Angular Introduction

Goal

In this lab, you will:

- Create a basic Angular application
- Display a list of to-do items
- Add new to-do items to the list
- Mark items as completed

Your mission

In this lab you will create a simple to-do application. Using this application you can manage a list of items that need to be done and add new items to the list. You can also mark items as completed and remove all completed items from the list.

Type it out by hand?

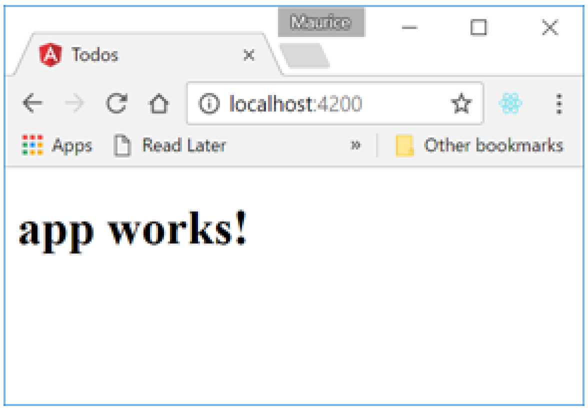
> Typing it drills it into your brain much better than simply copying and pasting it. You're forming new neuron pathways. Those pathways are going to help you in the future. Help them out now.

Create a basic application using the Angular CLI

In this section, you will use the Angular CLI to create a simple to-do list application.

1. Open the **begin** folder of the lab. Notice there is nothing here yet.
2. Create a new Angular application named **todos** using the Angular CLI. This can be done with the command: `ng new todos`
3. Make sure the application works by navigating to the new **todos** folder, starting the web server using the CLI and opening this in the web browser at http://localhost:4200/.

```
cd todos
ng server
```



app works!

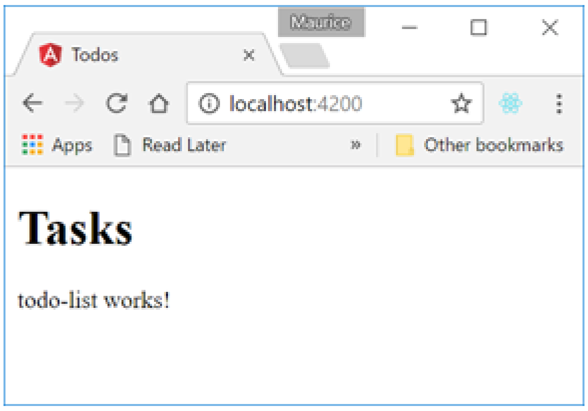
4. Create to-do list component using the Angular CLI. Note that it is useful to open a second terminal window so you don't have to stop the web server every time.

```
ng generate component TodoList
```

5. Open **app.component.ts** and change the **title** property to **Tasks**.
6. Open **app.component.html** and add the **TodoList** component to the markup.

```
<h1>
  {{title}}
</h1>

<app-todo-list></app-todo-list>
```

7. Switch back to the browser. You should see it update automatically.



Tasks

todo-list works!

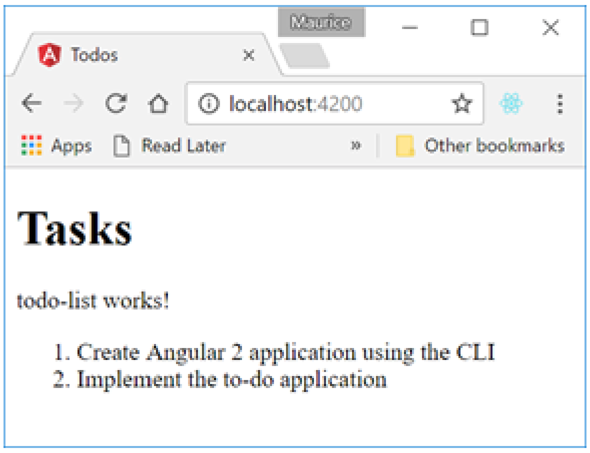
8. Open **todo-list.component.ts** and add an array named **todos** to contain all to-do items. Because there is no way to add new items yet it is helpful to add a few tasks.

```
  todos: any[];

  ngOnInit() {
    this.todos = [
      { title:'Create Angular application using the CLI', don
e: true },
      { title:'Implement the to-do application', done: false
},
    ];
  }
```

9. Open **todo-list.component.html** and add markup to display the list of items. Use the **ngFor** directive to display the list of to-do items.

```
<ol>
  <li *ngFor="let todo of todos">
    {{todo.title}}
  </li>
</ol>
```

10. Switch back to the browser. You should see the two to-do items.



Tasks

todo-list works!

1. Create Angular 2 application using the CLI
2. Implement the to-do application

Adding new to-do items and marking them as complete

In this section, you will make the application interactive and allow the user to add new tasks to the list.

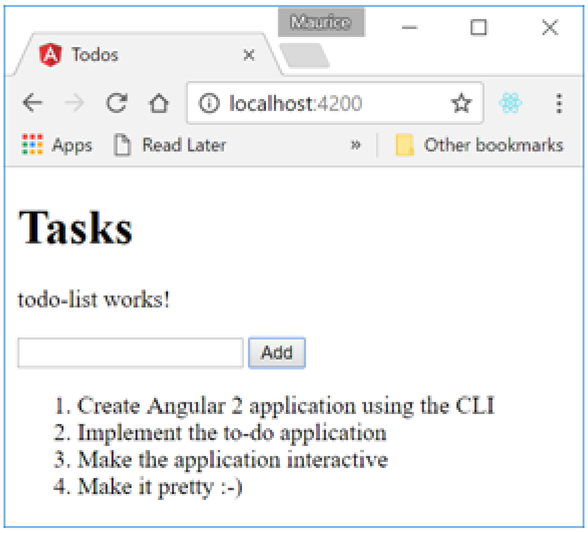
1. Open **todo-list.component.ts** and add a string property named **newTodo** to hold the new to-do text and the code to add a new to-do item to the the **todos** array.

```
  newTodo: string = 'Make the application interactive';
  addTodo() {
    this.todos.push({
      title: this.newTodo,
      done: false
    });
    this.newTodo = '';
  }
```

2. Open **todo-list.component.html** and add the required input element and button. Use the **ngModel** and event binding to the **click** property to make the form operational.

```
<div>
  <input type="text" [(ngModel)]="newTodo" />
  <button (click)="addTodo()">
    Add
  </button>
</div>
```

3. Switch back to the browser. You can now add new items to the list of to-do items.



Tasks

todo-list works!

1. Create Angular 2 application using the CLI
2. Implement the to-do application
3. Make the application interactive
4. Make it pretty :-)

4. Because a to-do item needs to have interactivity it is best to turn it onto a separate component. Create a new component named **TodoItem**.

```
ng generate component TodoItem
```

5. Open **todo-list.component.html** and replace the contents of the HTML `<li>` tag with the new **TodoItem** control. Pass the current to-do items as the item property of the TodoItem control.

```
<ol>
  <li *ngFor="let todo of todos">
    <app-todo-item [item]="todo"></app-todo-item>
  </li>
</ol>
```

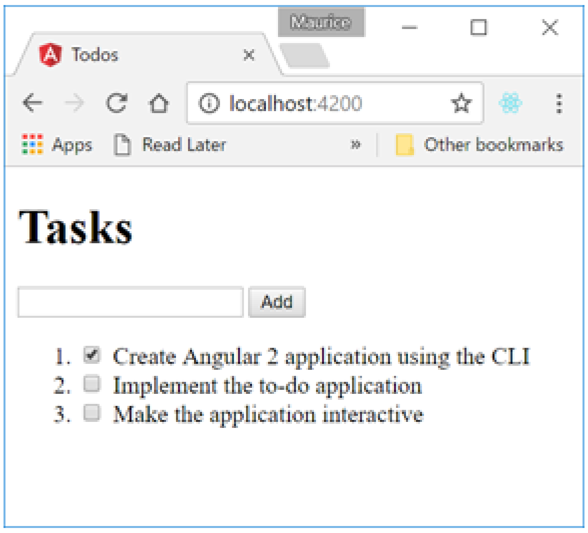
6. Open **todo-item.component.ts** and add an input property named **item** to hold the current to-do item.

```
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-todo-item',
  templateUrl: './todo-item.component.html',
  styleUrls: ['./todo-item.component.css']
})
export class TodoItemComponent implements OnInit {

  @Input() item: any;

  constructor() { }

  ngOnInit() {
  }
}
```

7. Open **todo-item.component.html** and add the markup to display the to-do item. Add a HTML `<input>` element of type checkbox and use the **ngModel** directive to bind it to the item **done** property.

```
<div>
  <input type="checkbox" [(ngModel)]="item.done" />
  {{item.title}}
</div>
```

8. Switch back to the browser. You can now mark to-do items as done using the checkbox.



Tasks

1. ☑ Create Angular 2 application using the CLI
2. ☐ Implement the to-do application
3. ☐ Make the application interactive

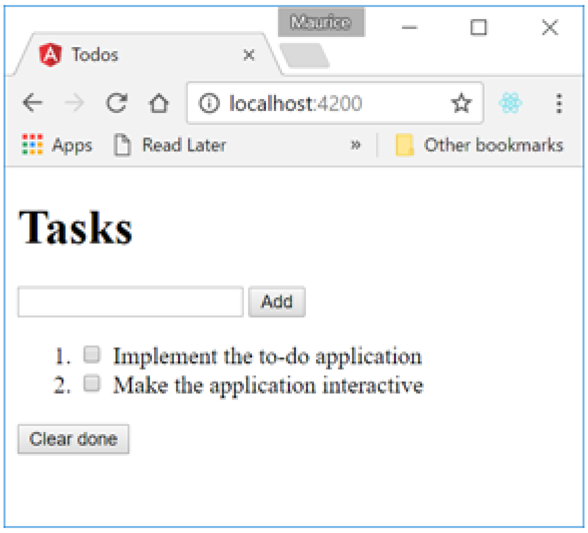
9. Open **todo-list.component.html** and add an HTML `<button>` element to remove all completed items from the list of tasks. Add a property binding for the **click** handler to call the `clearDone()` function.

```
<div>
    <button (click)="clearDone()">Clear done</button>
</div>
```

10. Open todo-**list.component.ts** and add the `clearDone()` function to remove done items from the list.

```
clearDone() {
  this.todos = this.todos.filter(todo => !todo.done);
}
```

11. Switch back to the browser. You can now remove all items that are marked as done.



Tasks

Add

1. Implement the to-do application
2. Make the application interactive

Clear done

Improve the user interface

The to-do list is functional but can be improved upon.

1. Use a strike-through font style when a to-do item is marked as **done**. Open **todo-item.component.css** and add the following CSS class.

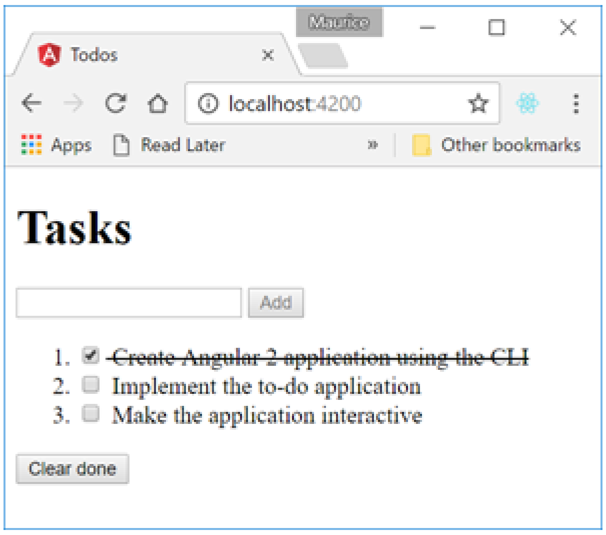
```
.done {
  text-decoration: line-through;
}
```

2. Open **todo-item.component.html** and update the markup to use the new class when an item is marked as done.

```
<div [ngClass]="{done: item.done}">
  <input type="checkbox" [(ngModel)]="item.done" />
  {{item.title}}
</div>
```

3. Disable the **Add** button if no text to add.

```
  <button (click)="addTodo()" [disabled]="!newTodo">
    Add
  </button>
```



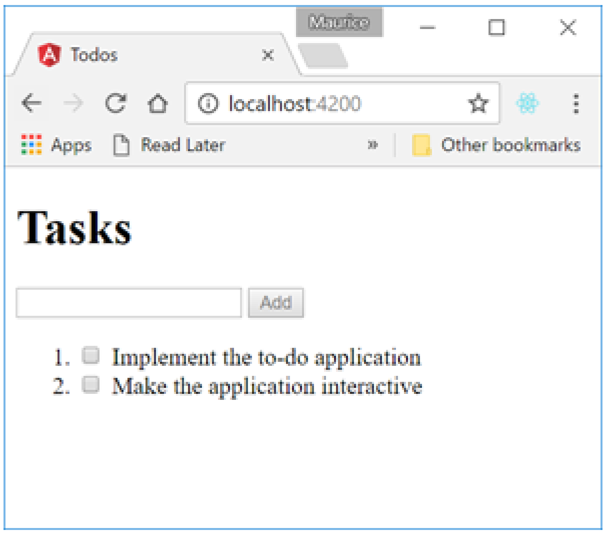
Tasks

1. ~~Create Angular 2 application using the CLI~~
2. Implement the to-do application
3. Make the application interactive

4. Remove the **Clear** button if there are no completed to-do items.

```
get allDone() {
  return !this.todos.some(todo => todo.done);
}
```

```
<div *ngIf="!allDone">
    <button (click)="clearDone()">Clear done</button>
</div>
```



Tasks

1. Implement the to-do application
2. Make the application interactive

Solution

The Solution can be found in **complete** folder