**Discussion 6.1 – Angular Input and Output**

Patrick Cuauro

Bellevue University

WEB 425-307O Angular with TypeScript

Professor Krasso

September 11, 2023

# Angular Input and Output Properties

Angular Framework is composed of components, there’s the need to transfer data between components, specifically between Parent Components and Child Components, the properties @Input and @Output are designed to achieve this functionality.

We can define the properties for components we create and make them available in our whole application. In Angular using “events” we can notify the parent components that something has changed.

Input property specifies the properties which we can set on a component from a parent, whereas Output property identifies the events a component can fire to send information up the hierarchy to its child component.

@Input Property

Angular input decorator sets an event triggering when you find a property binding with this name and map it to a different named component property. Or if we don’t give an alias use the component property name.

It is a recommended practice to avoid using that approach by default, instead using the call property name.

|  |
| --- |
| @Component({  selector: 'order-processor',  Child  template: `...`  })  class OrderComponent {  @Input() quantity: number;  @Input()  set stockSymbol(value:string) {  // process the stockSymbol change here  } |

|  |
| --- |
| <order-processor [stockSymbol]= ''stock'' quantity=''100''>  Parent  </order-processor> |

It is the Decorator that marks the class field as an input property and supplies the configuration meta-data. The input property is bound to the DOM property’s value.

## Output Property:

Unlike input property, output property is not straight forward. We need to use EventEmitter to emit the value when an event occurs and in our child component html we need to get the data using event binding syntax.

* Create an output property similar to Input but of type *EventEmitter.*
* Assign it to an instance of *EventEmitter*

|  |
| --- |
| @Output('child-text') childText : EventEmitter<any> = new EventEmitter<any>(); |

* Now add a method in class which will be bound to the child component with a DOM event like *keyup*in this case

|  |
| --- |
| <input *type*="text" *(keyup)*="detectChange($event)">Text from Child component |

* Inside out *detectChange()*method we will be emitting the value

|  |
| --- |
| detectChange(event: any) {  *this*.childText.emit(event.target.value);  } |

* In our parent component we need to create a method which would be used to receive the emitted value.

|  |
| --- |
| saveText(text : any) { *this*.childInputText = text;} |

* Now in parent component html with our child element bind the saveText method to the output property “*child-text”.*

|  |
| --- |
| <app-child *[parent-text]*="parentText" *(child-text)*="saveText($event)">  </app-child> |

* To refer the emitted value, we need to use “*$event*”
* Finally display the text in parent component using string interpolation

|  |
| --- |
| <h3 *class*="text-success input-label">Child says : {{childInputText}}</h3> |

Sources:

* Angular Input Output Properties

<https://javascript.plainenglish.io/angular-input-output-properties-cb02a1a543f8>

* Understanding Input and Output properties in Angular

<https://shinumathew.medium.com/understanding-input-and-output-properties-in-angular-845c21703a38>