# Instructions – Assignment 4.4 – Async Pipe

composer-app, final version

**Instructions**

* ~~Make a copy of the reactive-composer-app from Exercise 4.3 and add it to your week-4 directory~~
* ~~Rename the application to enterprise-composer-app~~
* ~~Delete the node\_modules directory~~
* ~~Delete the package-lock.json file~~
* ~~Open the angular.json file and find and replace all “reactive-composer-app” entries with “enterprise-composer-app”~~
* ~~Open the package.json file and change the name to “enterprise-composer-app”~~
* ~~Run npm install and ng serve~~
  + ~~You are doing this to test the application and confirm there are no errors~~
* ~~app.component.html~~
  + ~~Change the assignment name to Assignment 4.4 - Async Pipe~~
* composer.service.ts
  + ~~Add an import statement for rxjs Observable object~~
    - ~~import { Observable } from ‘rxjs’;~~
  + ~~Add an import statement for the “of” operator from rxjs~~
    - ~~import { of } from ‘rxjs’;~~
  + ~~Update the return type of the getComposer function to an Observable array of IComposer objects~~
    - ~~getComposers(): Observable<IComposer[]>~~
  + ~~Update the code in the getComposers() function to return an observable array~~
    - ~~return of(this.composers)~~
* composer-list.component.ts
  + ~~Add an import statement for Observable~~
    - ~~import { Observable } from ‘rxjs’;~~
  + ~~Update the composers variable to type Observable<IComposer[]>~~
    - ~~composers: Observable<IComposer[]>~~
* ~~composer-list.html~~
  + ~~Add the async pipe to the tbody \*ngFor loop~~
    - ~~\*ngFor=”let composer of composers | async~~
* ~~Run and test the application~~
  + ~~Make sure you run and test the application for errors before proceeding to the remaining steps in this assignment~~
* composer.service.ts
  + ~~Add a new function called filterComposers(name: string) and set the return type of Observable<IComposer[]>~~
    - ~~filterComposers(name: string): Observable<IComposer[]>~~
  + ~~Add an import statement for the rxjs map operator~~
    - ~~import { map } from ‘rxjs/operators’;~~
  + ~~In the body of the function, return an Observable array of all Composer objects containing the parameter name in their fullName.  We will be using the pipe, map, and filter functions~~
    - ~~return of(this.composers).pipe(map(composers => composers.filter(composer => composer.fullName.toLowerCase().indexOf(name) > -1)))~~
    - ~~Note: the pipe() operator is a built-in function that allows us to chain functions together.  The map() function is used when we want to return a new array of objects.  And, the filter() function, as the name suggests, filters an array of data.  We are basically saying, “give me a new Observable array (map()) where the name parameter has characters in the composer.fullName (hince the indexOf) call.”~~
* ~~composer-list.component.ts~~
  + ~~In the body of the filterComposers() function replace the alert() box with a call to the composerService.filterComposers(name) function and assign the results to the composers variable~~
    - ~~this.composers = this.composerService.filterComposers(name);~~
* Run and test the application by entering names in the txtSearchControl
* Deploy the application to GitHub pages (use the deployment guide I provided under the weekly resources section)
* Update your personal portfolio with a link to the deployed website on GitHub pages
  + Clicking the link in your personal portfolio should take users to the composer-app running as a static website on GitHub pages

Note: the goal of this assignment is to demonstrate how to use Observables and build Reactive applications.  Reactive simply means “live” loading and reloading of data.  Think about applications like Slack, Facebook messenger, and Google Docs.  They are reactive, because they respond immediately to user input and those changes are applied in real-time or near-real-time.