Building Application using Angular

Event Emitter

Passing Data From Parent to Child

- The Angular Components communicate with each other using **@Input Annotation.** We also look at how child components detect changes to these Input properties using OnChanges life Cycle hook or with a Property Setter
- The Parent Component can communicate with the child component by setting its Property**.** To do that the Child component must expose its properties to the parent component. The Child Component does this by using the @Input decorator. To do that the Child component the Parent Component can communicate with the child must expooes this by using the @Input decorator
- In the Child Component
  - Import the @Input module from @angular/Core Library

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

  - Mark those property, which you need data from parent as input property using @Input decorator

```
@Input() count: number;
```

- In the Parent Component
  - Bind the Child component property in the Parent Component when instantiating the Child

```
<child-component [count]=Counter></child-component>
```

Detecting Input Change

- There are two ways of detecting when input changes in the child component in Angular
- Using OnChanges LifeCycle Hook
  - ngOnChanges is a lifecycle hook, which angular fires when it detects changes to data bound input property. This method receives a SimpeChanges object, which contains the current and previous property values. We can Intercept input property changes in the child component using this hook.
- Using Input Setter
  - We can use the property getter and setter to detect the changes made to the input property as shown below
-

Example ngOnChanges for Change Detection

- Import the OnChanges interface, SimpleChanges, SimpleChange from @angule/core library.

```
import { Component, Input, OnChanges, SimpleChanges, SimpleChange  } from '@angular/core';
```

- Implement the ngOnChanges() method. The method receives the SimpleChanges object containing the changes each input property.

```
export class ChildComponent implements OnChanges {
  @Input() count: number;

  ngOnChanges(changes: SimpleChanges) {

    for (let property in changes) {
      if (property === 'count') {
        console.log('Previous:', changes[property].previousValue);
        console.log('Current:', changes[property].currentValue);
        console.log('firstChange:', changes[property].firstChange);
      }
    }
  }
```

Using Input Setter Example

- In the Child Component create a private property called _count

```
private _count = 0;
```

- Create getter & setter on property count and attach @Input annotation. We intercept the input changes from setter function and log it to console

```
@Input()
set count(count: number) {
   this._count = count;
   console.log(count);
}
get count(): number { return this._count; }
```

Passing Data from Child to Parent Component

- There are three ways in which parent component can interact with the child component
  - Parent Listens to Child Event
  - Parent uses Local Variable to access the child
  - Parent uses a @ViewChild to get reference to the child component

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.

Child Component Example

```
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);
  }
}
```

Parent Component Example

```
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)"></child-component>`  ,
  styleUrls: ['./app.component.css']
})
export class AppComponent  {
  title  = 'Component  Interaction';
  Counter  = 5;

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

Parent uses local variable to access the Child in Template

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--;
   }
}
```

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

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

Child Component

No change

Parent Component

```
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();
 }

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

Recap

- Passing Data from Parent to Child
- Passing Data from Child to Parent

Example

- https://angular.io/api/core/EventEmitter
- https://dzone.com/articles/understanding-output-and-eventemitter-in-angular
- https://dzone.com/articles/understanding-output-and-eventemitter-in-angular
- https://angular.io/api/core/ViewChild