Angular 2 Lab

**Setup:**The projects is divided into steps.   
Every step has a starter and a solution.  
 Starters are at **step#.1** foldersandsolutions are at **step#.2** folders.  
**“Step 0 – Typescript”** is not related to the rest of the steps. All other steps are Related and Dependent on each other.  
The project is based on [angular webpack seed](https://github.com/angular/angular2-seed) so you can either start with the seed on your own or use this project.   
To run your project you can(relevant for the Angular section):

* Start your project in step1.1 and not using starters.
  + In the cmd, from the project’s root, run: “npm start”, this will run webpack-dev-server.
* If you want to run with one of the steps starters
  + In the cmd, from the project’s root, run: “npm run start\_[#]”, where [#] should be replaced with the step number you currently want to run.

# **Typescript & ES2015**

***Step 0 – Typescript:***

1. in this section you will need to make a typescript file pass compilation.
   1. Install typescript on your machine.
   2. run **“npm install typescript –g”**
   3. now you should be able to run the typescript compiler with the command **“tsc”.**
2. Go to the **step0.1\_typescript folder**.
3. Find the **main.ts** file and under the models folder find the **album.model.ts**
4. Your goal is to make the code in main.ts compile successfully
5. to compile the code simply run in command line the command **“tsc”** from the step0.1\_typescript folder
6. if compilation pass you should see a **dist** folder containing the compiled javascript files Feel free to browse the compiled files.

# **Angular** 2

**Step 1- Components:***In this step we will build our first component.  
The starter for this section is under* ***step1.1*** *folder.*

1. Go to the step1.1 folder
2. In the app.component, create a component class and decorator.
3. Add an html file as the view of the app.component
4. The view should include rendering an album data.
5. You should display the album cover imager, album name and artist name.

**Step 2- Data Binding & Directives***in this step we add data and data bindings to our app component.  
The starter for this section is under* ***step2.1*** *folder.*

1. Under the project’s root find the **albums.json** file located inside the data folder.
2. Create an Album model class compatible with the album objects inside the **albums.json.**
   1. Use the one you created in the typescript exercise. ( step0.1\_typescript )
   2. If you haven’t completed it, you can grab it from the solution (step0.2\_typescript)
3. Create a const variable of the album model Type inside the app.component, and hard code the albums.json or take just a few albums objects into it.  
   This const variable will currently act as our data source, in the next sections we will retrieve this data dynamically.
4. Your app.component should now render an albums list based on the albums.json.  
   Hint: use \*ngFor
   1. Every item in the list should display the album image and the album name.  
      the list item should use the image size medium.
   2. When clicking on a list item a new albums details view will open aside the list.  
      Hint: use \*ngIf to hide the details view when no item has been clicked.
   3. The album details will display the album image with “extralarge” size,album name, artist name, play count, and the url which should be a link navigating to this url.

**Step 3- Composition :**

*In this step we will separate our app into modules and components.  
The starter for this section is under* ***step3.1*** *folder.*

1. Separate the app component into 2 components. One for the albums list and one for the album details.
2. Create a separate module that will be imported by the App module, and declare the components inside this module.  
   Hint: don’t forget to export them as well
3. You should use the appropriate input and output properties for every component.

**Step 4- Services & http:**

*In this step we will create a service that will make http requests to fetch the Albums data from the server.  
The starter for this section is under* ***step4.1*** *folder.*

1. Go to the step 4.1 folder
2. Add resources folder into app folder
3. Add beatles-albums.json file into resources folder
4. Copy the JSON from the step 4.2/src/app/resources/beatles-albums.json file
5. Add album.ts file into playlist folder and create 3 classes ImageObject, Album, Artist

according to the hard coded album object you created in a previous step.

1. Add playlist.service.ts file in playlist folder, import Injectable, Http, Response, Observable modules and album class. Use albumsUrl = **'app/resources/beatles-albums.json'** and create a service to get the albums from JSON file.
2. Change the component to get albums from the service.

**Step 5- Pipes***in this step we will add a new pipe to our app.  
The starter for this section is under step5.1 folder.*

1. Create a new pipe called “epllipsis.pipe”
2. The pipe should a receive a number parameter to after how many characters it should replace with “…” .
3. Place the pipe on the Albums names.
4. Update the app.component to Bind the albums list directly to the observable returned from the service. Use the Async pipe.

**Step 6- Forms:***In this step we create a new contact form with simple validation.  
The starter for this section is under step6.1 folder.*

1. Go to the step 6.1 folder
2. Add contact folder
3. Add contact model class with name, email, content properties
4. Add contact component with html and CSS files
5. Use [(ngModel)] binding to bind the contact model and input elements in html.
6. Use [hidden] attribute to show/ hide validation errors, according to the mode.valid, model.pristine or another input element status.
7. Add validation rule as you want.

**Step 7- Routing:***In this step we will add routing to our app.  
The starter for this section is under step7.1 folder.*

1. Our app should have 3 routes
   1. ‘/albums’ pointing the albums list view.
   2. ‘/albums/:mbid’ showing the details of an album that was clicked on the list
   3. ‘/contact’ showing the contact form.
2. Update The albums list component so that every click on a list item will navigate to a new route with the “mbid” of the album as a url parameter.
3. Add a method in the albums service to fetch the a single album by mbid.
4. In album details component use the ActivatedRoute service to get the mbid from the url, call the album service and display the details of the selected album.