04

Angular Modules and Components

WHAT YOU WILL LEARN

Contents

Typescript Module

JavaScript has a concept of modules from **ECMAScript 2015**. TypeScript shares this concept of a module.

A module is a way to create a group of related variables, functions, classes, and interfaces, etc. It executes in the **local scope**, not in the **global scope**.

n other words, the variables, functions, classes, and interfaces declared in a module cannot be accessible outside the module directly. We can create a module by using the **export** keyword and can use in other modules by using the **import** keyword.

Modules import another module by using a **module loader**. At runtime, the module loader is responsible for locating and executing all dependencies of a module before executing it. The most common modules loaders which are used in JavaScript are the **CommonJS** module loader for **Node.js** and **require.js** for Web applications.

-> In Large applications we will have multiple classes
-> It is highly recommended to write each class in a separate file.
-> To access the class of one file in another file we will use Modules concept in TypeScript.
-> Module is a file (ts file) which can export one or more classes to other files.
-> To export a class we will use 'export' keyword (source file)
-> To import a class we will use 'import' keyword (destination file)

```
export class Student {

    studentId : number;
    studentName : string;

    constructor(id:number, name:string){
                this.studentId = id;
            this.studentName = name;
    }
}
```

----------School.ts----------

```
import {Student} from "./Student" ;

class School{
```

```
        students : Student[] = [
                new Student (101, "John"),
              new Student(102, "Smith"),
              new Student(103, "Nick")
            ];

         display() : void {
                for(var i in this.students){
                        console.log(this.students[i]);
              }
            }
}

let school:School = new School();
school.display();
```

1. Internal Module

Internal modules were in the **earlier version** of Typescript. It was used for **logical grouping** of the classes, interfaces, functions, variables into a single unit and can be exported in another module. The modules are named as a **namespace** in the latest version of TypeScript. So today, internal modules are **obsolete**. But they are still supported by using namespace over internal modules.

Internal Module Syntax in Earlier Version:

```
module Sum {
  export function add(a, b) {
    console.log("Sum: " +(a+b));
  }
}
```

Internal Module Syntax from ECMAScript 2015:

```
namespace Sum {
  export function add(a, b) {
    console.log("Sum: " +(a+b));
  }
}
```

2. External Module

External modules are also known as a **module**. When the applications consisting of hundreds of files, then it is almost impossible to handle these files without a modular approach. External Module is used to specify the **load dependencies** between the multiple external js files. If the application has only one js file, the external module is not relevant. ECMAScript 2015(ES6) module system treats every file as a module.

Module declaration

We can declare a module by using the **export** keyword. The syntax for the module declaration is given below.

```
//FileName : EmployeeInterface.ts
export interface Employee {
  //code declarations
}
```

We can use the declare module in other files by using an **import** keyword, which looks like below. The **file/module** name is specified without an **extension**.

```
import { class/interface name } from 'module_name';
```

Angular built-in module and custom module

1. Angular is a modular-based architecture
   - There are lot of modules which are built-in
   - Examples
     - BrowserModule
     - BrowserAnimationsModule

   - Angular Material Library
     - MatButtonModule
     - MatDropDownModule

2. All the code and functionality is grouped in a module
3. Whenever you see a @ symbol - it's a decorator
4. What modules consist
   - declarations
     - this is where we will add all the components of the module
   - imports
     - we can import modules inside a module

   - providers
     - services that we need will be injected here

   - Bootstrap
     - what is the first component, the module should load

   - exports
     - is to export and expose the component outside of the module
5. Every Angular application should have atleast 1 module
6. By default, the Angular framework provides us with AppModule
7. The AppModule will have a component by the name
   - AppComponent

Components

A Component is nothing but **a simple typescript class, where you can create your own methods and properties as per your requirement which is used to bind with an UI (html or css page) of our application**.

2. Authentication Module

- new-user
- login
- forgot-password
- reset-password

3. Component is a smallest functionality that you will implement in your application

4. When we group multiple Components it becomes a module

5. We can have paren-child relationship of components

6. We can have components inside components

7. Tree-herirachy of components

Dashboard

display-contact-list

contact-grid

contact-import

contact-export

contact-options

8. lets create some custom components

ng g component add-contact

ng g component edit-contact

ng g component list-contacts

ng g component delete-contact

9. Every component has 4 files auto-generated with it

- component.html -> view or html or template file -> UI
- component.ts -> it will be a class file which will have methods -> Logic
- component.spec.ts -> It will have the unit test script for component
- component.scss -> stylesheet of the component

HOMEWORK

-> verify all the components generated and go through the code

10. Component decorator inside the component.ts file

selector -> unique identifier for the component

-> id of the component

-> using this selector we will use the component

templateUrl -> your HTML code

- component.html file

styleURLS -> for linking your component stylesheet

- component.scss

All The Best!!!