Tuesday:Emitting Events

Emitting Events

When we want a child component to communicate with a parent component, we make the child component emit an event that is taken up by the parent component. Let us make our child component *GoalDetailComponent,* communicate with its parent component, the *GoalComponent.* We'll do this by adding a button to the *GoalDetailComponent* that deletes a goal once we ascertain that we have completed it.

Let's create this button in our *GoalDetail* HTML template.

src/app/goal-detail.component.html

```
<p>{{goal.description}}</p>
<button (click)= 'goalComplete(true)'>Complete</button>
```

We have added a button below the goal description `<p>` tag and defined some logic inside it. We have added a click event binding syntax which will call the `goalComplete(true)` function once the button is clicked. Let's now create this function in the logic file of the *GoalDetailComponent*.

src/app/goal-detail.component.ts

```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
...
export class GoalDetailComponent implements OnInit {

  @Input() goal: Goal;
  @Output() isComplete = new EventEmitter<boolean>();

  goalComplete(complete:boolean){
    this.isComplete.emit(complete);
  }
  constructor() { }

  ngOnInit() {
  }

}
```

At the top, we have imported `Output` decorator and the `EventEmitter` class. We have then used the `Output` decorator to define `isComplete` as an EventEmitter that takes in a boolean. After that, we have created our `goalComplete()` function which calls the `emit` method on the `isComplete` EventEmitter. What this does is pass this event to the parent component. We, therefore, have to make the parent component process this event. Let's write the code for this in our parent component, *GoalComponent*.

src/app/goal/goal.component.html

```
<h1>My Goals</h1>
<hr>
<ul>
  <div *ngFor='let goal of goals;let i = index'>
    <li>{{goal.name}}</li>
    <button (click)='toggleDetails(i)'>Toggle Details</button>
    <app-goal-detail *ngIf='goal.showDescription' [goal]='goal' (isComplete) = 'completeGoal($even
t,i)'></app-goal-detail>
```

```
  </div>
</ul>
<p *ngIf='goals.length > 5'>You have too many goals</p>
```

To make the parent component receive this event, we need to catch the event being emitted in the parent component and define a function that will be triggered once this event is captured.

The event being emitted is `isComplete`. We catch this event in the parent component using parenthesis `()` and then define that when it is emitted, the `completeGoal($event,i)` function should be called. Inside the function, we have used the special variable, `$event` as a placeholder for the values that we expect to be emitted by the event. We have also passed in the index `i` of the goal item in the function.

Finally, let's create this `completeGoal()` function which will be triggered when our event is captured.

src/app/goal/goal.component.ts

```
...
export class GoalComponent implements OnInit {
...
  completeGoal(isComplete, index){
    if (isComplete) {
      this.goals.splice(index,1);
    }
  }

  constructor() { }

  ngOnInit() {
  }

}
```

Inside the function, we have passed in the `isComplete` event emitter and `index` as our arguments then used the `splice` javascript function to delete the goal at the index. Keep in mind, `isComplete` in this case, will return the value `true` that we set in the HTML template file of the *GoalDetailComponent* and `index` will be the exact index of a goal from the loop in the *GoalComponent* HTML template file.

When we serve our application now, we have the complete button which deletes a goal at a specific index from our goals array when clicked.