

WEB AND MOBILE PROGRAMMING

ICP-5:

COORDINATOR: Dr. Yugyung Lee

Instructor Name: Yeruva, Vijaya Kumari

TA's: Alanazi, Ahmed Hamdan, Cheng Shu

STUDENT:

Student Name: RITWIK REDDY KONGALA

Email: RKB4B@umsystem.edu

Video Link: <https://youtu.be/Fu0YXdcVw6s>

Source Code:

<https://github.com/RITWIKREDDY9/ICP-5-WEB>

TASK - 1: To-Do Application

In this Todo task, I have used an array to store the user defined data and retrieved back the data to the html page using getElementById annotation.

Initially create an array with one value as “My work” and check whether the search is empty or not using the if condition. If the condition satisfies then push the string data to the array.

To retrieve the data, the condition is used in the html page and displaying the data using `{{}}` annotation.

Two buttons are used, one to check the task and other is used to delete the task.

For check conditions, when the user is clicked on the button, the respective method is going to be executed and in the method used `` annotation to strike the word using `getElementById` annotation.

For delete, `splice` is used to remove the data in the array list.

HTML:

```
<div class="row" id="todo">
  <div class="col-sm-offset-3 col-md-6">
    <div class="input-group">
      <input id="addItem" type="text" class="form-control" [(ngModel)]="newItem" placeholder="Todo List item">
      <span class="input-group-addon" id="basic-addon2" (click)="submitNewItem()">add</span>
    </div>
    <br>
    <table class="table table-responsive">
      <thead>
        <tr>
          <th class="text-center">ToDo List Items</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let item of items; let i = index" class="text-center">
          <td id="{{i}}">
            {{item}}
          </td>
          <td class="text-right">
            <div class="btn-group">
              <button class="bt2" (click)="deleteItem(i)"><i class="fa fa-trash"></i></button>
              <button class="bt" (click)="completeItem(item,i)"><i class="glyphicon glyphicon-thumbs-up"></i></button>
            </div>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
</body>
</html>
```

JS:

```
export class AppComponent {
  title = 'ToDoList';
  items = ['My Work'];
  newItem = '';

  submitNewItem() {
    if (this.newItem !== '') {
      this.items.push(this.newItem);
      this.newItem = '';
    }

    // @ts-ignore
    document.getElementById('addItem').focus();
  }

  Complexity is 3 Everything is cool!
  completeItem(item: any, i: any) {
    // @ts-ignore
    document.getElementById(i).innerHTML = '<del>' + item + '</del>';
  }

  deleteItem(i: any) {
    this.items.splice(i,1);
  }
}
```


OUTPUT:


Todo List item

add


ToDo List Items


My Work





Driving





Todo List item

add

ToDo List Items

~~My Work~~





Driving





Todo List item

add

ToDo List Items

Driving





TASK - 2: Counter

In this task, a dynamic date is taken from the user and the counter is going to work based on the given date from the present date.

Declared variable to use the mathematical operations and Initially store difference date values of given and present date.

Initially, we will check whether the given date is greater than the present date or not. If true, then error message is popped up and if false the `math.floor` is used to calculate the days, months, hours, minutes and seconds and the calculated timer is displayed to the html page.

HTML:


```
<div class="event-form">
  <p *ngIf="ispresent" class="error">Date need to be greater than present date</p>
  <div class="form-item">
    <label for="birthday" style="color: #008080;font-size: large;font: bold;">Your Date</label>
    <input type="date" id="eventDate" name="eventDate" (change)="initializeCounter()" [(ngModel)]="dynamicDate">
  </div>
</div>

<div class="event-container">
  <div class="timer">
    <div class="days flex-item">
      <h3 class="big-text">{{counter[0]}}</h3>
      <p class="small-text">Days</p>
    </div>
    <div class="hours flex-item">
      <h3 class="big-text">{{counter[1]}}</h3>
      <p class="small-text">Hours</p>
    </div>
    <div class="minutes flex-item">
      <h3 class="big-text">{{counter[2]}}</h3>
      <p class="small-text">Minutes</p>
    </div>
    <div class="seconds flex-item">
      <h3 class="big-text">{{counter[3]}}</h3>
      <p class="small-text">Seconds</p>
    </div>
  </div>
</div>
```

JS:

```
export class AppComponent implements OnInit {  
  //variable declaration  
  public dynamicDate!: String;  
  public ispresent: boolean = false;  
  public counter: any = ['-','-','-','-'];  
  
  Complexity is 3 Everything is cool!  
  setCountDown () {  
    //const array used to store the values dynamically  
    const [Y, M, D ] = this.dynamicDate.split('-');  
    //selected date by the user  
    const choiceDate = new Date(+Y, +M - 1, +D);  
    // checking whether the selected date is greater than present date  
    if(choiceDate < new Date()) {  
      this.ispresent = true  
    } else {  
      // math calculation  
      const x = (choiceDate.getTime() - new Date().getTime())/1000;  
      const DD = Math.floor(x / (60 * 60 * 24));  
      const HH = Math.floor((x % (60 * 60 * 24)) / (60 * 60));  
      const MM = Math.floor((x % (60 * 60)) / (60));  
      const SS = Math.floor((x % (60)));  
      // values are stored in a array  
      this.counter = [DD,HH,MM,SS]  
    }  
  }  
  // primary method calling  
  initializeCounter() {  
    setInterval(() => {  
      this.setCountDown()  
    }, 1000)  
  }  
}
```

OUTPUT:

Your Date 11/30/2021 

7 0 53 3

Days Hours Minutes Seconds
