**Component Interaction:**

1. **With @Input Property**
2. **Getter and Setter with Input**
3. **On NgOnChanges**
4. **With @Output**
5. **Using Local Variable**
6. **Using ViewChild**
7. **Using Services**
8. **@Input:**

Parent Component:

<child-component [Childvariable] = ‘ParentVariable’></ child-component>

ParentVariable = ‘Some Random Value’

Child Component:

@Input() Childvariable!: Interface OR @Input() Childvariable = ‘ ’

*Note:*

1. *Import ‘Input’ from ‘@angular/core’*
2. **Getter and Setter with Input**

Parent Component:

<child-component [Childvariable] = ‘ParentVariable’></ child-component>

Child Component:

Private priChildVar;

@Input()

Get Childvariable (): string

{return this. PriChildVar;}

Set Childvariable (someVar: string)

{this. PriChildVar = <Some functions on someVar >}

1. **NgOnChanges:**

Child Component():

NgOnChanges(changes: SimpleChanges)

{ for (const propName in changes)

{const changedProp = changes[propName];

For (changedProp.currentValue/changedProp.previousValue/changedProp.isFirstChanged)

{Print the propName or Do something with propName}

}

}

1. **@Output**

Parent Component:

<child-component (Childvariable) = ‘ParentFunction($event)’></ child-component>

ParentFunction(valuefromchild){}

Child Component:

@Output ChildVariable = new EventEmitter<Type>()

Some function(value)

{

This. ChildVariable.emit(value);

}

*Note:*

1. *Import ‘Output’,’EventEmitter’ from ‘@angular/core’*
2. **Local Variable**

Parent Component:

<html>

<element localvariable.childfunction()></element>

<element localvariable.childmethod()></element>

<element>{{localvariable.childVariable}}</element>

</html>

<child-component #localvariable = ‘ParentFunction($event)’></ child-component>

Child Component:

1. **View Child**

Parent Component:

@ViewChild(ChildComponent)

Private parentVariable!: ChildComponent;

This. parentVariable.childMethod();

This. parentVariable.childMethod();

This. parentVariable.childVariable;

Child Component:

1. **Using Services**

Service:

Private parentToChild = new Subject<Type>();

Private childToparent = new Subject<Type>();

Private childConsumes = this. parentToChild.asObservables();

Private parentConsumes = this. childToparent.asObservables();

ParentEmits(somevar: type){

This.parentToChild.next(somevar);

}

ChildEmits(randomvar: type){

This.childToparent.next(randomvar);

}

Parent Component:

Import {Service} from ‘path’

[providers]: Service

Constructor(private service: Service)

This.service.parentConsumes.subscribe(val => print ‘val’);

SomeMethod() {

This.service.parentEmits(somevalue);

}

Child Component:

Import {Service} from ‘path’

Import {Subscription} from ‘rxjs’

Constructor(private service: Service) {

subscription: Subscription;

This.subscription = service. ChildConsumes.subscribe(val => print ‘val’)}

SomeMethodInChild(SomeVal){

This.service.ChildEmits(SomeVal)}

NgOnDestroy() {

This.subscription.unsubscribe;}