|  |
| --- |
|  |
| Angular 8 |
|  |
| Angular – Component Interaction |

**TechBrain Express**

December 16, 2019

Authored by: Ashu

Angular – Component Interaction

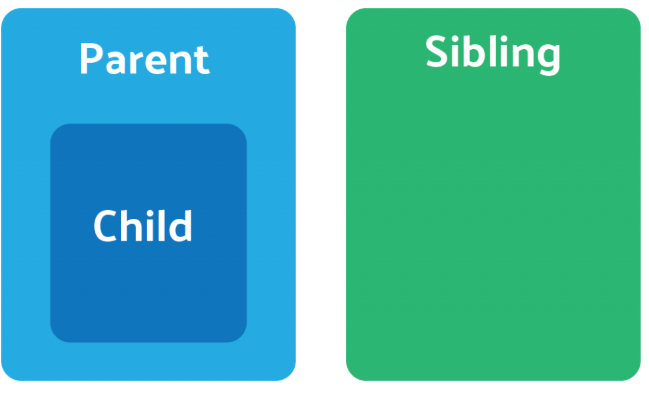
* **Demonstrate implementation of interaction between components**

Follow: F:\AngularOct19\ revatureBookStoreAddBookLogin-CTSOBL

# What is Component Interaction?

It contains recipes for common component communication scenarios in which two or more components share information.

Data sharing is an essential concept to understand before diving into your first Angular project



# Four Ways :

### 1 .Parent to Child: Sharing Data via Input

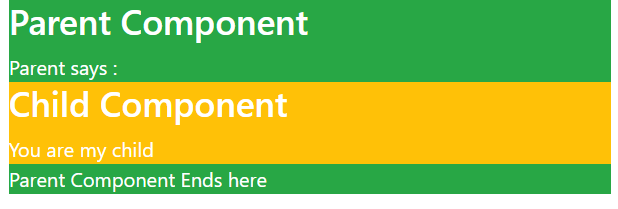
## 2. Child to Parent: Sharing Data via ViewChild

## 3. Child to Parent: Sharing Data via Output() and EventEmitter

## 4. Unrelated Components: Sharing Data with a Service

### Parent to Child: Sharing Data via Input

##### Lab: Create parent component and child component and paas a message from say” You are child “ to child component.



1. Create parent component > ng g c ParentComponent
2. Create Child component > ng g c ChildComponent
3. Create a Variable parent message in parent component

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

@Component({

selector: 'app-parent-component',

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

template:`<div class="col-sm-4">

<div class="bg-success text-white">

<h3>Parent Component</h3> <div> Parent Component Ends here </div> </div>

</div>`,

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

})

export class ParentComponentComponent implements OnInit {

parentMessage=" You are my child"

constructor() { }

ngOnInit() {

}

}

1. Declare a message variable to receive the message from parent as INPUT and print.

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

@Component({

selector: 'app-child-component',

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

template: `

<div class="bg-warning text-white">

<h3>Child Component</h3> {{message}}

</div>`,

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

})

export class ChildComponentComponent implements OnInit {

@Input() message : string;

bgColor;

constructor() { this.bgColor = 'BBFFFF';}

ngOnInit() {

}

}

1. Map the child “message” variable with “parentMessage” in parnent component

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

@Component({

selector: 'app-parent-component',

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

template:`<div class="col-sm-4">

<div class="bg-success text-white">

<h3>Parent Component</h3>

Parent says :

<div>

<app-child-component [message]="parentMessage"> </app-child-component>

</div>

<div> Parent Component Ends here </div>

</div>

</div>`,

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

})

export class ParentComponentComponent implements OnInit {

parentMessage=" You are my child"

constructor() { }

ngOnInit() {

}

}

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

To share data is to emit data from the child, which can be listed to by the parent. This approach is ideal when you want to share data changes that occur on things like button clicks, form entires, and other user events.

In the parent, we create a function to receive the message and set it equal to the message variable.

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.

The parent can now subscribe to this messageEvent that’s outputted by the child component, then run the receive message function whenever this event occurs.

Lab: On button click in child component send a message to parent

1. Create a button in child component and define a method sendMessage() to emit the response.

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

@Component({

selector: 'app-child-component',

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

template: `

<!--Case 1: 1.  Child to Parent: Sharing Data via Output() and EventEmitter-->

<div class="bg-warning text-white">

<h3>Child Component</h3>

{{message}}

</div>

<!---Case 2 : Child to Parent: Sharing Data via Output() and EventEmitter -->

<button (click)="sendMessage()">Send Message</button> `,

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

})

export class ChildComponentComponent implements OnInit {

//Case 1: Child to Parent: Sharing Data via Output() and EventEmitter-->

@Input() message : string;

bgColor;

constructor() { this.bgColor = 'BBFFFF';}

ngOnInit() { }

//Case 2: Child to Parent: Sharing Data via Output() and EventEmitter

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

childReply: string = "Yes Mom I am your kidoooo!"

sendMessage() {

this.childMessageEvent.emit(this.childReply) //

}

}

1. Receive a message from parent to child to show the response.

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

@Component({

selector: 'app-parent-component',

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

template:`<div class="col-sm-4">

<div class="bg-success text-white">

<h3>Parent Component</h3>

Parent says :

<div>

<app-child-component (childMessageEvent)="receiveMessage($event)"

[message]="parentMessage">

</app-child-component>

</div>

<div> Message from child: {{getMessage}}

<br>Parent Component Ends here </div>

</div>

</div>`,

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

})

export class ParentComponentComponent implements OnInit {

parentMessage=" You are my child"

constructor() { }

ngOnInit() { }

//Case 2: Child to Parent: Sharing Data via Output() and EventEmitter

getMessage:string;

receiveMessage($event) {

this.getMessage = $event

}

}

## Child to Parent: Sharing Data via ViewChild

[**ViewChild**](https://angular.io/api/core/ViewChild) allows a one component to be injected into another, giving the parent access to its attributes and functions. One caveat, however, is that child won’t be available until after the view has been initialized. This means we need to implement the AfterViewInit lifecycle hook to receive the data from the child.

Refer: <https://blog.angular-university.io/angular-viewchild/>

ViewChild is a decorator that creates a view or DOM query

Refrence: <https://dev.to/angular/the-angular-viewchild-decorator-424c>

<https://www.concretepage.com/angular-2/angular-2-viewchild-example>

 @ViewChild() decorator configures a view query. @ViewChild() decorator can be used to get the first element or the directive matching the selector from the view DOM. @ViewChild() provides the instance of another component or directive in a parent component and then parent component can access the methods and properties of that component or directive. In this way using @ViewChild() a components can communicate with a component or directive. In a parent component we can use @ViewChild() for components, directives and template reference variable with ElementRef or TemplateRef. To use @ViewChild() we need to pass child component name or directive name or template variable as an argument.

1. Create a myviewparentcomponent and my view Child component

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

import { MyviewchildComponent } from '../myviewchild/myviewchild.component';

@Component({

selector: 'app-myview-parent',

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

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

})

export class MyviewParentComponent implements AfterViewInit{

@ViewChild ( MyviewchildComponent) child: MyviewchildComponent;

constructor() { }

ngAfterViewInit() {

console.log('Values on ngAfterViewInit():');

console.log("childMessage:", this.child.viewMessage);

}

}

1. Declare message variable in childcomponent.

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

@Component({

selector: 'app-myviewchild',

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

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

})

export class MyviewchildComponent {

constructor() { }

viewMessage=" I am a child born with @ViewChild";

}

## 4. Unrelated Components: Sharing Data with a Service