Angular Pass data from Child to parent component

23 Comments / 7 minutes of reading / May 7, 2021

←
Passing data to child Component

Life Cycle Hooks →

In this tutorial, we will learn how to Pass data from child to Parent Component in Angular. In the previous tutorial, we looked at how the pass data from parent to the child component by setting its input property. The Child can send data to Parent by raising an event, Parent can interact with the child via local variable or Parent can call @ViewChild on the child. We will look at all those options in this article.

Table of Contents

Pass data from Child to parent component

There are three ways in which the parent component can interact with the child component

1. Listens to Child Event
2. Uses Local Variable to access the child
3. Uses a @ViewChild to get the reference to the child component

Let us look at each of those scenarios in detail

Parent listens for child event

The Child Component exposes an EventEmitter Property. This Property is adorned with the @Output decorator. When Child Component needs to communicate with the parent it raises the event. The Parent Component listens to that event and reacts to it.

EventEmitter Property

To Raise an event, the component must declare an EventEmmitter Property. The Event can be emitted by calling the .emit() method

For Example

```

countChanged: EventEmitter<number> = new EventEmitter()

```

And then call emit method passing the whatever the data you want to send as shown below

```

this.countChanged.emit(this.count);

```

@Output Decorator

Using the EventEmitter Property gives the components ability to raise an event. But to make that event accessible from parent component, you must decorate the property with @Output decorator

How to Pass data to parent component using @Output

In the child component

1. Declare a property of type EventEmitter and instantiate it
2. Mark it with a @Output Decorator
3. Raise the event passing it with the desired data

In the Parent Component

1. Bind to the Child Component using Event Binding and listen to the child events
2. Define the event handler function

Passing data to parent component Via Events (Example)

Now let us build an application to demonstrate this

In the last passing data to child component tutorial, we built a counter in the parent component. We assigned the initial value to the counter and added increment/decrement methods. In the child Component, we used the @Input decorator to bind count property to the parent component. Whenever parent count is changed in the parent the child component is updated and displayed the count.

In this tutorial, we will move the counter to the child component. We will raise an event in the child component whenever the count is increased or decreased. We then bind to that event in the parent component and display the count in the parent component.

Download the source code for this from the GitHub from the folder inputdecorator. The The final code is available in outputdecorator folder.

Child Component

Open the child.component.ts and copy the following code

```

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

@Component({
   selector: 'child-component',
   template: `<h2>Child Component</h2>
            <button (click)="increment()">Increment</button>
            <button (click)="decrement()">decrement</button>
            current count is {{ count }}
   `
})
export class ChildComponent {
   @Input() count: number;

   @Output() countChanged: EventEmitter<number> =   new EventEmitter();

   increment() {
      this.count++;
      this.countChanged.emit(this.count);
    }
   decrement() {
      this.count--;
      this.countChanged.emit(this.count);
   }
}

```

Now, let us look at the code in detail

First, as usual, we need to import output & EventEmitter from @angular/core

```

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

```

In the inline template, we have two buttons increment and decrement. We also displaying the current count

```

@Component({
   selector: 'child-component',
   template: `<h2>Child Component</h2>
           <button (click)="increment()">Increment</button>
           <button (click)="decrement()">decrement</button>
           current count is {{ count }}
   `
})

```

In the child component, define the countChanged event of type EventEmiiter. Decorate the property with @Output decorator to make it accessible from the parent component

```

@Output() countChanged: EventEmitter<number> = new EventEmitter();

```

Finally, we raise the event in increment & decrement methods using emit

```

  increment() {
    this.count++;
    this.countChanged.emit(this.count);
   }
  decrement() {
    this.count--;
    this.countChanged.emit(this.count);
  }

```

Parent Component

In the parent component , we need to listen to the “countChanged” event

The “countChanged” event is enclosed in Parentheses. It is then assigned to the method “countChangedHandler” in the component class. The syntax is similar to Event Binding

```

<child-component [count]=ClickCounter (countChanged)="countChangedHandler($event)">

```

The countChangedHandler($event) method accepts the $event argument. The data associated with event is now available to in the $event property

Our CountChangedHandler is as follows. It just updates the clickCounter and also logs the count to console

```

 countChangedHandler(count: number) {
   this.ClickCounter = count;
   console.log(count);
 }

```

The complete code is as follows

```

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

@Component({
  selector: 'app-root',
  template: `
        <h1>Welcome to {{title}}!</h1>
        <p> current count is {{ClickCounter}} </p>
        <child-component [count]=Counter (countChanged)="countChangedHandler($event)
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Component Interaction';
  Counter = 5;

  countChangedHandler(count: number) {
    this.Counter = count;
    console.log(count);
  }
}

```

Run the code. Whenever the increment/decrement buttons clicked, The child raises the event. The Parent component gets notified of the this and updates the counter with the latest value.

Parent uses local variable to access the Child in Template

Parent Template can access the child component properties and methods by creating the template reference variable

Child Component

Let us update the child component

```

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

@Component({
    selector: 'child-component',
    template: `<h2>Child Component</h2>
                current count is {{ count }}
    `
})
export class ChildComponent {
    count = 0;

     increment() {
        this.count++;
      }
    decrement() {
        this.count--;
    }
}
```

We have removed the input, output & eventemiitter.

Our component is now have property count and two methods to increment and decrement it

Parent component

```

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

@Component({
  selector: 'app-root',
  template: `
        <h1>{{title}}!</h1>
        <p> current count is {{child.count}} </p>
        <button (click)="child.increment()">Increment</button>
        <button (click)="child.decrement()">decrement</button>
        <child-component #child></child-component>` ,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Parent interacts with child via local variable';
}

```

We have created a local variable, #child, on the tag <child-component>. The “child” is called template reference variable, which now represents the child component

The Template Reference variable is created, when you use #<varibaleName> and attach it to a DOM element. You can then, use the variable to reference the DOM element in your Template

```

    <child-component #child></child-component>` ,

```

Now you can use the local variable elsewhere in the template to refer to the child component methods and properties as shown below

```

    <p> current count is {{child.count}} </p>
    <button (click)="child.increment()">Increment</button>
    <button (click)="child.decrement()">decrement</button>

```

The code above wires child components increment & decrement methods from the parent component

The local variable approach is simple and easy. But it is limited because the parent-child wiring must be done entirely within the parent template. The parent component itself has no access to the child.

You can't use the local variable technique if an instance of the parent component class must read or write child component values or must call child component methods.

Parent uses a @ViewChild() to get reference to the Child Component

Injecting an instance of the child component into the parent as a @ViewChild is the another technique used by the parent to access the property and method of the child component

The @ViewChild decorator takes the name of the component/directive as its input. It is then used to decorate a property. The Angular then injects the reference of the component to the Property

For Example

In the Parent component, declare a property child which is of type ChildComponent

```

child: ChildComponent;

```

Next, decorate it with @ViewChild decorator passing it the name of the component to inject

```

@ViewChild(ChildComponent) child: ChildComponent;

```

Now, when angular creates the child component, the reference to the child component is assigned to the child property.

We now update the code from previous section

Child Component

There is no change in the child component

Parent Component

In the parent component, we need to import the viewChild Decorator. We also need to import the child component

```

import { Component, ViewChild } from '@angular/core';
import { ChildComponent } from './child.component';

```

Next, create a property child which is an instance of type ChildComponent. Apply the viewChild Decorator on the child component as shown below

```

   @ViewChild(ChildComponent) child: ChildComponent;

```

Finally, add increment and decrement method, which invokes the methods in the child component

```

  increment() {
    this.child.increment();
  }

```

```
decrement() {
  this.child.decrement();
}

```

Now, the parent can access the properties and methods of child component

And in the template make necessary changes

```

      <h1>{{title}}</h1>
      <p> current count is {{child.count}} </p>
      <button (click)="increment()">Increment</button>
      <button (click)="decrement()">decrement</button>
      <child-component></child-component>`

```

The complete app.component.ts is as follows

```

import { Component, ViewChild } from '@angular/core';
import { ChildComponent } from './child.component';

@Component({
  selector: 'app-root',
  template: `
      <h1>{{title}}</h1>
      <p> current count is {{child.count}} </p>
      <button (click)="increment()">Increment</button>
      <button (click)="decrement()">decrement</button>
      <child-component></child-component>` ,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Parent calls an @ViewChild()';

  @ViewChild(ChildComponent) child: ChildComponent;

  increment() {
    this.child.increment();
  }

```