LAB6 - Using pipes

In this lab, we will create a custom pipe for task priorities, showing a label for each priority value from the `Priority` enum,

Creating the pipe

In this section, you will create a new pipe : `PriorityPipe`

1. Using the Angular CLI, generate a new pipe in the `src/app` folder

   ```
   ng generate pipe priority
   ```

   > You can delete the generate unit test file `priority.pipe.spec.ts`

2. Change the `transform` method signature to it accepts a `Priority` and returns a `string`

   ```
   transform(value: unknown, ...args: unknown[]): unknown {
   ```

3. Implement the `transform` function so that it returns a label according to each priority value from `Priority` enum

   ```
   transform(value: Priority): string {
       switch(value) {
           case  Priority.Low: {
               return  "low"
           }
           case  Priority.Normal: {
               return  "normal"
           }
           case  Priority.High: {
               return  "high"
           }
           default : {
               return  "--"
           }
       }
   }
   ```

Using the pipe

In the previous labs, in the `TaskCreatorComponent` template we were rendering the list of priorities manually in the select options

```
<select  [(ngModel)]="newTask.priority"  >
    <option  [value]="low.value">
        {{low.label}}
    </option>
    ...
</select>
```

That is not optimal, especially when the `Priority` enum changes over time, in this section we will use our pipe along with ngFor directive to render the `Priority` values dynamically.

1. Open the `task-creator.component.ts` file , and remove the previous properties : `low` , `normal` and `high` (We no longer need them)
2. Instead add a single property in the component : `priorities` which will contain the list of priorities that we want to make them available to user in the select options

   ```
   priorities = [Priority.Low, Priority.Normal, Priority.High]
   ```

3. Open the `task-creator.component.html` file and use `*ngFor` directive to loop over `priorities` to create the correponding `option`

   ```
   <option  *ngFor="let  value  of  priorities"  [value]="value">
       {{value}}
   </option>
   ```

4. If you run the code and inspect the UI, you will notice that in the `select` form element, the options are 0, 1, 2. That is not relevant for the user, to resolve this issue we will use our created pipe to transform the values into labels. Apply the 'priority' pipe to `value` in the HTML template

   ```
   <option  *ngFor="let  value  of  priorities"  [value]="value">
       {{value | priority}}
   </option>
   ```

Chaining pipes

Since the labels are in lowercase, we can apply another pipe like `titlecase` or `uppercase` to improve the text appearence

```
{{value | priority | uppercase}}
```