# **Component Interaction**

* Component Interaction is nothing communication done between 2 or more components/ passing data between 2 or more components.
* This can be done in 4 ways.

## Parent to Child: Sharing Data Via **Input**

* Need to create a variable containing a message from parent in the parent components ts file.

**parent.component.ts**

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

@Component({

  selector: 'app-parent',

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

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

})

export class ParentComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  parentMessage = "Message from Parent";

}

* Keep Child reference in the parent components HTML file.

**parent.component.html**

<app-child></app-child>

* Create a variable in the child component ts file with **@Input()** decorator and need to import the decorator.

**child.component.ts**

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

@Component({

  selector: 'app-child',

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

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

})

export class ChildComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  @Input() childMessage: string | undefined;

}

* Now we have to use the input decorator variable in child component ts file as property and bind it to the variable in parent component ts file using property binding in the HTML of parent component.

**parent.component.html**

<app-child [childMessage]="parentMessage"></app-child>

* Now print the childMessage in the child component html

**child.component.html**

<h2>{{childMessage}}</h2>

## **Child to Parent: Sharing Data Via ViewChild**

* Create a variable in the child component ts file.

**child.component.ts**

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

@Component({

  selector: 'app-child',

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

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

})

export class ChildComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  message = 'Hola Parent!';

}

* Now inject viewChild with child component reference in the parent component ts file.

**parent.component.ts**

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

import { ChildComponent } from '../child/child.component';

@Component({

  selector: 'app-parent',

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

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

})

export class ParentComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  @ViewChild(ChildComponent) child: any;

}

* Now create a button with corresponding click event and assign the value of variable containing child message from child component to the variable present in the parent component.

**parent.component.html**

<div>

    <p>Child Message: {{ childMessage }}</p>

    <button class="btn btn-secondary" (click)="seeChildMessage()">Print Child Message</button>

</div>

<app-child1></app-child1>

**parent.component.ts**

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

import { Child1Component } from '../child1/child1.component';

@Component({

  selector: 'app-parent1',

  templateUrl: './parent1.component.html',

  styleUrls: ['./parent1.component.css']

})

export class Parent1Component implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  @ViewChild(Child1Component) child: Child1Component | undefined;

  childMessage: any = null;

  seeChildMessage() {

    if (this.child) {

      this.childMessage = this.child.message;

    }

  }

}

## **Child to Parent: Sharing Data Via Output() and EventEmitter**

* Since we are using event emitter we need to communicate between parent and child using event i.e if we want to pass data when ever event is triggered like button click then we have to follow this method.
* So now we will create a button with corresponding click event in the child component html.

**child.component.html**

<button class="btn btn-secondary" (click)="sendMessage()">Send Message</button>

* In the child, we declare a messageEvent variable with the Output decorator and set it equal to a new event emitter. Then we create a function named sendMessage that calls emit on this event with the message we want to send. Lastly, we create a button to trigger this function.

**child.component.ts**

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

@Component({

  selector: 'app-child2',

  templateUrl: './child2.component.html',

  styleUrls: ['./child2.component.css']

})

export class Child2Component implements OnInit {

  message: string = "Hello Parent!"

  @Output() messageEvent = new EventEmitter<string>();

  constructor() { }

  ngOnInit() {

  }

  sendMessage() {

    this.messageEvent.emit(this.message)

  }

}

* In parent component html we will be keeping the child reference using the child components selector value as a tag and in the tag we will be using the event emitter reference present in child as a event in the parent and we will assign a function to it which will be passing the $event as argument.
* In parent ts file we will be creating the function and assign the event argument which was passed in the html and assign it to a variable and view it in the html.

**parent.component.html**

Message: {{message}} <br>

<app-child2 (messageEvent)="receiveMessage($event)"></app-child2>

**child.component.html**

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

@Component({

  selector: 'app-parent2',

  templateUrl: './parent2.component.html',

  styleUrls: ['./parent2.component.css']

})

export class Parent2Component implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

  message:string | undefined;

  receiveMessage($event: string) {

    this.message = $event

  }

}

## **Unrelated components: Sharing Data Via Service**