ICP 5: Angular

Joe Moon
Email: jmn5y@umsystem.edu
Github: https://github.com/joemoon-0/WebMobile-2022Spring/tree/main/2022Spring-Web/Web_ICP5

Nathan Cheney
Email: ncxn8@umsystem.edu
Github: https://github.com/nathancheney/Web-Mobile-Programming/tree/main/ICP-5

Introduction

As the internet evolves, more websites are shifting focus from static webpages to single page applications that can perform many functions and present a variety of information without having to refresh the page. Normally, these features would require extensive JavaScript but the rise of JavaScript frameworks have allowed that development to be faster and simpler.

Angular is one such framework which was developed by Google and serves as the front-end framework for the popular MEAN web stack. It is designed to be modular allowing for code to be easily reused and is based on an architecture of component hierarchy.

To Do Task List

This application allows a user to add a task, mark it for completion, and delete it from the list. An <app-task-input> component is used to capture a new task entered by the user through the UI and then send that information to the parent component through the use of an EventEmitter as shown below:

```
22          <app-task-input
23            (addTaskEvent)="submitNewItem($event)"
24          ></app-task-input>
```

app.component.html

A similar procedure is used for displaying the tasks in a list however a structural directive is used that conditionally executes one template over another depending on a given condition. In this case, whether the user clicks the checkbox or not to indicate a task is complete will enable a class (completed) in the element which crosses the item out.

```
32          <input type="checkbox" (click)="completeItem(item)" />
33          <div
34            *ngIf="item.completed; then completedTask; else incompleteTask"
35          ></div>
36
37          <!-- Display crossed out task depending on completion -->
38          <ng-template #completedTask>
39            <div>
40              <span class="completed">{{ item.description }}</span>
41            </div>
42          </ng-template>
43
44          <ng-template #incompleteTask>
45            <div>
46              <span>{{ item.description }}</span>
47            </div>
48          </ng-template>
```

app.component.html

Countdown Timer

Two components are used to implement the countdown timer. The <app-time-setter> captures the user-specified Date and Time input as to when the timer will expire and saves it to an interface and then passes it to the parent component for the next step. This process is illustrated below:

```
65          <app-time-setter
66            id="time-set"
67            (addTimerEvent)="startClock($event)"
68          ></app-time-setter>
```

app.component.html

```
23    newTimer(date: any, time: any) {
24      // Construct user specified date into standard format
25      const userDate = new Date(`${date} ${time}:00`);
26
27      const newTimer: TimerInterface = {
28        years: userDate.getFullYear(),
29        months: userDate.getMonth(),
30        days: userDate.getDate(),
31        hours: userDate.getHours(),
32        minutes: userDate.getMinutes(),
33        seconds: userDate.getSeconds()
34      }
35      this.addTimerEvent.emit(newTimer);
36    }
```

time-setter.component.ts

With the user-specified target time, that information is then passed to the <app-clock> component as an @Input() parameter (in this case userDate) so that it can be implemented in the countdown function which uses the setInterval() method.

```
73          <app-clock
74            id="countdownTimer"
75            class="hide"
76            [userDate]="userDate"
77            (clearTimerEvent)="clearClock()"
78          ></app-clock>
```

app.component.html

Contribution

All members contributed equally to this report and ICP.