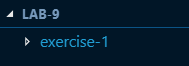
**COMP 3123 – Full Stack Development – Lab 9**

* Angular Application Structure
* Components + Angular CLI

**Developer Note:**When working on your exercises, please create separate folder for your work. This way you won’t putting all your code in the same file, which can pollute the global name space. In short, it will prevent you from overwriting your own work and causing your code to compile incorrectly.

Organize your folder structure in this way.

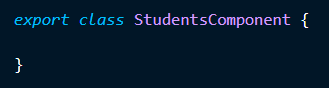


**Exercise #1 – Angular Component**

1. Create a folder named Lab 9
2. Open a command prompt create a directory for **exercise-1**
3. Open Visual Studio Code and open the folder **exercise-1**
4. Create a new project and default application named ***student-app*** by running the following Angular CLI command.



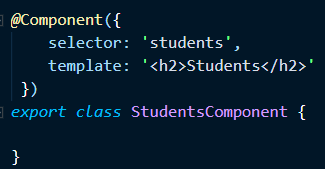
1. In the project structure, navigate to the source foleder (src) and then to the app folder. Right click and add a new file named ***students.component.ts***
2. Create basic TypeScript class named StudentsComponent using Pascal casing, where every word starts with a capital letter.



1. Import the Component decorator from angular core.

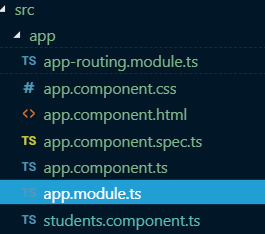


1. Add the Component metadata to make the class an Angular Component.

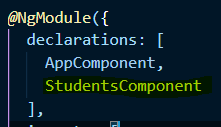


1. Next we need to register this component in App module. Navigate to ***app.module.ts***

in app folder.



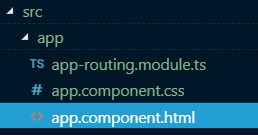
1. In the ***app.module.ts*** modify the ***NgModule*** decorator section to register the ***StudentComponent***..



Type the name of the ***StudentComponent*** and press TAB, the import statement should auto-populate at the top of the page.



1. Navigate to the app.component.html file. Open the file and delete all of the existing HTML markup in the page.



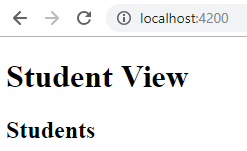
1. Type following HTML code with out student selector <students></students>



1. Use the Angular CLI to run the ng serve command to compile and load the application on localhost.



1. Navigate to localhost:4200 to view the component in browser.

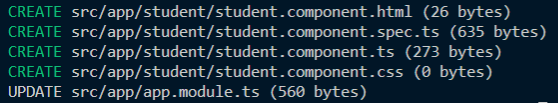


**Exercise #2 – Creating Component with Angular CLI**

1. Continue to work the same application in ***exercise-1*** folder.
2. Use the following ***Angular CLI*** command to generate a Component named student



The following boiler plate code files will be created for the Student component.

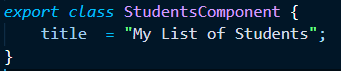


1. Navigate to the new Student folder and review the generated ***StudentComponent.ts*** file.

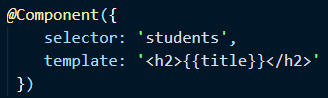


**Exercise #3 – Component Templates and First Binding**

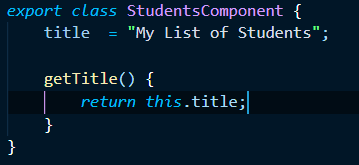
1. Continue to work the same application in ***exercise-1*** folder.
2. Navigate back to our original Component file ***StudentsComponent.ts***
3. Add a field in the ***StudentComponent*** class named title.



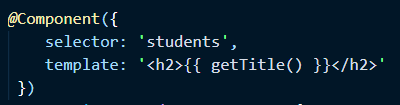
1. Update the ***Component Decorator*** to include a dynamic expression to update the view with the title.



1. Add a method named ***getTitle()*** to the Component as follows



1. Modify the Component Decorator to call function in the binding of the Template title

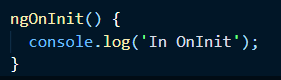


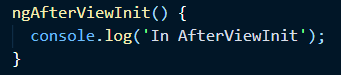
1. Create a new method named ***getCurrentDate()*** that return the current date in the ***StudentComponent.***
2. Update the Template Title to also display the Current Date from ***getCurrentDate().*** Use the ES2015 Template Literal syntax in the string html.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals>

**Home Work – Component Life Cycle Hooks**

1. In the same Component file ***StudentsComponent.ts,*** *implement some of the* ***Component Life Cycle Hooks*** [***https://angular.io/guide/lifecycle-hooks***](https://angular.io/guide/lifecycle-hooks)
2. In the OnInit method add the following logging code.



1. Add another Life Cycle event to the Import decorator  
   
2. Add the following code to implement the AfterViewInit interface.  
   
3. Add the Component selector element to the **app.component.html** file and run **ng serve**
4. Investigate where the console logging will occur. Server console or Client console ?