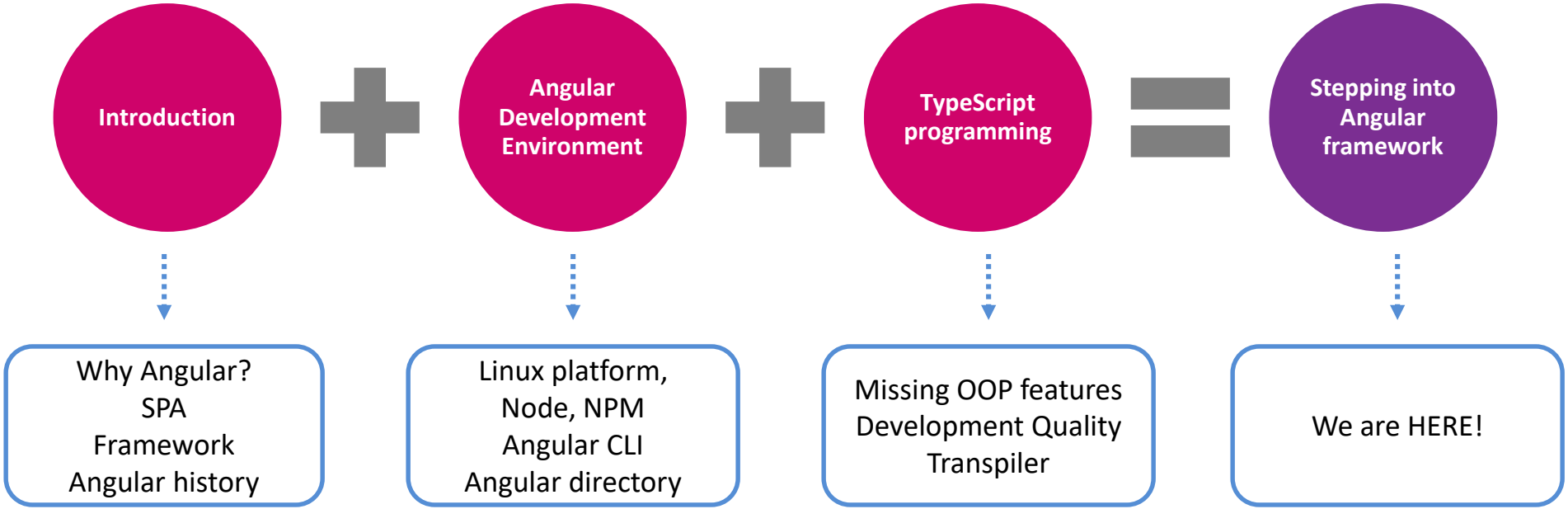
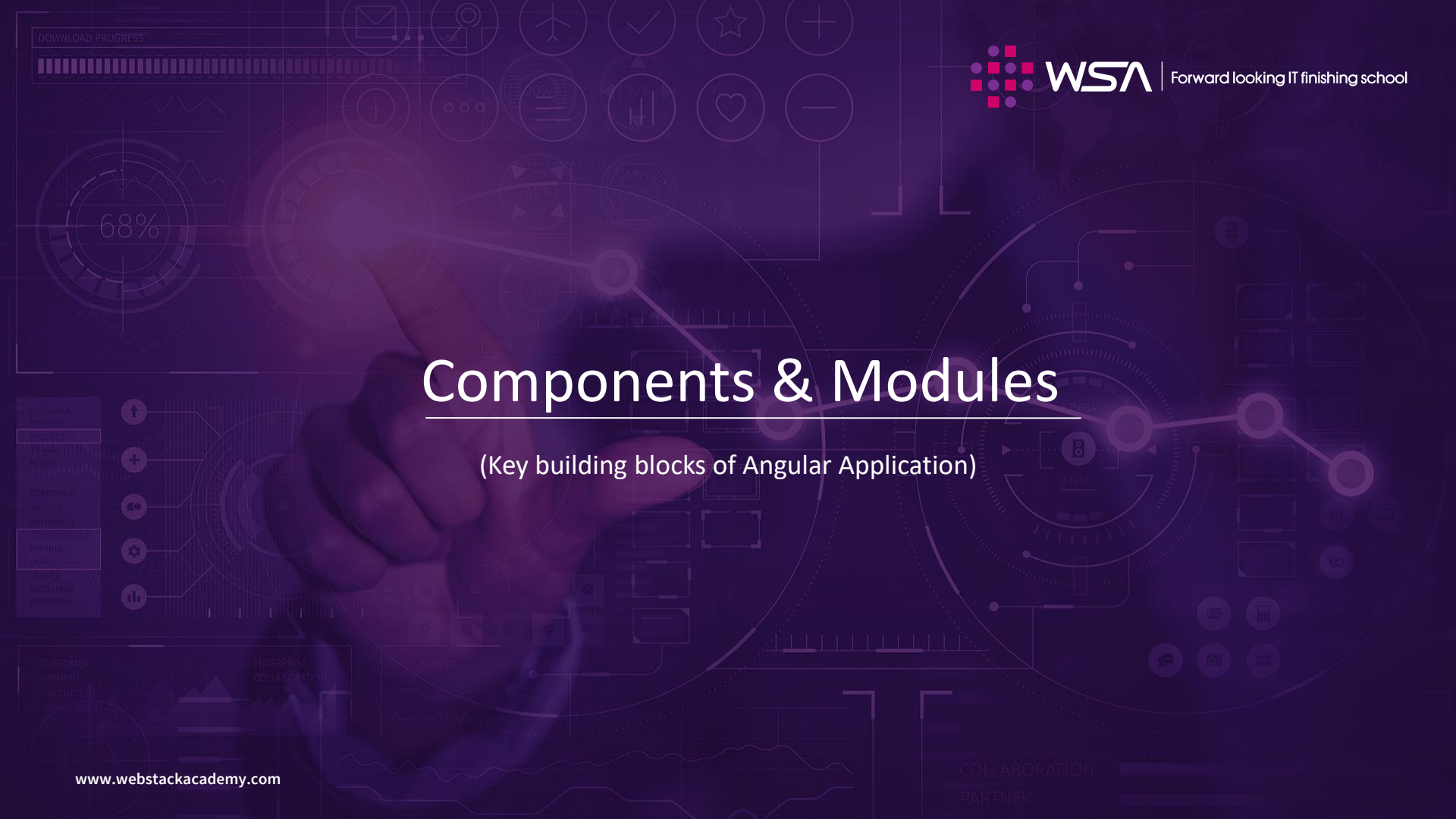
WSA | Forward looking IT finishing school

Components
Angular


WSA
Forward looking IT finishing school
Connecting the dots
(A Quick looking-back!)
www.webstackacademy.com

What we have done so far?

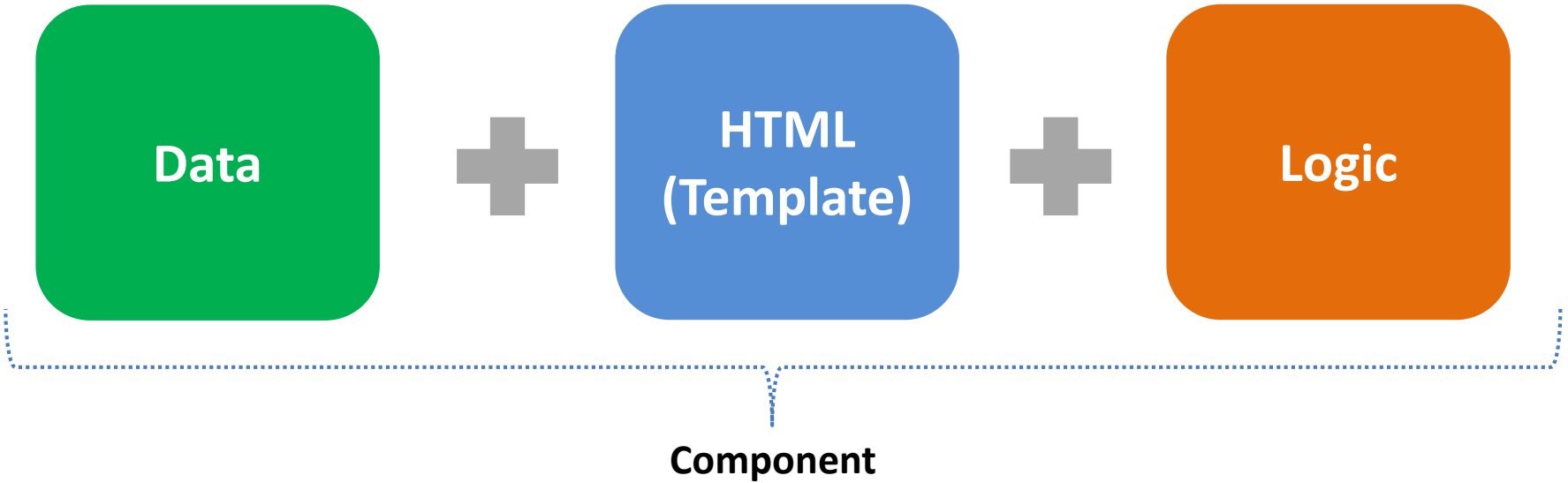



WSA
Forward looking IT finishing school
Components & Modules
(Key building blocks of Angular Application)
www.webstackacademy.com

What is a component?

- A component is anything visible to end user and can be reused many times within an app
- It controls a patch of screen called a view
- Component can intern can have multiple sub-components
- All components will have a beginning called root-component. From there components will further have tree structure in a modular fashion

What does the component consist of?

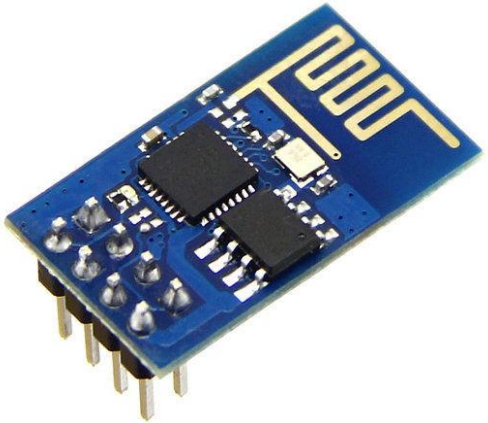


Why Component? - Benefits

- **Modular:** It makes the application more modular, in alignment with framework
- **Data encapsulation:** Data access via methods
- **Re-usable:** Makes the development more efficient

Analogy: Individual electronic (RLC) components

What is a Module?



- Module is a container for a group related components, one level above the component
- It also contains other parts of an app: **Services, Directives, Pipes** (will see them one-by-one!)
- Every Angular app has at least one module, called root module. Root module is conventionally named **AppModule**
- Most apps have more than one modules (Root module + Individual Feature modules)

Analogy: Individual electronic components, mounted together to give a meaningful functionality (ex: WiFi, SSD etc…) called as "module"


WSA
Forward looking IT finishing school
Creating a Component - GUI
(Hooking your first piece of code into Angular)
www.webstackacademy.com

Creating a component – GUI mode

- **Step-1: Creating / Adding your component TS file**
- Open your Angular project folder (ex: myFirstApp) in your editor (ex: VS code). Go to `src/app` folder.
- Add a new file in the following format: <component_name>.component.ts (ex: `courses.component.ts`)
- Add the following lines into your new component file. Explanation are given as comments.

```
// Decorator class
import { Component } from '@angular/core'

// Selector is a custom HTML tag. Template has the HTML code
@Component({
selector: 'courses',
template: '<h2>My list of courses</h2>'
})

// Main class of component, you can define attributes and methods here...
export class CoursesComponent {

}
```

Creating a component – GUI mode

- **Step-2:** Make your component as a part of the module (app.module.ts file)
  - 2.1 Make your component available for Module by importing it into the module

```
// Add your new component here by importing it
import { CoursesComponent } from './courses.component';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
```

Creating a component – GUI mode

- **Step-2:** **Make your component as a part of the module (app.module.ts file)**
  - **Add your new component (ex: CoursesComponent) under the declarations sections of the module.**
  - **Please note the modules contain more than components (ex: bootstrap), which we will study in further chapters**

```
@NgModule({
declarations: [
AppComponent,
CoursesComponent
],
imports: [
BrowserModule,
],
providers: [],
bootstrap: [AppComponent]
})
```

Creating a component – GUI mode

Step-3: Go-to **`app.component.html`** file and add your selector (ex: <courses> into this file). U can add any HTML code here.

Also ensure you delete previously existing HTML code to ensure the file looks cleaner and easier to read.

Step-4: Save all files and run your Angular App again (ng serve --open).
U should be able to see your custom selectors displaying the corresponding template code in the browser.


WSA
Forward looking IT finishing school
Creating a Component - CLI
(Hooking your first piece of code into Angular)
www.webstackacademy.com

Creating a component – CLI mode

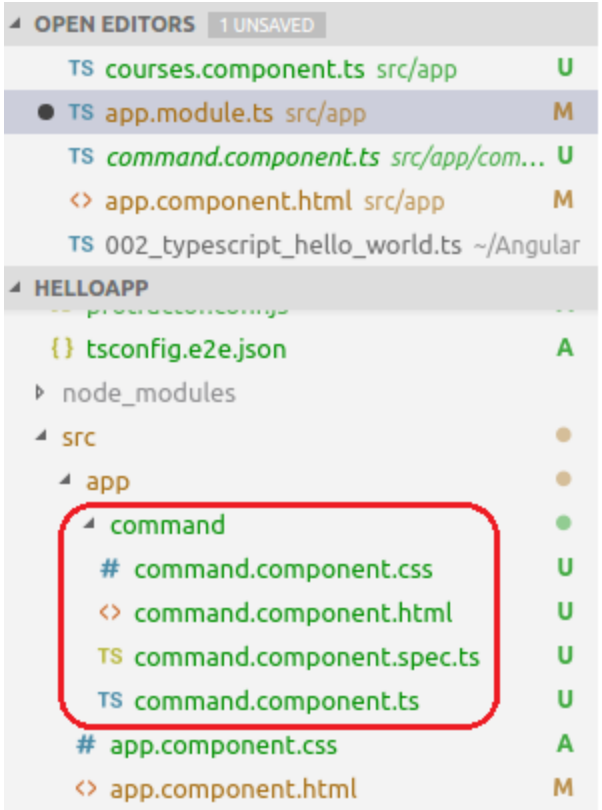
- Sometimes you might find it little hard to create a component using GUI. In case you miss out any steps, the app will not work.

- Angular CLI offers much more reliable way to create a component. Try out the following command, which will not only generate a component but also make appropriate entries

```
$ ng g c command      // Creates a new component - command
```

```
wsa@wsa-VirtualBox:~/helloApp$ ng g c command
CREATE src/app/command/command.component.css (0 bytes)
CREATE src/app/command/command.component.html (26 bytes)
CREATE src/app/command/command.component.spec.ts (635 bytes)
CREATE src/app/command/command.component.ts (273 bytes)
UPDATE src/app/app.module.ts (488 bytes)
wsa@wsa-VirtualBox:~/helloApp$
```

Creating a component – CLI mode




```
1  import { CommandComponent } from './command/command.component';
2  // Add your new component here by importing it
3  import { CoursesComponent } from './courses.component';
4  import { BrowserModule } from '@angular/platform-browser';
5  import { NgModule } from '@angular/core';
6
7  import { AppComponent } from './app.component';
8
9  @NgModule({
10   declarations: [
11     AppComponent,
12     CommandComponent,
13     CoursesComponent
14
15   ],
16   imports: [
17     BrowserModule,
18
19   ],
20   providers: [],
21   bootstrap: [AppComponent]
```

Creating a component – CLI mode

- Now you may see in your newly generated component, you have three files (HTML / CSS / TS), the three important files to run a typical front end application
- Any changes you can make into the HTML file and include corresponding selector into `app.component.html` to make this active
- By having multiple components mean there are multiple pieces coming together and working in an app. This is just a beginning, so many things we can do on top of this.


WSA
Forward looking IT finishing school
Services
(De-coupling view and its functionality)
www.webstackacademy.com

What is a Service?

- In real-time web application, the component should display (present) the content using the template (HTML)
- The content gets fetched from a server and make it available for a component, which should be de-coupled from the component
- Only presentation is the component's responsibility, it should NOT be dealing with the data
- Hence data handling need to be delegated to a somebody, which is called as **'Service'** in Angular. The service primarily deals with an HTTP end-point (ex: RESTful interfaces / REST APIs) from the server
- To demonstrate HTTP data fetching requires a service (which we will do in the Node.js integration topic later). In the current topic let us create a service with a fake HTTP endpoint

What is a Service? - Analogy

Analogy: Components & Modules only makes sense when a meaningful functionality is achieved (ex: SSD display).


WSA
Forward looking IT finishing school
Creating a Service - GUI
(Giving some meaningful functionality for your component)
www.webstackacademy.com

Creating a service – GUI mode

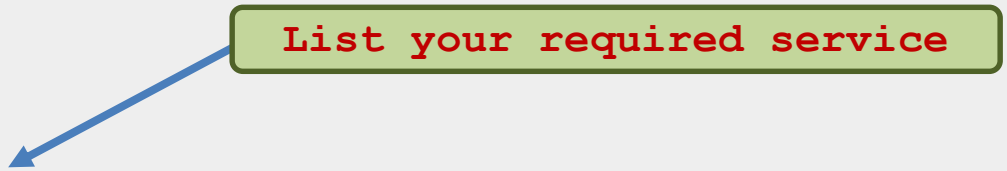
- **Step-1: Creating / Adding your service TS file**
  - Go to your `src/app` folder.
  - Add a new file in the following format: <component_name>.service.ts (ex: `courses.service.ts`)
  - Add the following lines into your new service. Explanation are given as comments
  - Unlike component, no decorator class is required here. This is a plain TypeScript class

```
export class CoursesService {

getCourseList()
{
   // Create an array for courses and return
   var courseList = ["FullStack dev", "FrontEnd dev" , "Backend dev"];
   return courseList;
}

}
```

Creating a service – GUI mode

- **Step-2:** **List this new services as a dependency for my component (app.module.ts file)**

```
import { CoursesService } from './courses.service';

@NgModule({
declarations: [
    AppComponent,
    CommandComponent,
    CoursesComponent
],
imports: [
    BrowserModule,
],
providers: [
    CoursesService
],
bootstrap: [AppComponent]
})
```



Creating a service – GUI mode

- **Step-3:** Make changes in your component file (courses.component.ts)

```
// Import a service
import { CoursesService } from './courses.service';

@Component({
    selector: 'courses',
    template: `
        <h2> {{title}}</h2>
        <ul> {{courses}}</ul>
    `
})

export class CoursesComponent {
    title = 'List of courses in WSA:';
    courses;
    constructor (service: CoursesService)
    {
        this.courses = service.getCourseList();
    }
}
```

These changes can be done in the HTML file pointed by templateURL as well


WSA
Forward looking IT finishing school
Creating a Service - CLI
(Using CLI to generate your services)
www.webstackacademy.com

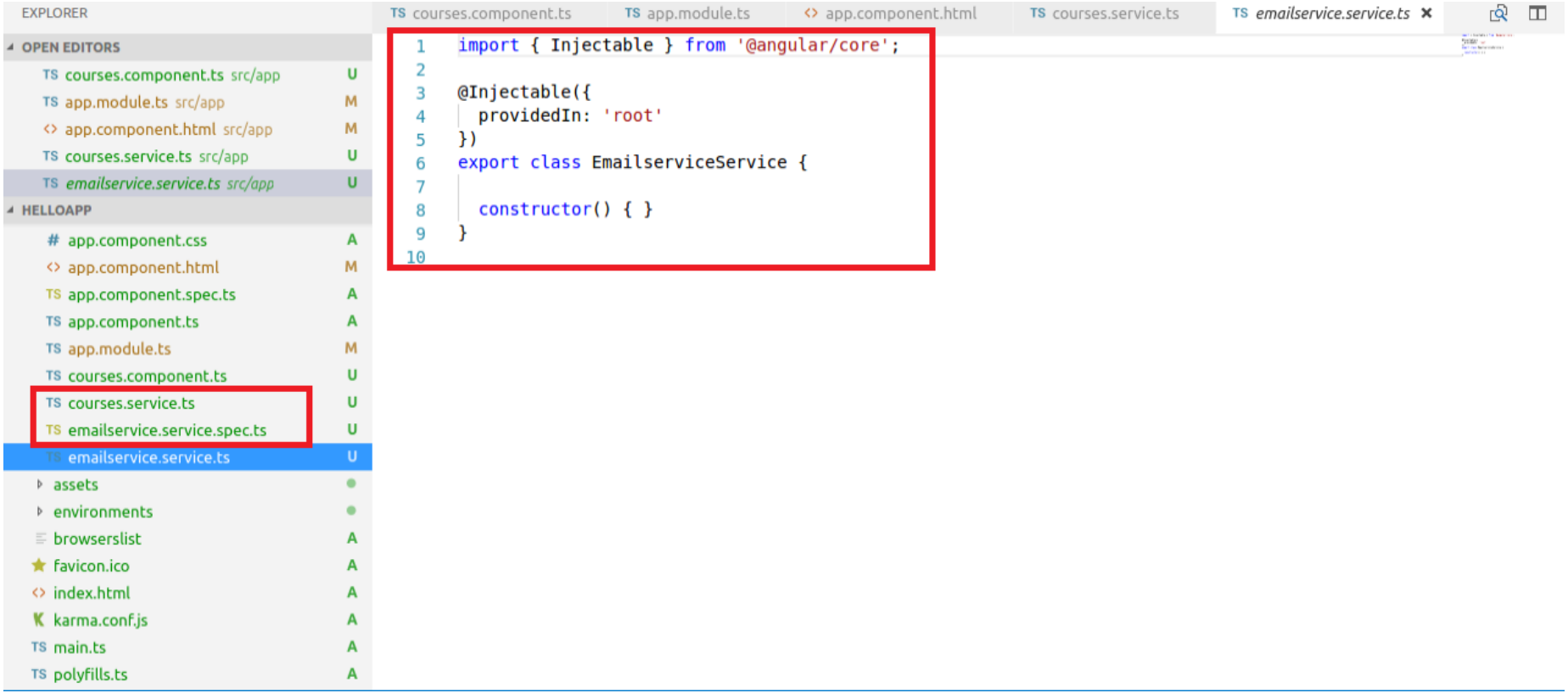
Creating a service – CLI mode

Similar to component, services can be added to a component using CLI. Along with generation it will make appropriate entries into other files

```
$ ng g s emailservice                    // Creates a new service
```

```
wsa@wsa-VirtualBox:~/helloApp$ ng g s emailservice
CREATE src/app/emailservice.service.spec.ts (410 bytes)
CREATE src/app/emailservice.service.ts (141 bytes)
wsa@wsa-VirtualBox:~/helloApp$
```

Creating a component – CLI mode



Dependency Injection

- Dependency Injection (DI, one of the **application design pattern**) is a way to create objects that depend upon other objects.
- One object supplies the dependencies of another object. A dependency is an object that can be used (a service).
- An **injection** is the passing of a dependency to a dependent object that would use it, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern.
- This allows the component to make acquiring dependencies someone else's problem (in our case it is the Angular framework).
- The intent behind dependency injection is to **decouple objects** to the extent that no client code has to be changed simply because an object it depends on needs to be changed to a different one.

Exercise

- Assume you are developing a simple e-commerce application. Which will fetch following information from the server and display them in a **tabular format** in the browser:
  - Item name (Pen, iPhone 10, Levis jean)
  - Item category (Stationary, Electronics, Apparels)
  - Item price (100, 100000, 1500)
- Implement the above functionality using Angular with following way:
  - Create a new project
  - Create a new component & service (CLI usage is recommended)
  - The service should have three methods:
    - getItemName()
    - getItemCategory()
    - getItemPrice()
  - All methods to simulate a fake HTTP endpoint (by returning static arrays)
  - Assume the number of items are fixed (later we can make it dynamic)

WSA | Forward looking IT finishing school

WebStack Academy

#83, Farah Towers,
1st Floor, MG Road,
Bangalore – 560001

M: +91-809 555 7332
E: training@webstackacademy.com

WSA in Social Media:

www.webstackacademy.com