**EXERCISE SEVEN: COMPONENTS INPUT OUTPUT**

We will make a photo gallery and use @Input and @Output

1. Images should go into the /assets folder. Note that when adding something outside the root /src folder, we must restart the server – live reload does not work.
2. Create a component for a photo gallery.
3. Inside the ts file, import the necessary things.
   1. In the PhotoGalleryComponent class, add the following. The event is raised every time the user goes to a new photo.

@Input("photos")

imageList: string[] = []

@Output("on-navigate")

Emitter = new EventEmitter

currentIndex = 0

* 1. Add moveNext() and movePrevious() methods to the class:

moveNext() {

if (this.currentIndex < this.imageList.length - 1) {

this.currentIndex += 1

//Raise event with index as payload this.emitter.emit(this.currentIndex)

}

}

movePrevious() {

if (this.currentIndex > 0) { this.currentIndex -= 1

//Raise event with index as payload this.emitter.emit(this.currentIndex)

}

}

1. In the HTML template – note that we are binding the src attribute to a variable with [src]:

<div>

<div>

<img [src]="imageList[currentIndex]"/>

</div>

<div>

<button (click)="movePrevious()">Previous</button>

<button (click)="moveNext()">Next</button>

</div>

</div>

1. In the CSS for this component, set images to 100% width.
2. Use the photo gallery component in the books-inventory component template. We will add a gallery for every book that has photos, and provide the list of photos with [photos]
   1. Under the heading 3:

<p class="gallery">

<app-photo-gallery \*ngIf = "book.photos.length > 0" [photos]="book.photos">

</app-photo-gallery>

</p>

1. In the CSS for book inventory, style the gallery class to be 400px wide.
2. Disable Previous and Next buttons when applicable.

<button [disabled]="currentIndex==0"

(click)="movePrevious()">Previous</button>

<button [disabled]="currentIndex==imageList.length-1"

(click)="moveNext()">Next</button>