Tiutin Andrada-Georgia Group 248/2

Web Component - My Slider

Requirement

Build a software system that will have to illustrate the creation of a new component for a given framework.

Project

The web component which I created is called “my-slider” and consists of a horizontal slider for photos. It was realised using Angular framework.

Angular

Angular is an application design framework and development platform for creating efficient and sophisticated single-page apps. Moreover, Angular is a development platform, built on TypeScript. As a platform, it includes: a component-based framework for building scalable web applications, a collection of well-integrated libraries that cover a wide variety of features, such as routing, forms management, client-server communication and, also, a suite of developer tools to help with the development, building, testing and updating of the code.

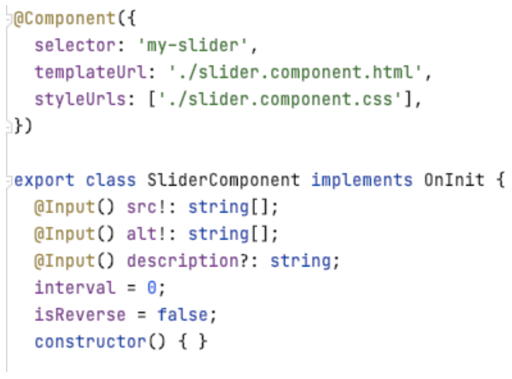
Components

In Angular, components are the building blocks that compose an application. A component includes a TypeScript class, which specifies the following Angular-specific information: a CSS selector that defines how the component is used in a template, an HTML template that instructs Angular how to render the component and an optional set of CSS styles that define the appearance of the template's HTML elements.

My Slider

As mentioned before, the created component relies on a horizontal slider for photos. The purpose of this component is to allow an easy integration of a continuous photo slider, inside a web page. The slider component consists of three files: a Typescript file, an HTML file and a CSS file.

The Typescript file (slider.component.ts) has an @Component decorator, an HTML template and styles. This decorator specifies the CSS selector, which states how the component will be named inside a template (in this case it will be “my-slider”), an html template (here is the relative path to slider.component.html file) and an optional set of CSS styles (the same way, it is the relative path to slider.component.css file).



```
@Component({
  selector: 'my-slider',
  templateUrl: './slider.component.html',
  styleUrls: ['./slider.component.css'],
})

export class SliderComponent implements OnInit {
  @Input() src!: string[];
  @Input() alt!: string[];
  @Input() description?: string;
  interval = 0;
  isReverse = false;
  constructor() { }
```

slider.component.ts

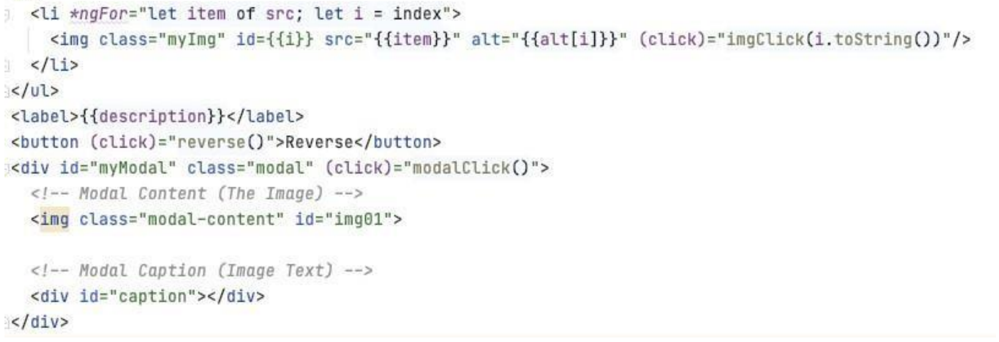
The slider component has three input values which are given as attributes, when used in an HTML file. These include: a required array containing the relative image paths, a required array containing an alternate text for of each image (in case the image cannot be displayed) and an optional description of the slide show.

The ngOnInit() function initialises the corresponding variables. Basically, what ngOnInit does in this context is to iterate through the list items from the HTML file, where each such item includes an img tag with provided src and alt attributes, and place them next to each other, on a horizontal line. Then, setInterval() is called, such that every 100 milliseconds, each list item is moved with ten pixels to the right, creating the idea of a slider. When a list item reaches the right border of the window, it is sent to appear from the left border of the page or, in case the list has more items than the window can display, it is positioned before the last element to be displayed, giving the impression of continuity. Therefore, even after all images have been displayed, the items will continue to appear again and again.

Each image in the slide show has a click event, which triggers the imgClick() function. In this case, the slider will stop (through clearInterval()) and the image that was clicked will be displayed in as a modal image, in a larger format, together with its alternative text. When the modal image or the background is clicked, the modal image disappears and the slider continues from the state where it was left.

There exists also a reverse button, which has associated a click event with the reverse() function. When clicked, the slider stops, due to clearInterval(), and setInterval() is called, such that every 100 milliseconds, each list item is moved with ten pixels to the left, changing the direction of the slider. The same way, when a list item reaches the left border of the window, it is sent to appear from the right border of the page or, in case the list has more items than the window can display, it is positioned before the first element to be displayed this time, giving once again the impression of continuity.

The HTML file (slider.component.html) is a template which instructs Angular how to render the component. It is comprised of an unordered list, where each item includes an image, having associated: src and alt attributes, id, class and a click action. The optional description is specified through a label, followed by the reverse button and then a div section for the modal content.

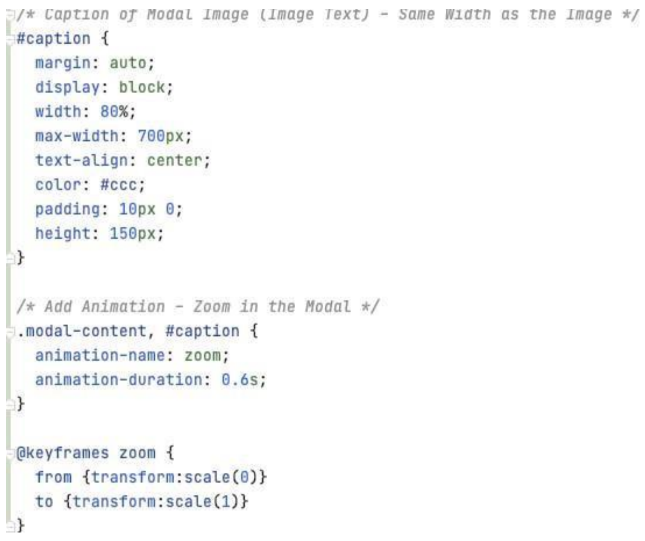


```html
<ul>
  <li *ngFor="let item of src; let i = index">
    <img class="myImg" id={{i}} src="{{item}}" alt="{{alt[i]}}" (click)="imgClick(i.toString())"/>
  </li>
</ul>
<label>{{description}}</label>
<button (click)="reverse()">Reverse</button>
<div id="myModal" class="modal" (click)="modalClick()">
  <!-- Modal Content (The Image) -->
  <img class="modal-content" id="img01">

  <!-- Modal Caption (Image Text) -->
  <div id="caption"></div>
</div>
```

slider.component.html

The CSS file (slider.component.css) defines the appearance of the template's HTML elements. It specifies both styles for the used HTML tags and classes, and zoom animation used for the modal content.



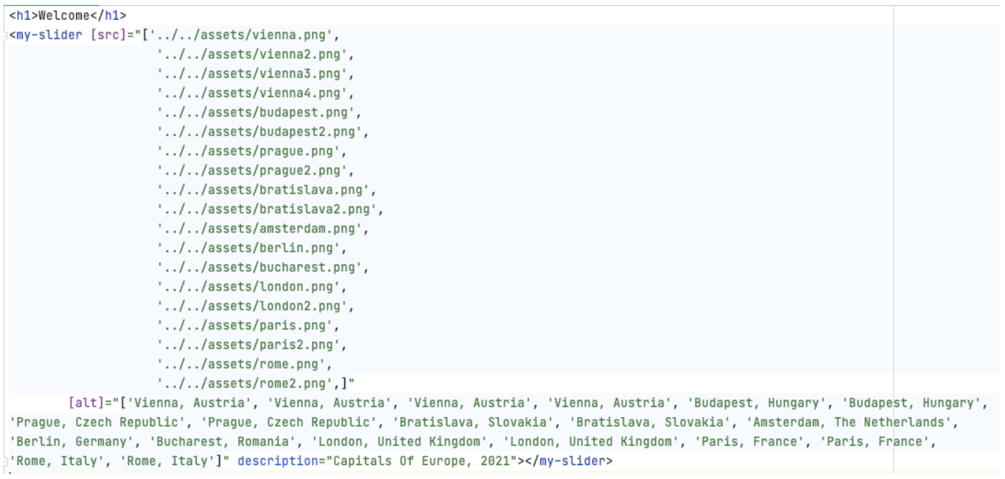
```css
/* Caption of Modal Image (Image Text) - Same Width as the Image */
#caption {
  margin: auto;
  display: block;
  width: 80%;
  max-width: 700px;
  text-align: center;
  color: #ccc;
  padding: 10px 0;
  height: 150px;
}

/* Add Animation - Zoom in the Modal */
.modal-content, #caption {
  animation-name: zoom;
  animation-duration: 0.6s;
}

@keyframes zoom {
  from {transform:scale(0)}
  to {transform:scale(1)}
}
```

slider.component.css

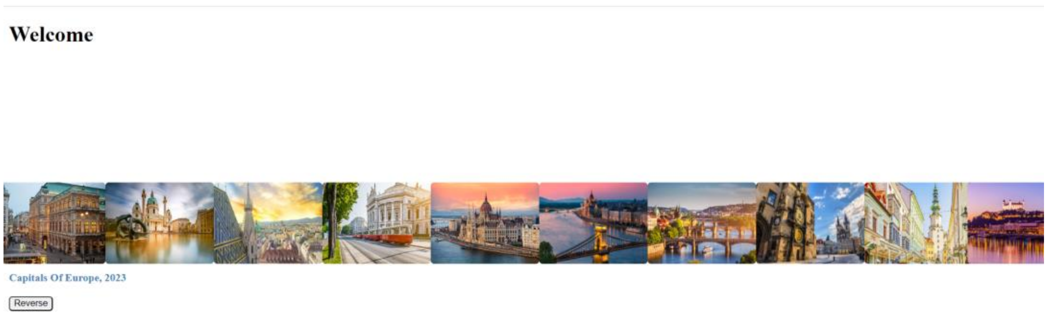
In the end, the “my-slider” component can be called from another HTML file, thus integrating the component described before in another component.



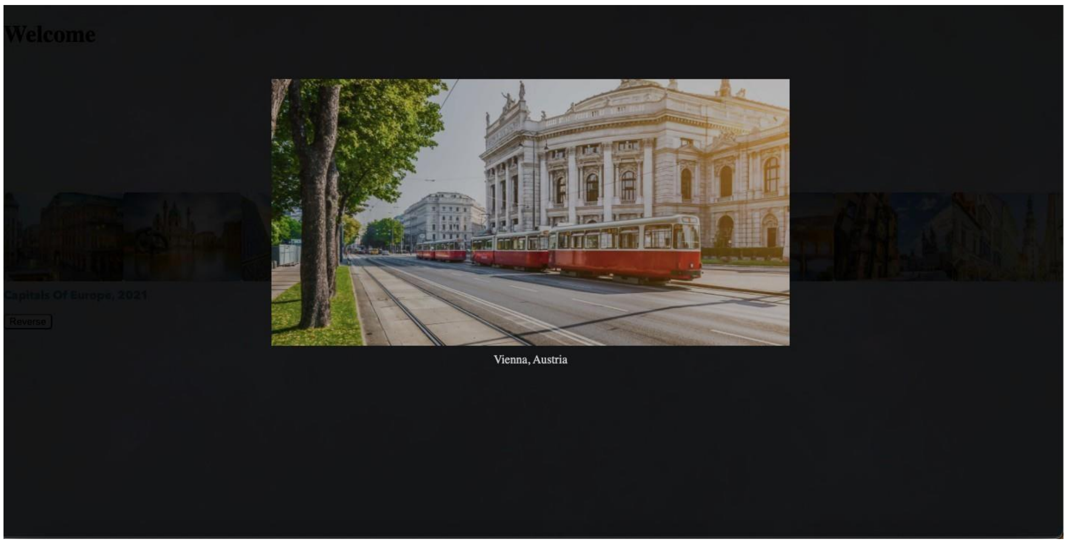
```html
<h1>Welcome</h1>
<my-slider [src]="['../../assets/vienna.png',
                   '../../assets/vienna2.png',
                   '../../assets/vienna3.png',
                   '../../assets/vienna4.png',
                   '../../assets/budapest.png',
                   '../../assets/budapest2.png',
                   '../../assets/prague.png',
                   '../../assets/prague2.png',
                   '../../assets/bratislava.png',
                   '../../assets/bratislava2.png',
                   '../../assets/amsterdam.png',
                   '../../assets/berlin.png',
                   '../../assets/bucharest.png',
                   '../../assets/london.png',
                   '../../assets/london2.png',
                   '../../assets/paris.png',
                   '../../assets/paris2.png',
                   '../../assets/rome.png',
                   '../../assets/rome2.png',]"
        [alt]="['Vienna, Austria', 'Vienna, Austria', 'Vienna, Austria', 'Vienna, Austria', 'Budapest, Hungary', 'Budapest, Hungary',
'Prague, Czech Republic', 'Prague, Czech Republic', 'Bratislava, Slovakia', 'Bratislava, Slovakia', 'Amsterdam, The Netherlands',
'Berlin, Germany', 'Bucharest, Romania', 'London, United Kingdom', 'London, United Kingdom', 'Paris, France', 'Paris, France',
'Rome, Italy', 'Rome, Italy']" description="Capitals Of Europe, 2021"></my-slider>
```

app.component.html

The resulting web application can be seen below:



Horizontal slider



Modal content

To sum it up, “my-slider” is a nice and easy to use web component, which can be integrated without problems inside another component of a web application.