academeez

@Output

Pass event from component/directive

up to the parent

@Output

- @Output is used to send a custom event that happened in the child, back to the parent

Parent

```
@Component({
  selector: 'app-root',
  template: `
    <academeez-child (someEvent)="childEvent()">
    </academeez-child>
  `,
})
```

Child

```
export class ChildComponent {
  @Output()
  someEvent: EventEmitter<string> = new EventEmitter();

  somethingHappened() {
    this.someEvent.emit('hello parent');
  }
}
```

@Output decorator

academeez

- To create a custom event in the child we have to use the @Output decorator
- That decorator can get an optional string to specify the attribute name the parent need to use to connect to that event

Parent

```
@Component({
  selector: 'app-root',
  template: `
    <academeez-child (eventName)="childEvent()">
    </academeez-child>
  `,
})
```

Child

```
export class ChildComponent {
  @Output('eventName')
  …
}
```

EventEmitter

academeez

- In this child the @Output decorator need to decorate a property in the child of type **EventEmitter**
- The **EventEmitter** gets a generic type describing the data type the child will send to the parent
- Calling **emit** will invoke the event synchronously or asynchronously

```
export declare interface EventEmitter<T> extends Subject<T> {
  new(isAsync?: boolean): EventEmitter<T>;
  emit(value?: T): void;
  subscribe(generatorOrNext?: any, error?: any, complete?: any): Subscription;
}
```

$event

- When the child want to trigger the event, it can pass with the event data that will be sent to the parent
- The child can do that by calling the emit and passing data. **eventEmitter.emit(someData)**
- The parent can reference that data with **$event** in the template

Parent

```
@Component({
  selector: 'app-root',
  template: `
    <academeez-child (someEvent)="childEvent($event)">
    ↙academeez-child>
  `,
})
```

Child

```
export class ChildComponent {
  @Output()
  someEvent: EventEmitter<string> = new EventEmitter();

  somethingHappened() {
    this.someEvent.emit('hello parent');
  }
}
```

Summary

academeez

- With @Output we create an event on a child component or directive
- That event will be triggered when something happens on the child and the parent can attach a method on the class that will be called when the event happens
- To pass an event the child will define a property of type **EventEmitter** which extends a **Subject**
- The child can also pass data with the event by passing the data to **emit** the parent can reference the data with **$event**
- It is often the case where you want to create a reusable component that you will aim for minimal amount of coupling, therefore the main method of communication with that component will be **@Input, @Output**

academeez

Thank You

Next Lesson: Template reference variables