coursebrew

Front End Developer's Guide

coursebrew's front end is separated into components. Each component consists of:

1. An HTML file
   The HTML file is what holds the main markup for the UI. Angular HTML files can display variables from the typescript file by encapsulating code in double curly braces -> `{{myTypeScriptObject}}`
2. A Typescript file
   The typescript file holds all of the functions that the component can run, anything that requires any sort of logic should go here.
3. A CSS file
   The CSS file holds all of the styling information

Components are able to share information with each other and with the back end server through the use of services. A service is injected into a component's Typescript file. When injecting a service, make sure to include it in the imports, the constructors, and to create the service object.

Let's look at a full example of how this works.

So on our add instructor page, in the HTML code located at **add-instructor.html** for the ADD button we see this:

```
<button (click)="submitInstructor()" type="add"
data-toggle="modal" data-target="#popUp" class="btn col-5
add-button">ADD</button>
```

The `(click)="submitInstructor()"` binds our button to a Typescript function, which is located in **add-instructor.component.ts**

The correlating function's code is this:

```
  submitInstructor() {
     this.newInstructor.wl_credits =
(parseInt(this.newInstructor.wl_courses) * 3).toString()
     this.newInstructor.year = this.scope.year;
     this.serverMessageSub =
this.instructorsApi.addInstructor(this.newInstructor)
     .subscribe(res => { this.serverMessage = res;
     },
       error => alert(error.message)
     );
  };
```

Once this code runs, the component uses a service to send the client side object to the server, using the code that begins at `this.instructorsApi.addinstructor`. The instructorsApi object is a service object, that implements an HTTP Restful call to the server, which is located in **instructors-api.service.ts**

```
addInstructor(instructor: Instructor): Observable<any>{
      return this.http
         .post(this.appGlobal.baseAppUrl +
this.appGlobal.basePort + "/add_instructor",
JSON.stringify(instructor))
}
```

The service will launch the call and return the server payload in a JSONified format.