Angular tutorial 2 (medium)

Plunker link: http://bit.ly/2bKUxuo

In this tutorial we are going to create a simple todo list. To get started open the plunker link. You will see that we already have a todo list. The problem is that this is hard coded and we can not enter anything new in. Lets first start by making the list show our own tasks. Open the app/app.component.ts file and add a new variable to your class called ‘tasks’ of type string array

```
export class AppComponent {
    tasks:string[] = ['Clean car', 'Finish assignment'];
}
```

We now have some tasks to add to our template. We seen in tutorial 1 how to add a value from our class to our template, but that was for a single value. How can we add a collection of values? Luckily Angular have thought about this and have given us a directive to help out. It is the ‘ngFor’ directive that will loop over a collection and repeat the element it was attached to. Let’s use this in our app

```
@Component({
    selector: 'my-app',
    template: `
      <h1>Angular Tutorial</h1>
```

```
      <ul>
        <li *ngFor="let task of tasks">{{item}}</li>
      </ul>
  `
})
export class TodoComponent {
    tasks: string[] = ['Clean car', 'Finish assignment'];
...
```

Now it should repeat for every task in the tasks variable! Because we attached the ngFor directive to the list item (li) element, ngFor will create a li element for every task in the tasks array. In the expression we pass ngFor we see that it will assign each value of the tasks array to the task variable. Because ngFor does this for us we can then use that task in the template (e.g like {{task}}). The assignment is only temporary though and when ngFor finishes looping we won't be able to access the task variable any more.

We now have an app that will show all our tasks we have to do. But the functionality is a little light. It would be better if we could add a task to our todo list. Because our task list template is using the tasks array we should add the new task to that and Angular will automatically pick up the change and show it. Let's start by adding an input box and an add button

```
@Component({
  selector: 'my-app',
```

```
  template: `
  <h1>Tasks</h1>
  <ul>
    <li *ngFor="let task of tasks">{{item}}</li>
  </ul>
  <form class="form-inline">
    <input type="text" class="form-control" placeholder="
Enter in task" />
    <button class="btn  btn-default">Add</button>
  </form>
  `
})
...
```

Don’t worry about the attributes on the input and button elements as they are just styling sugar. Now that we have an input and an add button we want the app to add the new task in the input when the user clicks the add button. Recall from tutorial 1 that we can bind a JavaScript value to an element using ngModel. We can do this in this app so that when users enter in a new task we will have a JS object to use.

```
<input [(ngModel)]="newTask" type="text" class="form-cont
rol" placeholder="Enter in task" />
```

Now we want the app to add the newTask JS value when the user clicks the add button. But first we need to have a function that will add

a new task to our task list.

```
export class TodoComponent {
    tasks: string[] = ['Clean car', 'Finish assignment'];

    /* adds a new task to our task list */
    addTask(task: string): void {
        this.tasks.push(task);
    }
...
```

This function will take a string, our new task, and add it to our tasks array. Now we can use this function when a user clicks the add button. To do this we bind to the click event. Angular has an easy way to do this using (event)=“expression” in your template and passing the code you want to run to this event. Let’s add this to our app to see how it works

```
<button (click)="addTask(newTask)" class="btn  btn-defaul
t">Add</button>
```

We can call the addTask function from our todo component class and pass it the newTask object. We can use the newTask object because it was defined by ngModel on our input (even if the input is empty). Now when a user clicks add, our app will add a new task to the list and update our view!

Check the plunker link below to see the complete version

Challange

What happens if we add a task that we dont want? We have no way of deleting it from our list. Try and add a delete button that will remove a task from the task list.

Hint: You can find the index of a string in an array of strings using array.indexOf(string) replacing ‘array’ and ‘string’ with your values.

Resources

Plunker link: http://bit.ly/2bKUxuo

Plunker tutorial complete link: http://bit.ly/2bvOC8P