**1.Component Communication**:- The angular components can communicate with each other. The component can also share data between them. There are two possible scenario.

1.1.parent-to-child communication.

1.2.child-to-parent communication.

**1.1.Parent-to-child communication**:-If two components have parent and child relationship then parent component can send the data to child component using ‘@input() ‘ decorator.

Follow below steps in child component.

* Import the @Input module from @angular/Core Library
* Mark those property, which you need data from the parent as input property using’ @Input decorator’.

Q) what is @input?

A) It is decorator. When you mark component property as @input property,then angular injects value into component property using property binding.

syntax:

@input()propertyname;

Follow the below steps in parent component.

1. Declare property in parent component. That property value has to be passed to child component.
2. Open parent component template/view( .html file). Write the following statement in child component tag.

Syntax:

<childcomponent-name [child-component –propery]=parent-component-property>….</childcomponent-name>

Diagram representation:-1

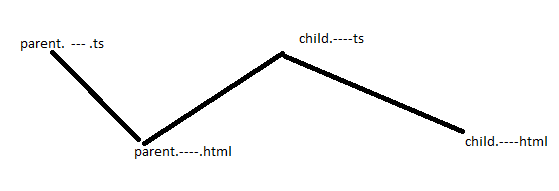
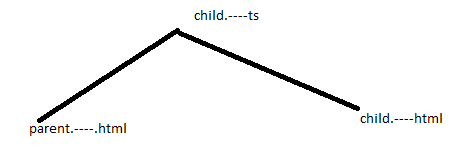


Diagram representation:-2



Example:

1.app.component.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  sData:Student=new Student('suku',41,[1,2,3],{'street':'kamati','drno':16,'town':'nlr'});

  sData1:Student=new Student('suku1',42,[4,5,6],{'street':'kamati','drno':16,'town':'nlr'});

  sData2:Student[]=[this.sData,this.sData1];

}

class Student{

  public name:string;

  public age:number;

  public marks:number[];

  public address:object;

  public tdate:Date=new Date;

  constructor(a:string,b:number,c:number[],d:object)

  {

    this.name=a;

    this.age=b;

    this.marks=c;

    this.address=d;

  }

}

2.app.component.html

<app-second-app [one]='sData2'>

</app-second-app>

3.second-app-component.cs

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

@Component({

  selector: 'app-second-app',

  templateUrl: './second-app.component.html',

  styleUrls: ['./second-app.component.css']

})

export class SecondAppComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  @Input() one:any;

  xy:object={'background-color':'red','width':'50vw','position':'relative','left':'25vw'};

  ab:string='color:blue';

}

4.second-app-component.html

<table border="1" [ngStyle]='xy'>

    <caption [style]='ab'>STUDENT TABLE</caption>

    <tr>

        <th>name</th>

        <th>Age</th>

        <th>Marks</th>

        <th>Street</th>

        <th>DoorNo</th>

        <th>Town</th>

    </tr>

    <tr \*ngFor="let x of one">

        <td>{{x.name}}</td>

        <td>{{x.age}}</td>

        <td>{{x.marks}}</td>

        <td>{{x.address.street}}</td>

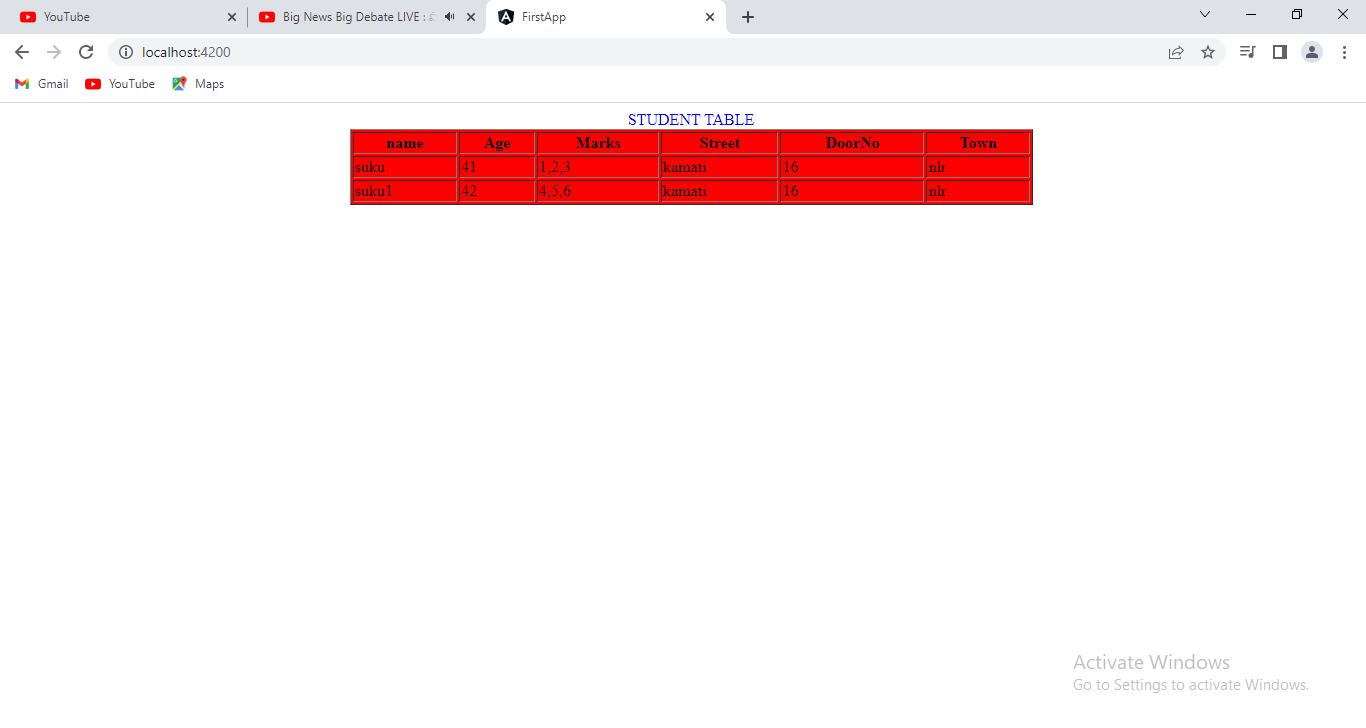
        <td>{{x.address.drno}}</td>

        <td>{{x.address.town}}</td>

    </tr>

</table>

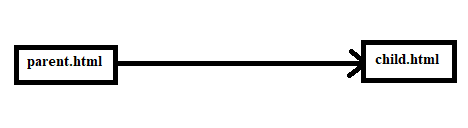
Output:



ParentTemplate to childTemplate Communication: we use @input directive to pass data from parent to child. It is only limited to data. We can’t use @input to pass content which include html element, css code to child Template from parent template.

For this purpose, we should use ng-content directive. You use the <ng-content></ng-content> tag as a placeholder for that dynamic content, then when the template is parsed Angular will replace that placeholder tag with content from parent component. This data passing is said to be “projecting content”.

Diagramatic Representation:



Example:

**1.app.component.html(parent)**

<app-second>

    Hai How r u?

</app-second>

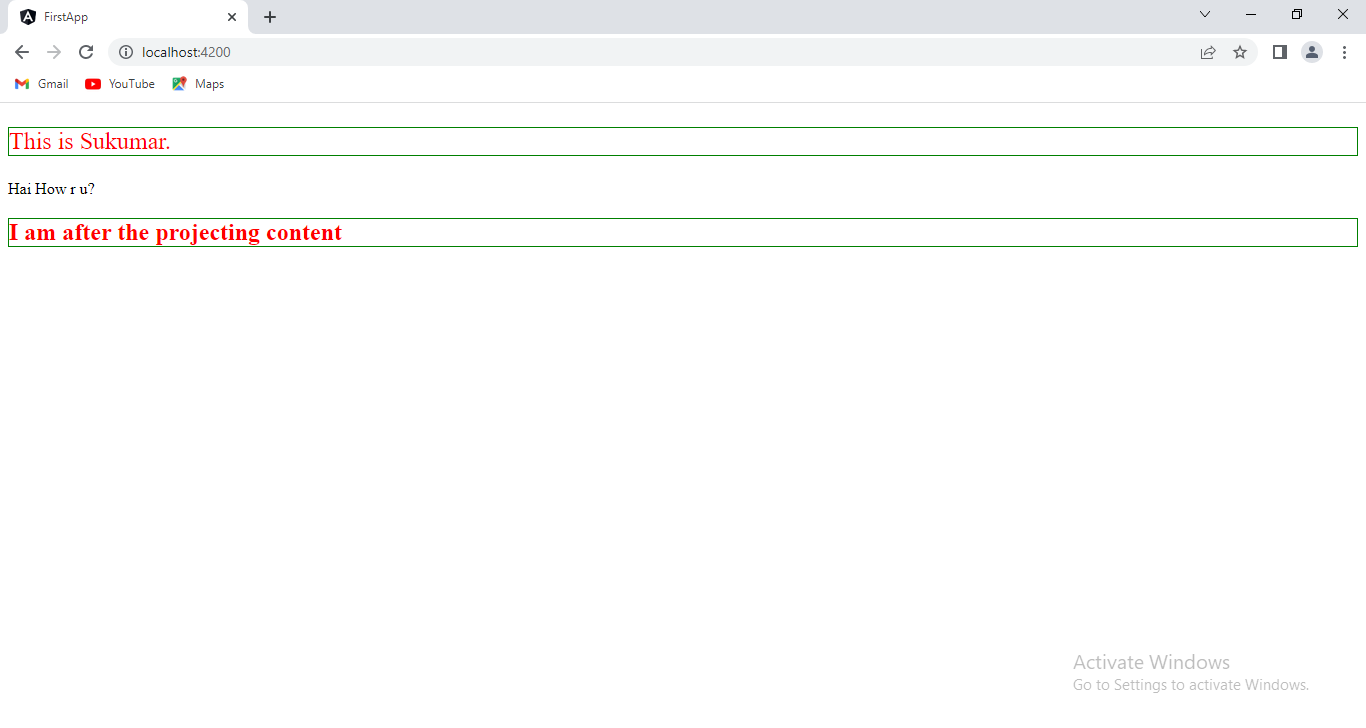
2.second.component.html(child)

<p [style]='x'>This is Sukumar.</p>

<ng-content ></ng-content>

<h2 [style]='x'>I am after the projecting content</h2>

Output:



**1.2.Child-to-Parent Communication:** The Child to Parent communication can happen in three ways.

a. Listens to Events from Child.

b. Uses Local Variable to access the child in the Template.

c. Uses a @ViewChild to get a reference to the child component.

**1.2.1.Listens to Events from child**:- When Child Component needs to communicate with the parent it raises the event. The Parent Component listens to that event and reacts to it.

Follow the below steps in child component

* + - Import EventEmitter module.
    - Declare the property which will stores the reference of EventEmitter object.

Q) what is EventEmitter?

A)To raise an event , the component must declare Eventemitter.

* + - to make that event accessible from parent component, you must decorate the property with @Output decorator.

Syntax:

@Output() var-name:EventEmitter<Datatype>=new EventEmitter();

Raise the event passing it with desired data.

Propertyname.emit(input property);

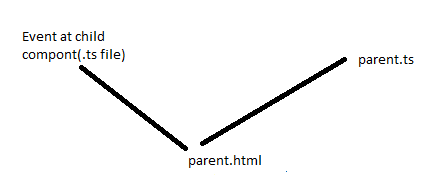
Follow The below steps in Parent Component:

1. Open parent.ts file and define event handler.
2. Open parent.html file. Write the following statement to bind event which happens in client component with event handler which is defined in parent component.

<child-component selector (child-@outpt propertyname)=”event handler name”>…..</child-component selector>

Diagram representation

---------------------------------



**1.2.2 .Using Template Reference Variable:-** In this way, Programs does not need Input , Ouptut and EventEmitter modules.The Template Reference variable is created, when you use #<varibaleName> and attach it to a DOM element in parent template.

Syntax:

<child-component selector #variablename></child-component selector>

#variablename is template reference variable. Using that , we can access properties and methods in parent template.

Example:

1.sec-com-app.ts

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

@Component({

  selector: 'app-second-app',

  templateUrl: './second-app.component.html',

  styleUrls: ['./second-app.component.css']

})

export class SecondAppComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  sData:Student=new Student('suku',41,[1,2,3],{'street':'kamati','drno':16,'town':'nlr'});

  sData1:Student=new Student('suku1',42,[4,5,6],{'street':'kamati','drno':16,'town':'nlr'});

  sData2:Student[]=[this.sData,this.sData1];

}

class Student{

  public name:string;

  public age:number;

  public marks:number[];

  public address:any;

  public tdate:Date=new Date;

  constructor(a:string,b:number,c:number[],d:object)

  {

    this.name=a;

    this.age=b;

    this.marks=c;

    this.address=d;

  }

}

2.app-com-app.html

<app-second-app #sec>

</app-second-app>

<table border="1" [ngStyle]='xy'>

    <caption [style]='ab'>STUDENT TABLE</caption>

    <tr>

        <th>name</th>

        <th>Age</th>

        <th>Marks</th>

        <th>Street</th>

        <th>DoorNo</th>

        <th>Town</th>

    </tr>

    <tr \*ngFor="let x of sec.sData2">

        <td>{{x.name}}</td>

        <td>{{x.age}}</td>

        <td>{{x.marks}}</td>

        <td>{{x.address.street}}</td>

        <td>{{x.address.drno}}</td>

        <td>{{x.address.town}}</td>

    </tr>

</table>

3.app-com-app.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

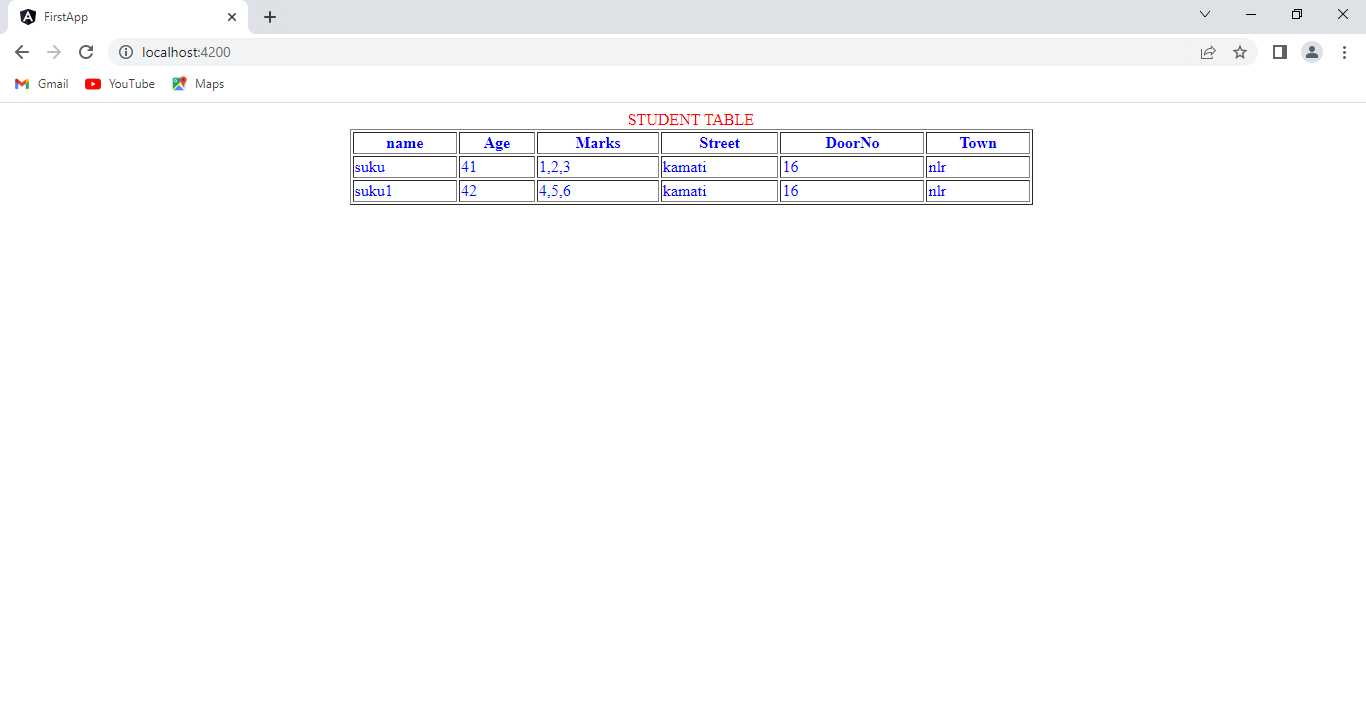
export class AppComponent {

  xy:object={'color':'blue','position':'relative','left':'25vw','width':'50vw'};

  ab:string='color:red';

}

output



**1.2.3. View Child**:-

**2.Parent Template to Child template Communication:-**