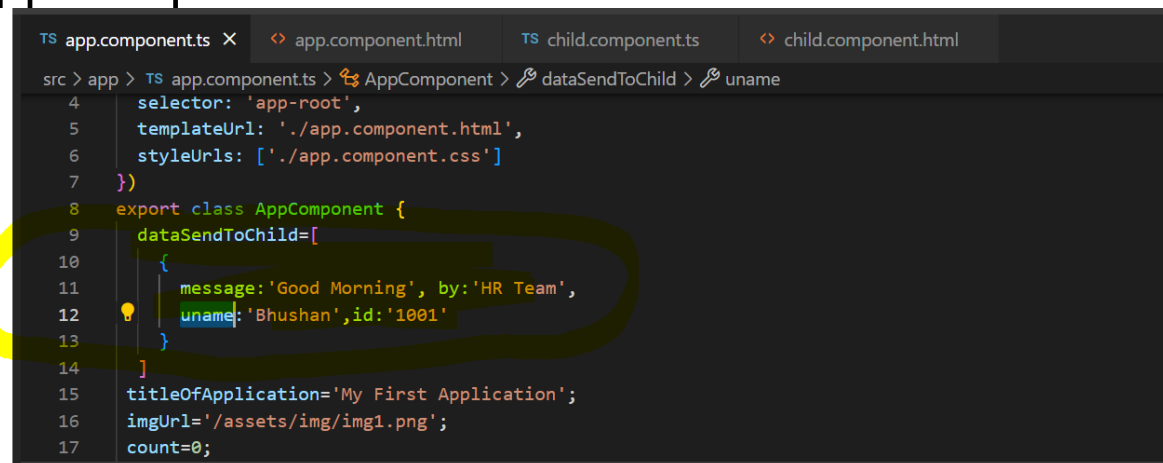
```
TS app.component.ts    <> app.component.html    TS child.component.ts    <> child.component.html X
src > app > child > <> child.component.html > h2
1 <h2>Message Coming from Parent {{sendDataToChild}}</h2>
```


Now in output we can see the data present in parent component.ts file is visible in browser with child.

Same way we can also send multiple values as an object

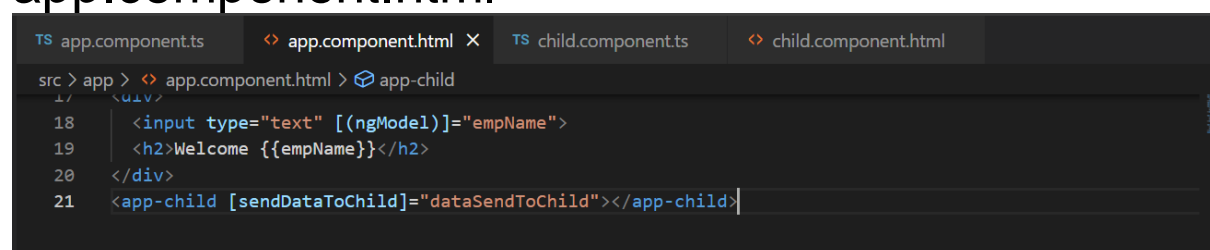
Do given below changes only:

app.component.ts



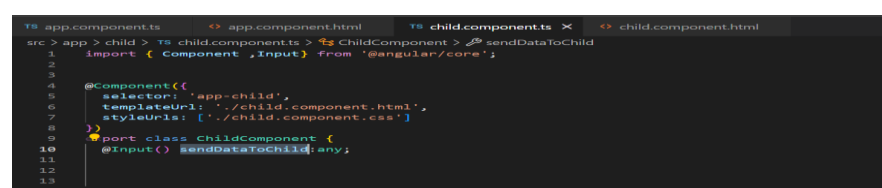
```
TS app.component.ts X app.component.html TS child.component.ts child.component.html
src > app > TS app.component.ts > AppComponent > dataSendToChild > uname
4 selector: 'app-root',
5 templateUrl: './app.component.html',
6 styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   dataSendToChild=[
10     {
11       message: 'Good Morning', by: 'HR Team',
12       uname: 'Bhushan', id: '1001'
13     }
14   ]
15   titleOfApplication='My First Application';
16   imgUrl='/assets/img/img1.png';
17   count=0;
```

app.component.html



```
TS app.component.ts app.component.html X TS child.component.ts child.component.html
src > app > app.component.html > app-child
18 <input type="text" [(ngModel)]="empName">
19 <h2>Welcome {{empName}}</h2>
20 </div>
21 <app-child [sendDataToChild]="dataSendToChild"></app-child>
```

app.component.ts (no need any change)



```
TS app.component.ts app.component.html TS child.component.ts X child.component.html
src > app > child > TS child.component.ts > ChildComponent > sendDataToChild
1 import { Component, Input } from '@angular/core';
2
3
4 @Component({
5   selector: 'app-child',
6   templateUrl: './child.component.html',
7   styleUrls: ['./child.component.css']
8 })
9 export class ChildComponent {
10   @Input() sendDataToChild:any;
11
12
13
14 }
```

App.component.html

```
TS app.component.ts  app.component.html  TS child.component.ts  child.component.html X
src > app > child > child.component.html > h1
1 <h1 *ngFor="let items of sendDataToChild">
2   {{items.message}}--{{items.uname}}
3 </h1>
```

Step By Step Implementation of sending data from Child to Parent.

Step 1. Here we will consider AppComponent as Parent and we will create one more child component by name as newchild.

```
C:\Users\BhushanP5\OneDrive - CitiusTech\Angular Calender Training\Day 5>cd testProject
C:\Users\BhushanP5\OneDrive - CitiusTech\Angular Calender Training\Day 5\testProject>ng g c newc
hild
CREATE src/app/newchild/newchild.component.html (23 bytes)
CREATE src/app/newchild/newchild.component.spec.ts (613 bytes)
CREATE src/app/newchild/newchild.component.ts (210 bytes)
CREATE src/app/newchild/newchild.component.css (0 bytes)
UPDATE src/app/app.module.ts (624 bytes)
```

Step 2. Open newchild.component.ts file and add two more components in import as given in picture.

```
TS newchild.component.ts X
testProject > src > app > newchild > TS newchild.component.ts > NewchildComponent
1 import { Component, EventEmitter, Output } from '@angular/core';
2
```

Step 4. Now, time to use @Output decorator and create object of EventEmmitter. For this purpose add a given below code.

```

TS newchild.component.ts X
testProject > src > app > newchild > TS newchild.component.ts > NewchildComponent
1  import { Component, EventEmitter, Output } from '@angular/core';
2
3  @Component({
4    selector: 'app-newchild',
5    templateUrl: './newchild.component.html',
6    styleUrls: ['./newchild.component.css']
7  })
8  export class NewchildComponent {
9    @Output() parentFunction:EventEmitter<any> = new EventEmitter();
10

```

Step 3. Let's consider a scenario, on button click event we will have to send data from Child to Parent and display.

For this purpose, we will create a function as in given picture.

```

TS newchild.component.ts X
testProject > src > app > newchild > TS newchild.component.ts > NewchildComponent
1  import { Component, EventEmitter, Output } from '@angular/core';
2
3  @Component({
4    selector: 'app-newchild',
5    templateUrl: './newchild.component.html',
6    styleUrls: ['./newchild.component.css']
7  })
8  export class NewchildComponent {
9    @Output() parentFunction:EventEmitter<any> = new EventEmitter();
10
11    sendDataToParent(){
12      let data={ename:'Bhushan',age:35}
13      this.parentFunction.emit(data);
14    }
15  }
16

```

Step 4. Open app.component.html and add selector of newchild component

```

<> app.component.html X
testProject > src > app > <> app.component.html > h2
1  <h1>This is App Component</h1>
2  <app-child [sendDataToChild]="dataToSendToChild"></app-child>
3  <app-newchild (parentFunction)="parentFunction($event)"></app-newchild>

```

Step 5. Now, Open newchild.component.html file and add button element to fire event.

```
<> newchild.component.html X
testProject > src > app > newchild > <> newchild.component.html > button
1 <h2>This is Child Component</h2>
2 <button (click)="sendDataToParent()">Send data To Parent</button>
3
```

Step 6. Add given below code in app.component.ts

```
<> app.component.html TS app.component.ts X
testProject > src > app > TS app.component.ts > AppComponent > parentFunction
10 title = 'testProject';
11 dataToSendToChild = [{
12   message: 'This Is Our Test Angular Application',
13   info: 'For Testing Purpose Only'
14 }]
15
16 iname = ''
17 iage = 0
18
19 parentFunction(myData: any) {
20   this.iname = myData.ename;
21   this.iage = myData.age;
22 }
23
24 }
25
```

Step 7. To display data of Child to Parent, we can use interpolation in app.component.html

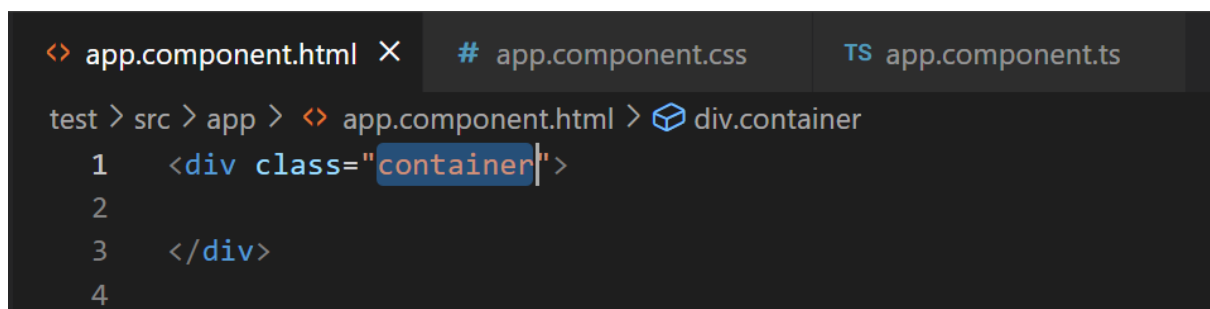
```
<> app.component.html X
testProject > src > app > <> app.component.html > h2
1 <h1>This is App Component</h1>
2 <app-child [sendDataToChild]="dataToSendToChild"></app-child>
3 <app-newchild (parentFunction)="parentFunction($event)"></app-newchild>
4 <h1>{{iname}}</h1>
5 <h2>{{iage}}</h2>
```

Component Interaction using Service.

Component Interaction using Service is very easy.

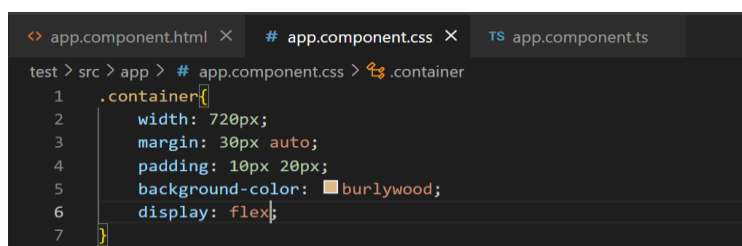
For understanding this we will create two components. EmployeeInfo and DisplayEmployeeDetails. These two components are completely different. In EmployeeInfo component we will display List of Employee with one button for each employee to see their details and when we click on this button it should display Employee details in DisplayEmployeeDetails component. Let's start implementing this requirement.

Step 1. Create a div in app.component.html and provide class selector container as shown in given below picture.



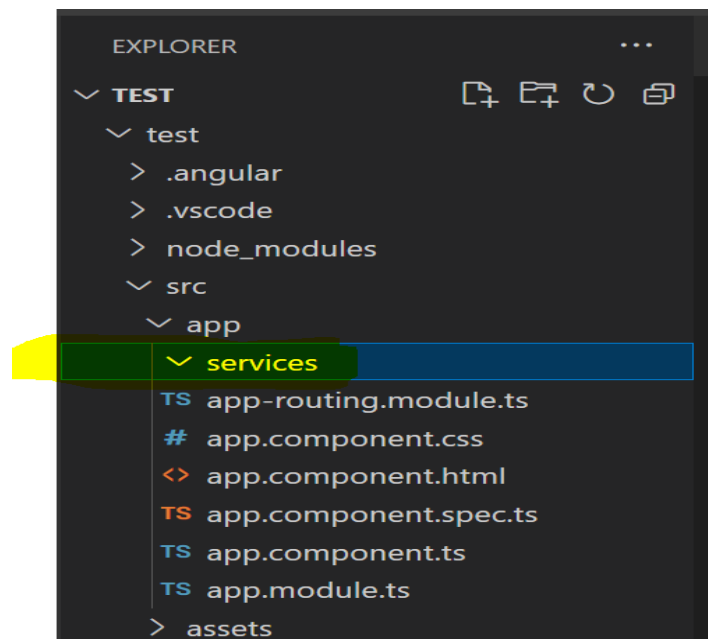
```
<> app.component.html X # app.component.css TS app.component.ts
test > src > app > <> app.component.html > div.container
1  <div class="container">
2
3  </div>
4
```

Step 2. Now in app.component.css file add some styling for this container.

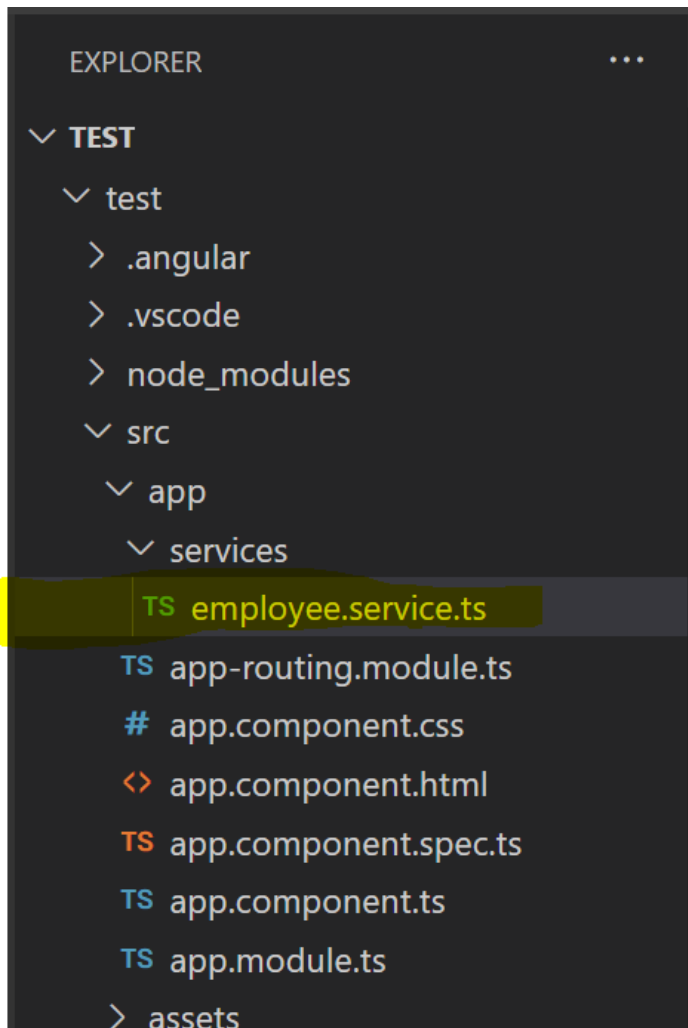


```
<> app.component.html X # app.component.css X TS app.component.ts
test > src > app > # app.component.css > .container
1  .container{
2    width: 720px;
3    margin: 30px auto;
4    padding: 10px 20px;
5    background-color: burlywood;
6    display: flex;
7  }
```

Step 3. Create a new folder as services in app folder.



Step 4. Inside this folder create new file as **employee.service.ts** file.



Step 5. Open employee.service.ts file and add given below code in it. We will create an array of employees in this file.

```
test > src > app > services > TS employee.service.ts > employee > employees > email
1  export class employee{
2      employees =[
3          {ename:'John',dept:'Delivery', gender:'male',age:35,location:'Mumbai',email:'john@gmail.com'},
4          {ename:'James',dept:'Production', gender:'male',age:42,location:'Pune',email:'james@gmail.com'},
5          {ename:'Tina',dept:'HR', gender:'female',age:30,location:'Mumbai',email:'tina@gmail.com'},
6          {ename:'Kevin',dept:'IT', gender:'male',age:35,location:'Chennai',email:'kevin@gmail.com'}
7      ]
8  }
9  }
```

Step 6: Now, let's inject this service in app component. At this point just try to understand that inject means we will have to get this service here

in app component so, all it's nested component can access this service. We will discuss dependency injection in depth when we'll see Services.

For injecting service in app component we have to follow given below steps.

a. Import employee at top

```
import { employee } from './services/employee.service';
```

b. Create constructor of app.component.ts class to inject it.

```
constructor(private employees:employee){  
}
```

c. We will also have to specify providers too in @Component decorator.

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css'],  
  providers:[employee]  
})
```

Our final app.component.ts file will look like:


```
TS employee.service.ts    TS app.component.ts X
test > src > app > TS app.component.ts > AppComponent
1  import { Component } from '@angular/core';
2  import { employee } from '../services/employee.service';
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css'],
7    providers: [employee]
8  })
9  export class AppComponent {
10    title = 'test';
11    constructor(private employees: employee){
12
13    }
14
15  }
16
```

So, here we have injected employees service in app component and now, it can be use with child components of app components.

Step 7. Now, we will have to add new component employeeinfo.

And we have to get values of employees array from employeeservice.ts in employeeinfo.componen.ts file.

Final code of **employeeinfo.componen.ts** will be like :

Employeeinfo.component.ts

```
TS employeeinfo.component.ts X
test > src > app > employeeinfo > TS employeeinfo.component.ts > EmployeeinfoComponent > ngOnInit
1 //step 2 add OnInit
2 import { Component, OnInit } from '@angular/core';
3 // step1
4 import { employee } from '../services/employee.service';
5
6 @Component({
7   selector: 'app-employeeinfo',
8   templateUrl: './employeeinfo.component.html',
9   styleUrls: ['./employeeinfo.component.css']
10 })
11 //step 3 implements OnInit
12 export class EmployeeinfoComponent implements OnInit {
13
14   //step 5 add constructor to inject
15   constructor(private employees:employee){ }
16   //step 6 create property which will hold values in employees from employee service
17   employeesinfo:{ename: string,dept:string, gender:string,age:number,location:string,email:string}[][]=[];
18
19   //Step 4 implement this method.
20   ngOnInit(): void {
21     //assign value of employee object to employeesinfo
22     this.employeesinfo=this.employees.employees;
23   }
24
25 }
26
```

Step 8. Now, open app.component.html file and add selector of employeesinfo component.
app.component.html file will look like:

```
TS employeeinfo.component.ts employeeinfo.component.html app.component.html X
test > src > app > app.component.html > div.container > div > app-employeeinfo
1 <div class="container">
2 <div>
3   <h3>List Of Employees</h3>
4   <app-employeeinfo></app-employeeinfo>
5 </div>
6 </div>
7
```

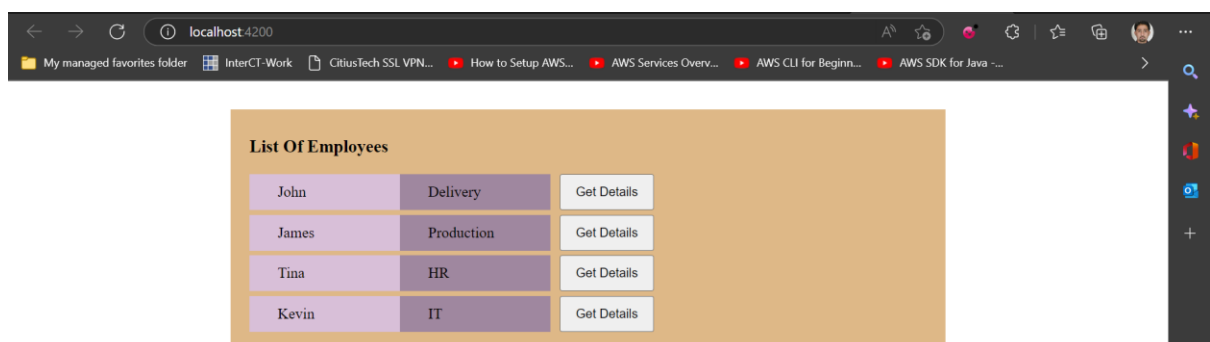
Step 9. Open employeeinfo.component.html file and add given below code.

```
TS employeeinfo.component.ts employeeinfo.component.html X employeeinfo.component.css app.component.html
test > src > app > employeeinfo > employeeinfo.component.html > div.container > div.emp-container > div.emp-dept
1 <div class="container" *ngFor="let emp of employeesinfo">
2   <div class="emp-container">
3     <div class="emp-name">{{emp.ename}}</div>
4     <div class="emp-dept">{{emp.dept}}</div>
5     <button>Get Details</button>
6   </div>
7
8 </div>
```

Also add css in employeeinfo.component.css file.

```
TS employeeinfo.component.ts  <> employeeinfo.component.html  # employeeinfo.component.css X  <>
test > src > app > employeeinfo > # employeeinfo.component.css > button
1  .emp-container{
2      display: flex;
3      margin: 5px 0px;
4  }
5  .emp-name{
6      width: 100px;
7      padding: 10px 30px;
8      background-color: thistle;
9  }
10 .emp-dept{
11     width: 100px;
12     padding: 10px 30px;
13     background-color: rgb(159, 135, 159);
14 }
15 button{
16     width: 100px;
17     margin: 0px 10px;
18 }
```

Till this point output of our application will look like:



Now, our target is to get details of Employee on click event of button in another component.

Step 10. We will create new component as DisplayEmployeeDetails to display all details of employee.

And open display-employee-details.component.html file first. Code in html file will look like.

```
<> display-employee-details.component.html X # display-employee-details.component.css
test > src > app > display-employee-details > <> display-employee-details.component.html > div.container > div.user-details
1 <div class="container">
2 <div class="user-details">
3   <p><b>Employee Name:</b></p>
4   <p><b>Employee Dept:</b></p>
5   <p><b>Employee Gender:</b></p>
6   <p><b>Employee Location:</b></p>
7   <p><b>Employee Email:</b></p>
8 </div>
9 </div>
```

Add css in **display-employee-details.component.css**.

```
<> display-employee-details.component.html # display-employee-details.component.css X
test > src > app > display-employee-details > # display-employee-details.component.css > .user
1 .container{
2   margin: 20px 0px;
3   padding: 20px 20px;
4   display: flex;
5 }
6 .user-details{
7   padding: 20px 20px;
8 }
```

Step 11. Now, open **employeeinfo.component.html** file and for button we will add (click) event.

```
<> display-employee-details.component.html # display-employee-details.component.css <> employeeinfo.component.html X
test > src > app > employeeinfo > <> employeeinfo.component.html > div.container > div.emp-container > button
1 <div class="container" *ngFor="let emp of employeesinfo">
2   <div class="emp-container">
3     <div class="emp-name">{{emp.ename}}</div>
4     <div class="emp-dept">{{emp.dept}}</div>
5     <button (click)="showDetails(emp)">Get Details</button>
6   </div>
7
8 </div>
9
```

Step 12. Open **employeeinfo.component.ts** file and define **showDetails()** function.

```
display-employee-details.component.html  employeeinfo.component.html  TS employeeinfo.component.ts X
test > src > app > employeeinfo > TS employeeinfo.component.ts > EmployeeinfoComponent > showDetails
13
14 //step 5 add constructor to inject
15 constructor(private employees:employee){ }
16 //step 6 create property which will hold values in employees from employee service
17 employeesinfo:{ename: string,dept:string, gender:string,age:number,location:string,email:string}[];
18
19 //Step 4 implement this method.
20 ngOnInit(): void {
21 //assign value of employee object to employeesinfo
22 this.employeesinfo=this.employees.employees;
23 }
24
25 showDetails(emp:{ename: string,dept:string, gender:string,age:number,location:string,email:string})
26 |
27 }
28
29 }
30
```

But what we are supposed to do in this method.

Step 13. Open our service.ts file and use EventEmitter to emit data.

For this first we will have to import EventEmitter in employees.service.ts file. And then create object to emit.

```
display-employee-details.component.html X  employeeinfo.component.html  TS employeeinfo.component.ts  TS employee.service.ts X
test > src > app > services > TS employees.service.ts > employee
1 import {EventEmitter} from '@angular/core'
2 export class employee{
3   employees=[
4     {ename:'John',dept:'Delivery', gender:'male',age:35,location:'Mumbai',email:'john@gmail.com'},
5     {ename:'James',dept:'Production', gender:'male',age:42,location:'Pune',email:'james@gmail.com'},
6     {ename:'Tina',dept:'HR', gender:'female',age:30,location:'Mumbai',email:'tina@gmail.com'},
7     {ename:'Kevin',dept:'IT', gender:'male',age:35,location:'Chennai',email:'kevin@gmail.com'}
8   ]
9
10   OnDisplayDetailsClicked = new EventEmitter<{ename: string,dept:string, gender:string,age:number,location:string,email:string}>();
11   showEmpDetails(emp:{ename: string,dept:string, gender:string,age:number,location:string,email:string}){
12     this.OnDisplayDetailsClicked.emit(emp);
13   }
14 }
```

Now, we will have to call this method in showDetails() method in employeeinfo.component.ts file.

Step 14. Open employeeinfo.component.ts file and add given below code.

```
display-employee-details.component.html  employeeinfo.component.html  TS employeeinfo.component.ts  TS employee.service.ts
test > src > app > employeeinfo > TS employeeinfo.component.ts > EmployeeinfoComponent > showDetails
11 //step 3 implements OnInit
12 export class EmployeeinfoComponent implements OnInit {
13
14     //step 5 add constructor to inject
15     constructor(private employees:employee){ }
16     //step 6 create property which will hold values in employees from employee service
17     employeesinfo:{ename: string,dept:string, gender:string,age:number,location:string,email:string}[][]=[];
18
19     //Step 4 implement this method.
20     ngOnInit(): void {
21         //assign value of employee object to employeesinfo
22         this.employeesinfo=this.employees.employees;
23     }
24
25     showDetails(emp:{ename: string,dept:string, gender:string,age:number,location:string,email:string}){
26         this.employees.showEmpDetails(emp);
27     }
28
29 }
30
```

So, whenever we clicked on this button this method will be called and from this showDetails method, showEmpDetails() method in service will be called. When showEmpDetails() will be called , it will raise an event that is OnDisplayDetailClick event and when event will raise it will emit data.

Our emitted data will have to inject in display-employee-details.component.ts file.

Step 15: Add given below code in **display-employee-details.component.ts** file

```
app.component.html  TS display-employee-details.component.ts
test > src > app > display-employee-details > TS display-employee-details.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { employee } from '../services/employee.service';
3
4 @Component({
5     selector: 'app-display-employee-details',
6     templateUrl: './display-employee-details.component.html',
7     styleUrls: ['./display-employee-details.component.css']
8 })
9 export class DisplayEmployeeDetailsComponent implements OnInit{
10     constructor(private employees:employee){
11         emp!: { ename: string; dept: string; gender: string; age: number; location: string; email: string; };
12     }
13     ngOnInit(){
14         this.employees.OnDisplayDetailsClicked.subscribe((data:{ ename: string; dept: string; gender: string; age: number; location: string; email: string; })=>{
15             this.emp=data;
16         })
17     }
18 }
19
```

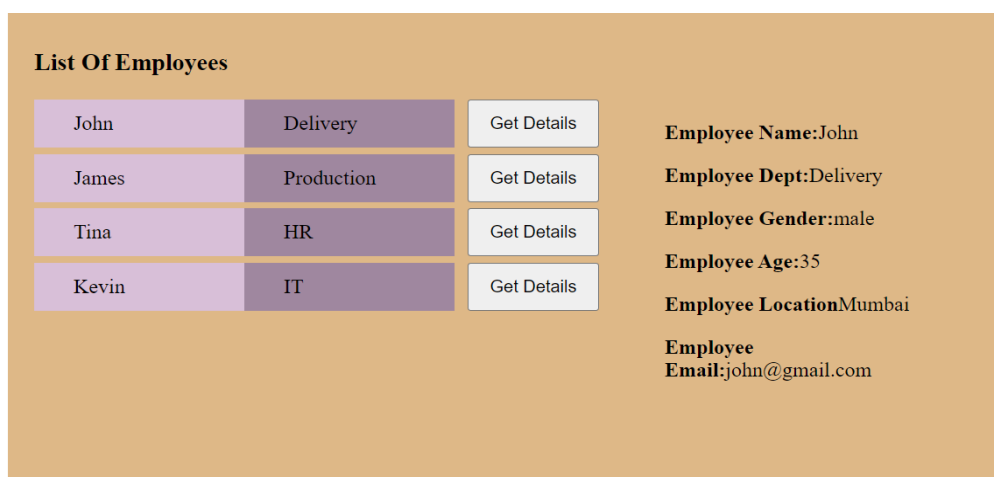
Step 16. Open display-employee-details.component.html file and add given below code.

```
app.component.html  display-employee-details.component.html X
test > src > app > display-employee-details > display-employee-details.component.html > div.container
1  <div class="container" *ngIf="emp != undefined">
2  <div class="user-details">
3      <p><b>Employee Name:</b>{{emp.ename}}</p>
4      <p><b>Employee Dept:</b>{{emp.dept}}</p>
5      <p><b>Employee Gender:</b>{{emp.gender}}</p>
6      <p><b>Employee Age:</b>{{emp.age}}</p>
7      <p><b>Employee Location</b>{{emp.location}}</p>
8      <p><b>Employee Email:</b>{{emp.email}}</p>
9  </div>
10 </div>
```

Step 17. Open app.component.html file and add div to add selector of display-employee-details component.

```
app.component.html X  display-employee-details.component.html
test > src > app > app.component.html > div.container > div
1  <div class="container">
2  <div>
3      <h3>List Of Employees</h3>
4      <app-employeeinfo></app-employeeinfo>
5  </div>
6  <div>
7      <app-display-employee-details></app-display-employee-details>
8  </div>
9  </div>
10
```

After all above steps we will get below output.



Directives:

[Directives in Angular Applications - YouTube](#)

<https://www.youtube.com/watch?v=LtT01ZCHRjk>

Forms:

[Building Forms in Angular Apps | Mosh - YouTube](#)

[https://www.youtube.com/watch?v=hAaoPOx_olw
&list=RDCMUUCWv7vMbMWH4-
V0ZXdmDpPBA&start_radio=1&rv=hAaoPOx_olw
&t=294](https://www.youtube.com/watch?v=hAaoPOx_olw&list=RDCMUUCWv7vMbMWH4-V0ZXdmDpPBA&start_radio=1&rv=hAaoPOx_olw&t=294)

Full Course and pick required

[Introduction to Angular | Angular Basics | Angular 12+ - YouTube](#)

[https://www.youtube.com/watch?v=NMzI2pGOK_8
&list=PL1BztTYDF-QNrtkvjkT6Wjc8es7QB4Gty](https://www.youtube.com/watch?v=NMzI2pGOK_8&list=PL1BztTYDF-QNrtkvjkT6Wjc8es7QB4Gty)