

Angular Fundamentals

In this section...

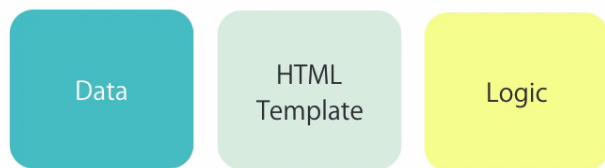
- Components
- Templates
- Directives
- Services

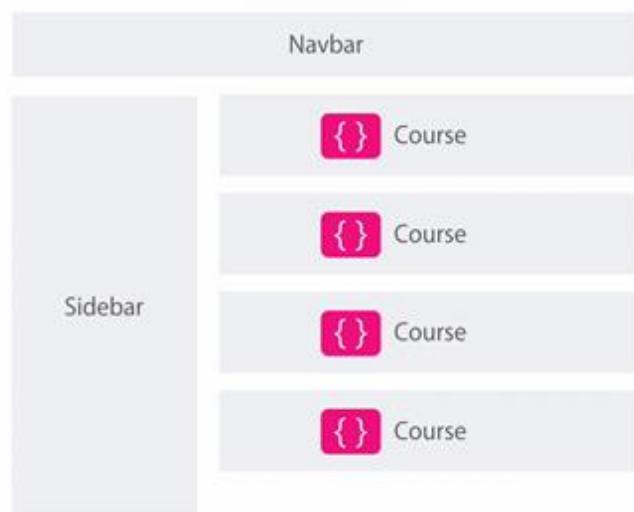
Building Blocks of Angular Apps

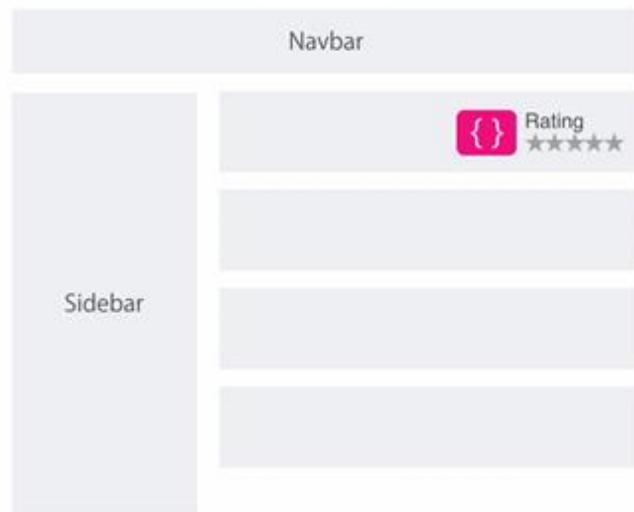
Building Blocks of Angular Apps

Components

Component

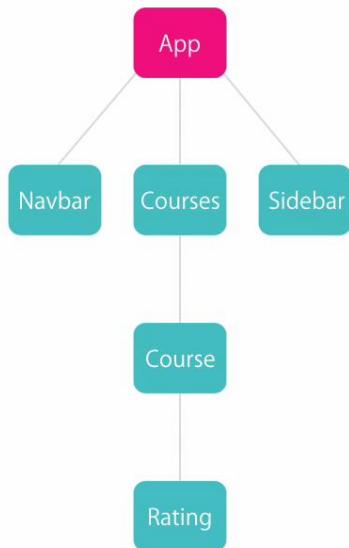






Component





Modules

Modules

Courses

Messaging

Instructor

Admin

App

M2

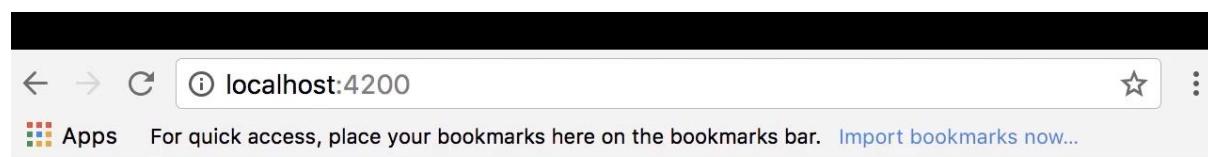
M3

Components

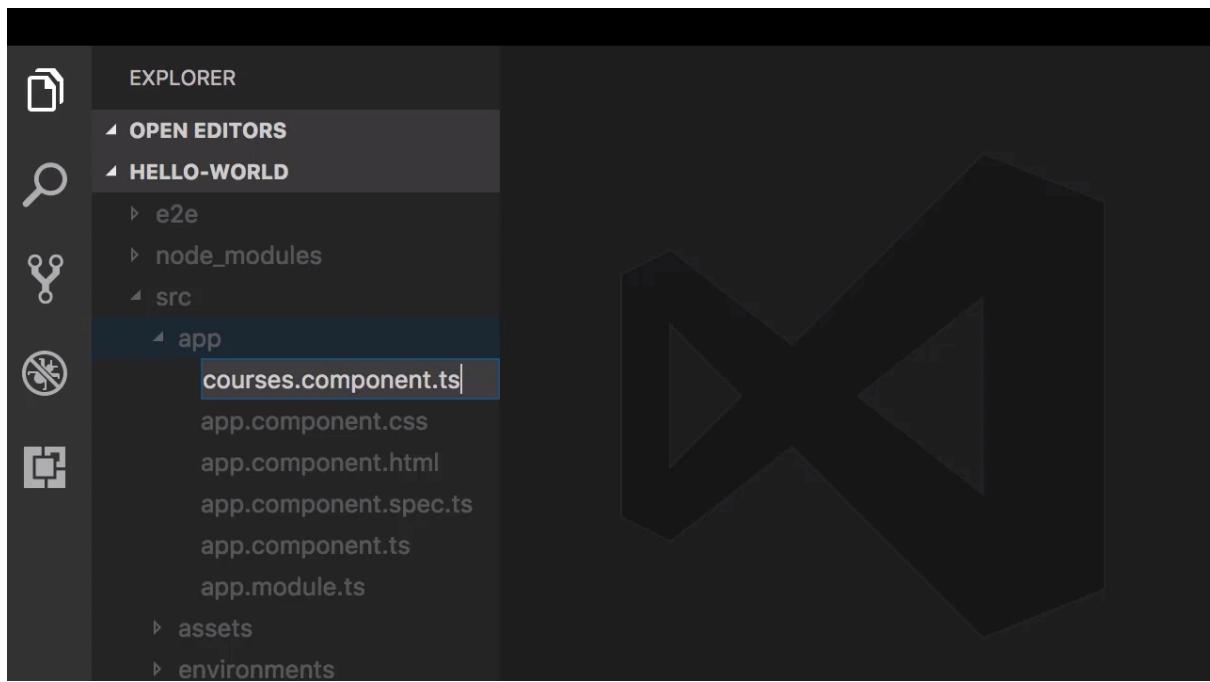
Creating Components

- **Create** a component
- **Register** it in a module
- Add an element in an **HTML markup**

```
hello-world $  
hello-world $ng serve  
** NG Live Development Server is listening on localhost:4200, open your  
browser on http://localhost:4200 **  
Hash: 90d90e5e77621daaaecf  
Time: 8196ms  
chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 1  
58 kB {4} [initial] [rendered]  
chunk {1} main.bundle.js, main.bundle.js.map (main) 5.29 kB {3} [ini  
tial] [rendered]  
chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 10.5 kB {4  
} [initial] [rendered]  
chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.11 MB [i  
nitial] [rendered]  
chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [e  
ntry] [rendered]  
webpack: Compiled successfully.
```

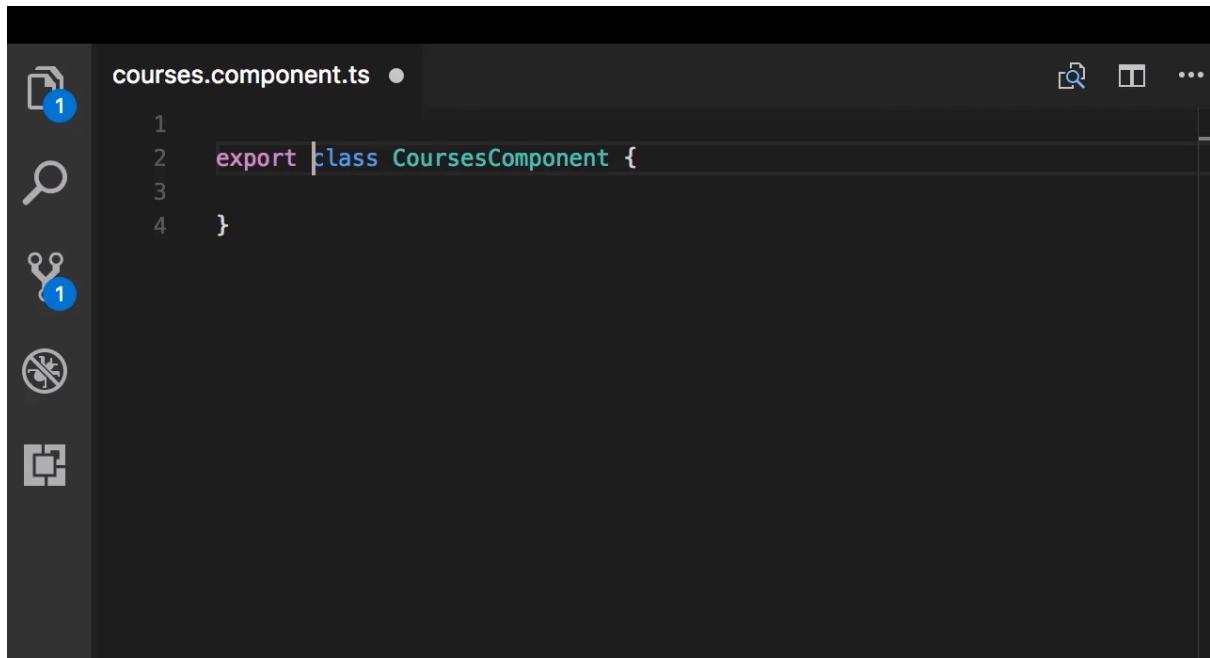


Welcome to app!!



```
courses.component.ts ●
```

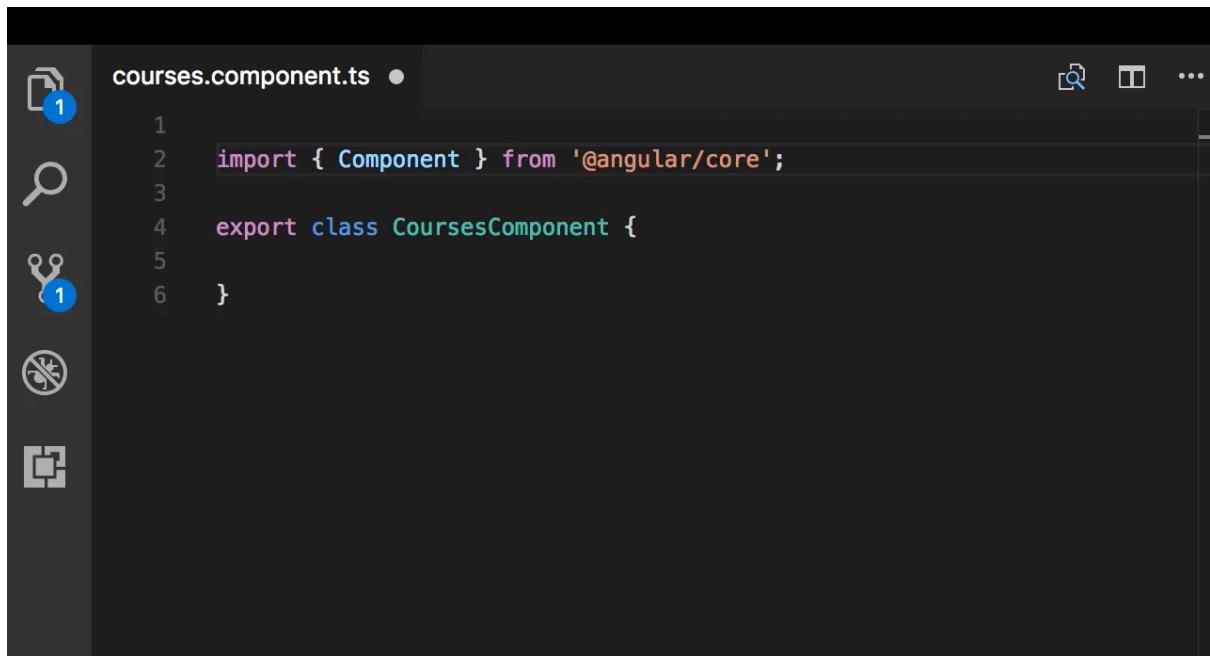
```
1
2  class CoursesComponent {
3
4 }
```



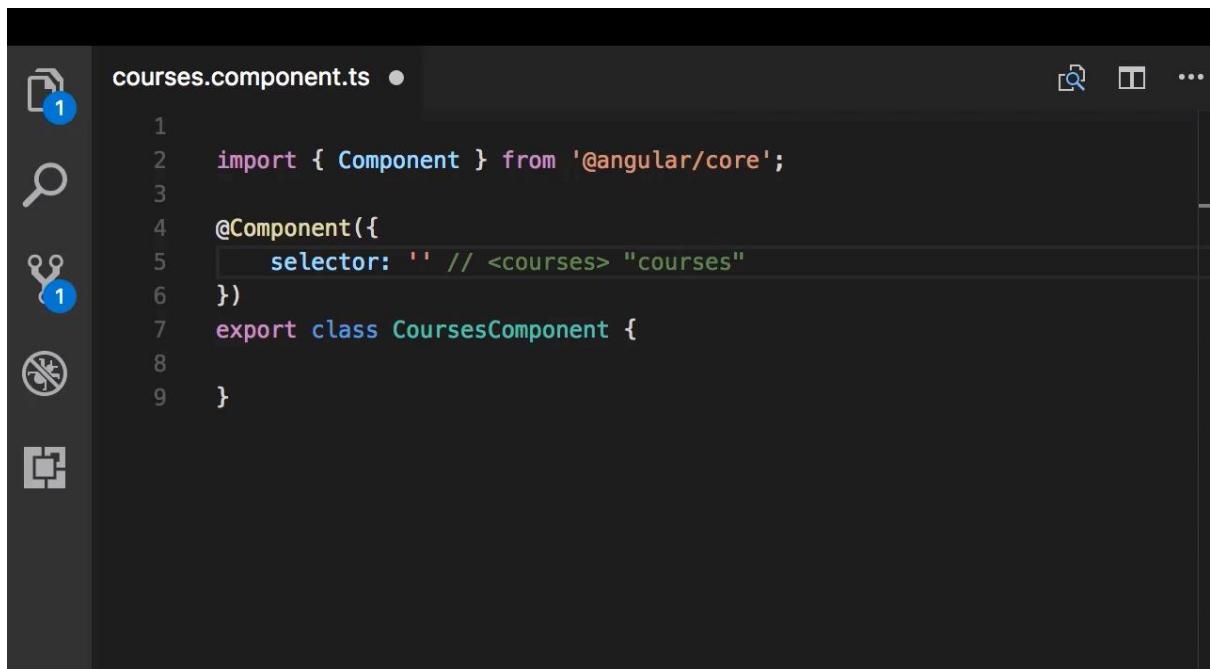
```
courses.component.ts ●
1
2  export class CoursesComponent {
3
4 }
```

Component decorator makes a component visible to angular:

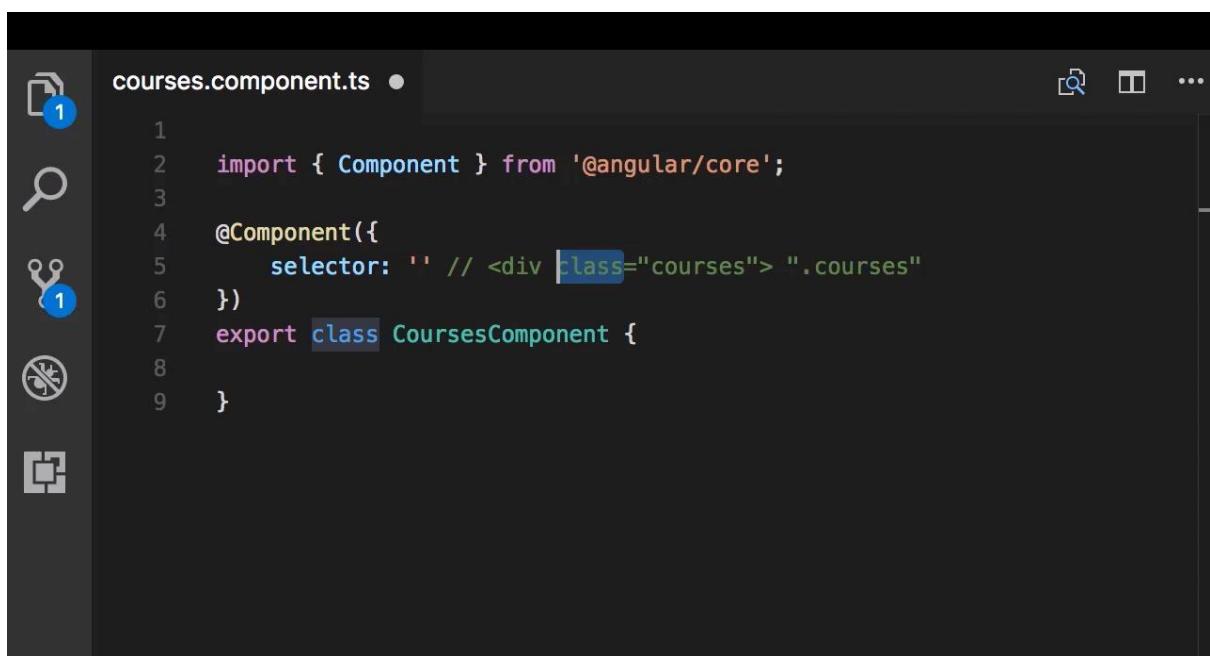
So import it and apply it



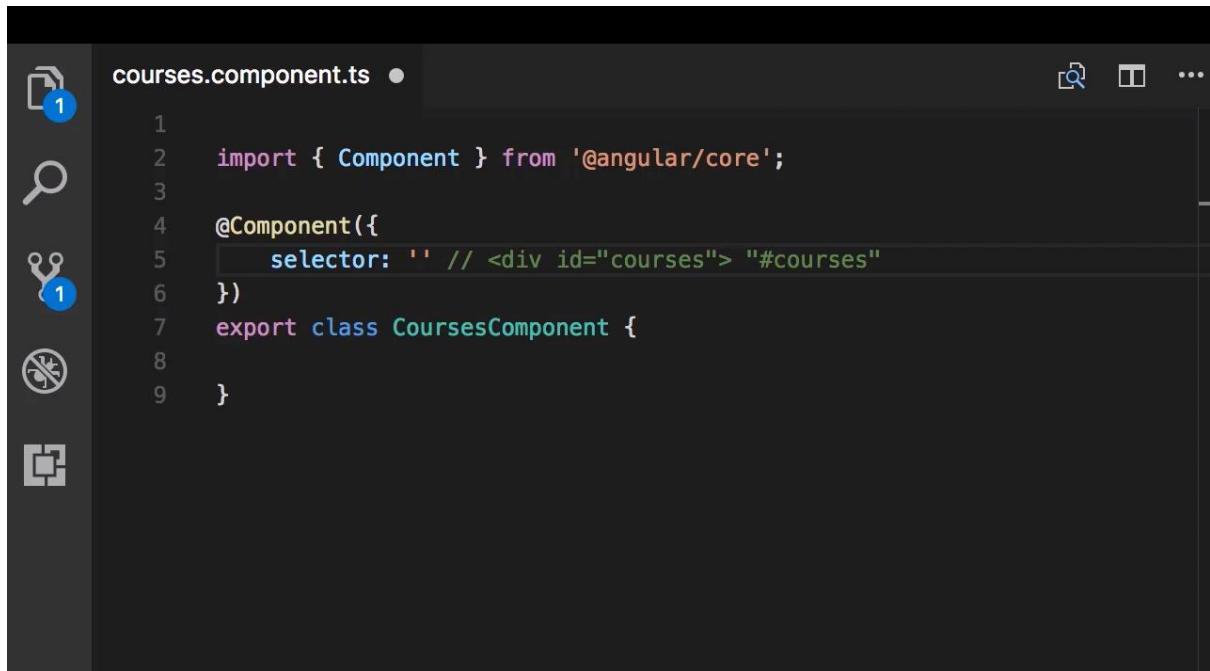
```
courses.component.ts ●
1
2  import { Component } from '@angular/core';
3
4  export class CoursesComponent {
5
6 }
```



A screenshot of a code editor interface showing a file named "courses.component.ts". The code contains a component definition:`1 import { Component } from '@angular/core';
2
3 @Component({
4 selector: '' // <courses> "courses"
5 })
6 export class CoursesComponent {
7
8 }
9`The editor has a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other tools. A blue circular badge with the number "1" is visible next to the file icon. The status bar at the top right shows icons for search, refresh, and more.

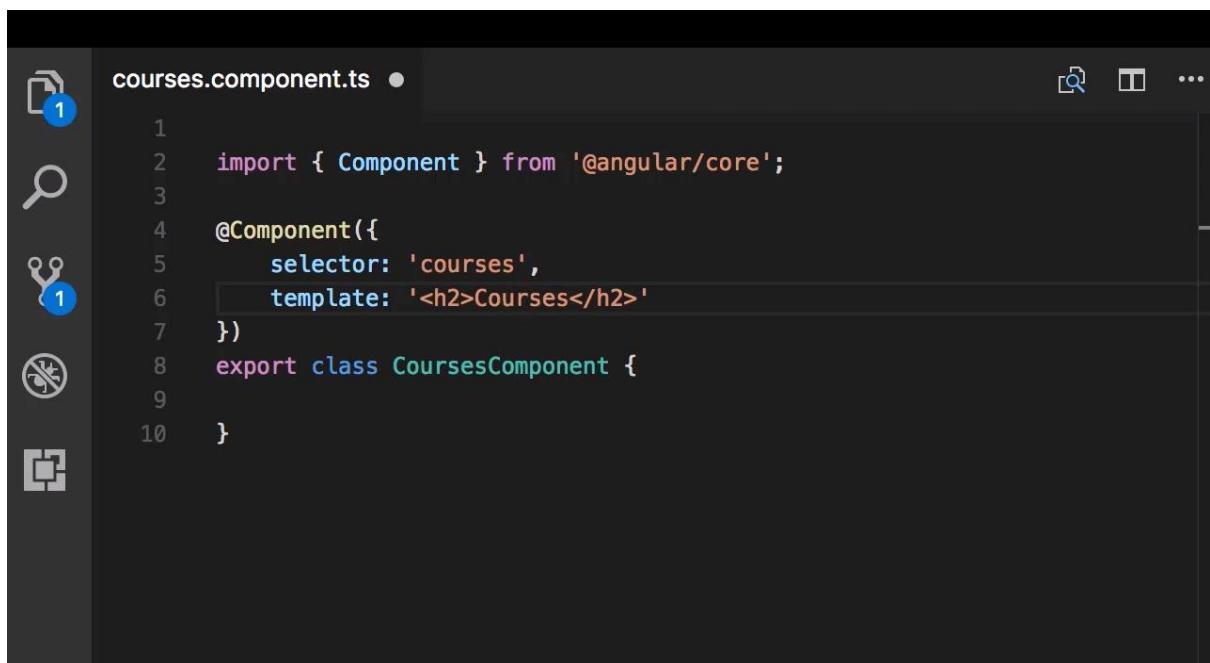


A screenshot of a code editor interface showing the same "courses.component.ts" file. The code is identical to the first screenshot:`1 import { Component } from '@angular/core';
2
3 @Component({
4 selector: '' // <div class="courses"> ".courses"
5 })
6 export class CoursesComponent {
7
8 }
9`However, the editor now displays multiple error icons (blue circles with exclamation marks) on the far left of the code editor area, indicating multiple syntax errors. The rest of the interface, including the toolbar and status bar, remains the same.



```
courses.component.ts ●

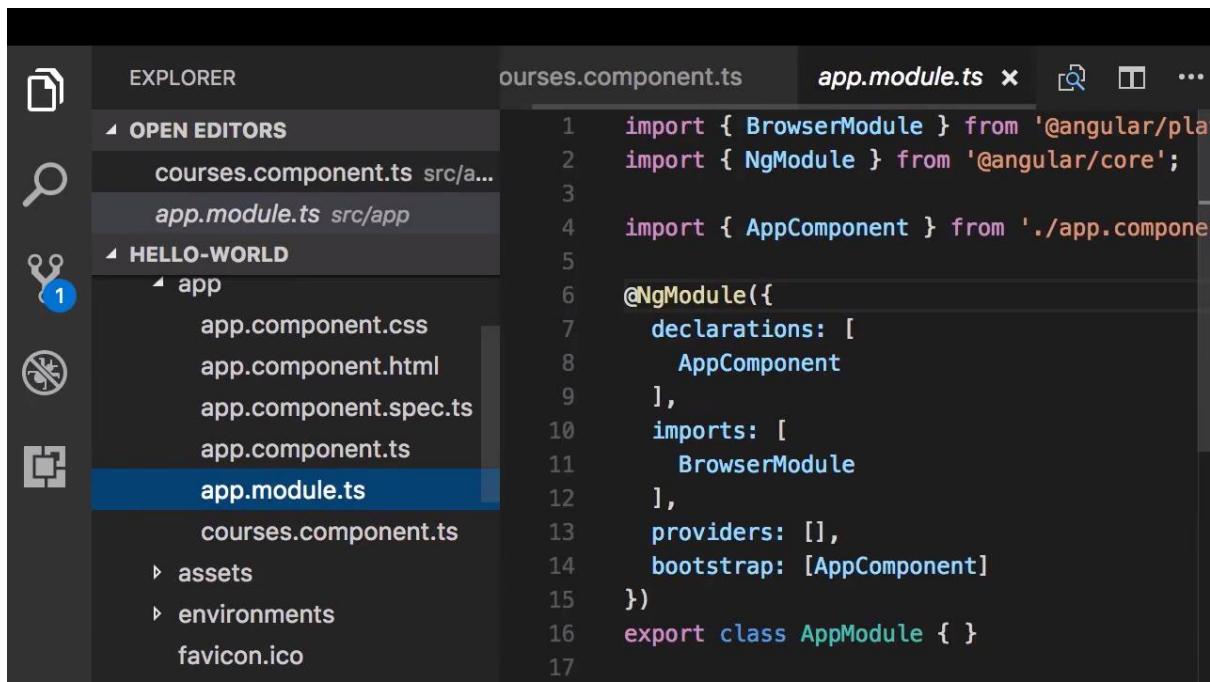
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: '' // <div id="courses"> "#courses"
5 })
6 export class CoursesComponent {
7
8
9 }
```



```
courses.component.ts ●

1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>Courses</h2>'
6 })
7 export class CoursesComponent {
8
9
10 }
```

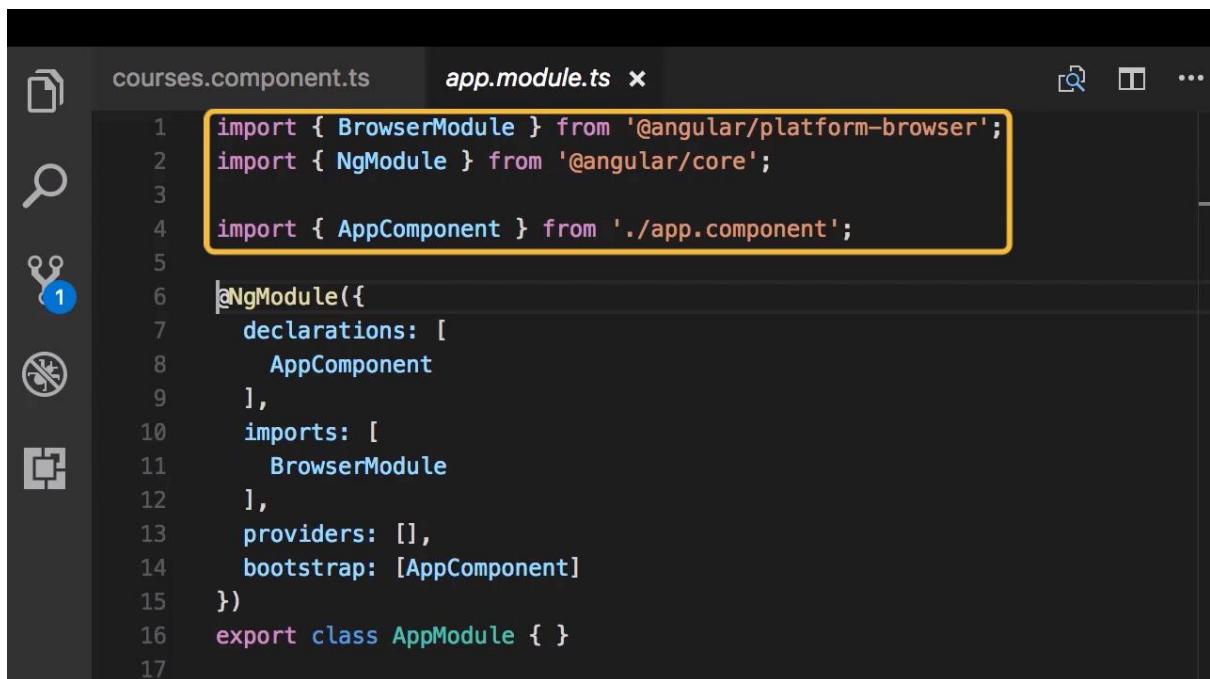
The next step is to register the component



The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar on the left showing the project structure:
 - OPEN EDITORS**: courses.component.ts, app.module.ts
 - HELLO-WORLD**:
 - app: app.component.css, app.component.html, app.component.spec.ts, app.component.ts, app.module.ts (selected)
 - assets
 - environments
 - favicon.ico
- Editor Area**: The file `app.module.ts` is open and displayed:

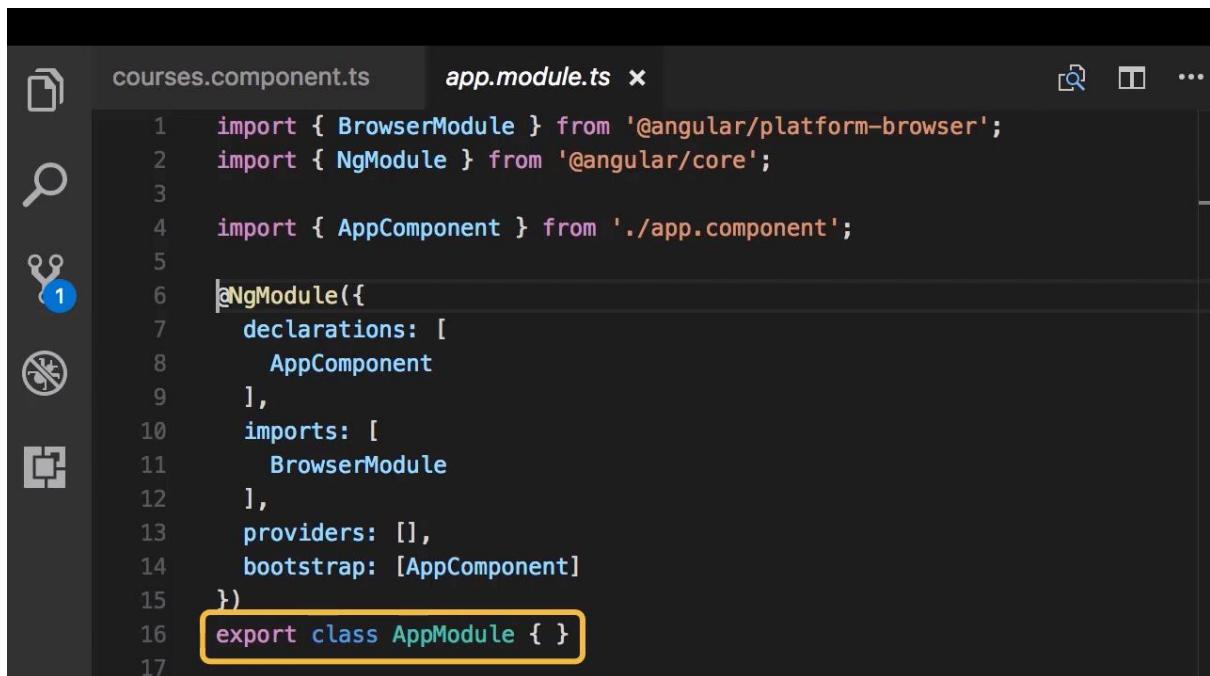
```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 
4 import { AppComponent } from './app.component';
5 
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10  imports: [
11    BrowserModule
12  ],
13  providers: [],
14  bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```



The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar on the left showing the project structure.
- Editor Area**: The file `app.module.ts` is open and displayed, with the import statement for `AppComponent` highlighted by a yellow box:

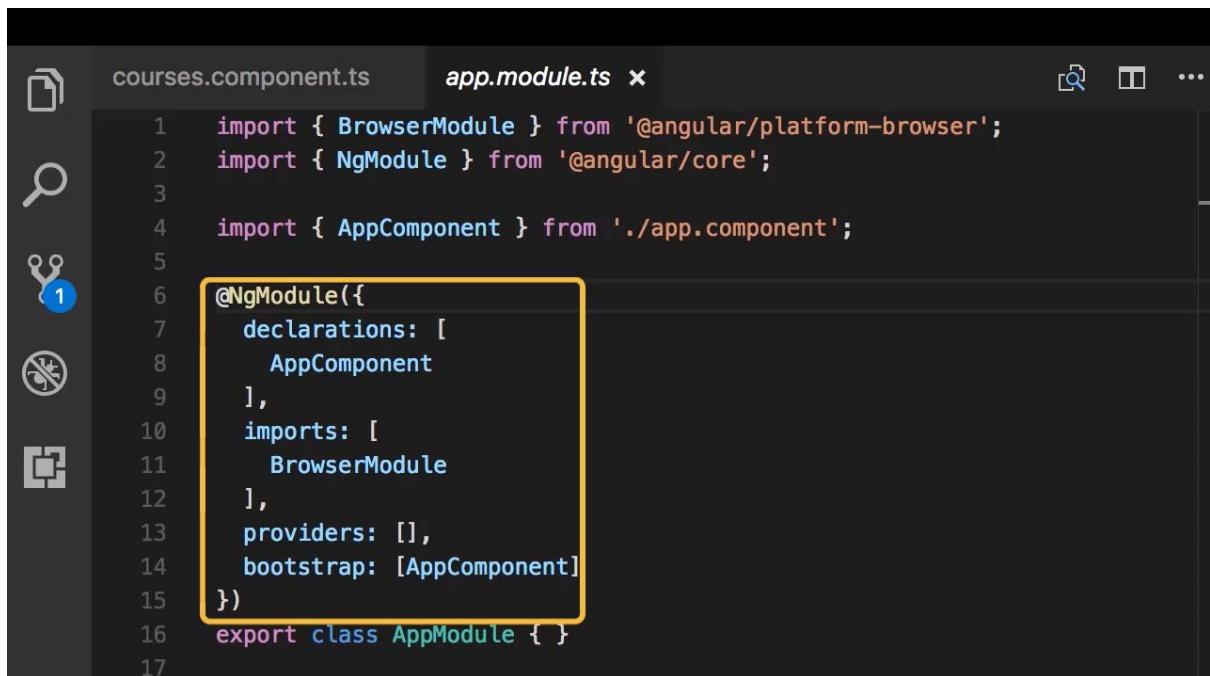
```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 
4 import { AppComponent } from './app.component';
5 
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10  imports: [
11    BrowserModule
12  ],
13  providers: [],
14  bootstrap: [AppComponent]
15 })
16 export class AppModule { }
```



```
courses.component.ts      app.module.ts x
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10  imports: [
11    BrowserModule
12  ],
13  providers: [],
14  bootstrap: [AppComponent]
15 })
16 export class AppModule { }
17
```

The screenshot shows the VS Code interface with the file 'app.module.ts' open. The code defines an Angular module named 'AppModule'. It includes declarations for the 'AppComponent', imports of 'BrowserModule', and bootstrap it with 'AppComponent'. The entire code block is highlighted with a yellow box.

Decorator function @NgModule



```
courses.component.ts      app.module.ts x
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10  imports: [
11    BrowserModule
12  ],
13  providers: [],
14  bootstrap: [AppComponent]
15 })
16 export class AppModule { }
17
```

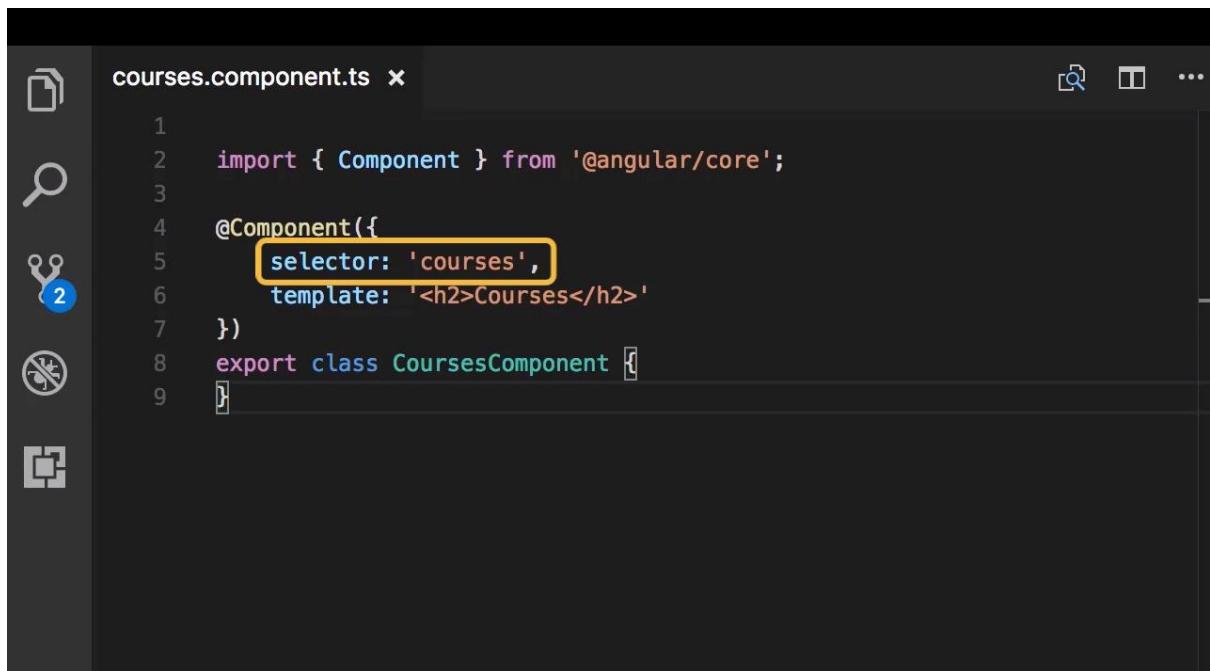
The screenshot shows the VS Code interface with the file 'app.module.ts' open. The code is identical to the previous one, defining the 'AppModule' with its declarations, imports, providers, and bootstrap. The entire code block is highlighted with a yellow box, and the '@NgModule' decorator itself is also highlighted with a yellow box.

```
courses.component.ts app.module.ts ●
1 | import { CoursesComponent } from './courses.component';
2 | import { BrowserModule } from '@angular/platform-browser';
3 | import { NgModule } from '@angular/core';
4 |
5 | import { AppComponent } from './app.component';
6 |
7 | @NgModule({
8 |   declarations: [
9 |     AppComponent,
10 |     CoursesComponent
11 |   ],
12 |   imports: [
13 |     BrowserModule
14 |   ],
15 |   providers: [],
16 |   bootstrap: [AppComponent]
17 | })
```

A screenshot of the Visual Studio Code interface. The left sidebar shows icons for files, search, and extensions. The main editor area has two tabs: 'courses.component.ts' and 'app.module.ts'. The 'app.module.ts' tab is active, displaying the code above. A yellow arrow points from the text 'Auto-import Plugin' in a tooltip at the bottom right towards the 'BrowserModule' import statement. The status bar at the bottom shows the file path 'C:\Users\...'. The title bar includes the VS Code logo, the file name 'app.module.ts', and a status indicator.

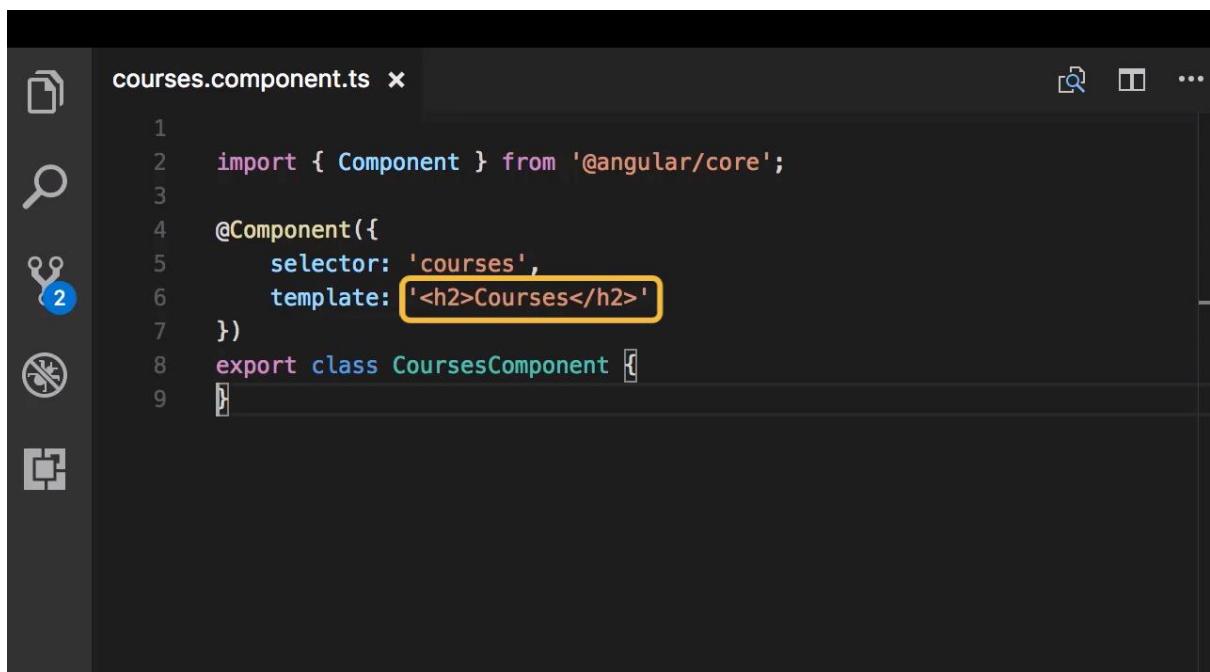
```
EXTENSIONS courses.component.ts app.module.ts ●
1 | import { CoursesComponent } from './courses.component';
2 | import { BrowserModule } from '@angular/platform-browser';
3 | import { NgModule } from '@angular/core';
4 |
5 | import { AppComponent } from './app.component';
6 |
7 | @NgModule({
8 |   declarations: [
9 |     AppComponent,
10 |     CoursesComponent
11 |   ],
12 |   imports: [
13 |     BrowserModule
14 |   ],
15 |   providers: [],
16 |   bootstrap: [AppComponent]
17 | })
```

A screenshot of the Visual Studio Code interface. The left sidebar shows icons for files, search, and extensions. The main editor area has two tabs: 'courses.component.ts' and 'app.module.ts'. The 'app.module.ts' tab is active, displaying the code above. In the extensions sidebar, the 'auto import' extension is selected and highlighted with a blue border. Other extensions listed include 'Auto Import' by steoates, 'Auto Import - ES6 & TS' by Martin Oppitz, 'Auto Close Tag' by Jun Han, 'Auto Rename Tag' by Jun Han, and 'Auto-Open Markdown Preview' by Jun Han. The status bar at the bottom shows the file path 'C:\Users\...'. The title bar includes the VS Code logo, the file name 'app.module.ts', and a status indicator.



A screenshot of a code editor window titled "courses.component.ts". The code is written in TypeScript and defines a component named "CoursesComponent". The "selector" property is set to "courses", and the "template" property contains the string "<h2>Courses</h2>". The "selector" line is highlighted with a yellow box.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>Courses</h2>'
6 })
7 export class CoursesComponent {
8 }
9
```

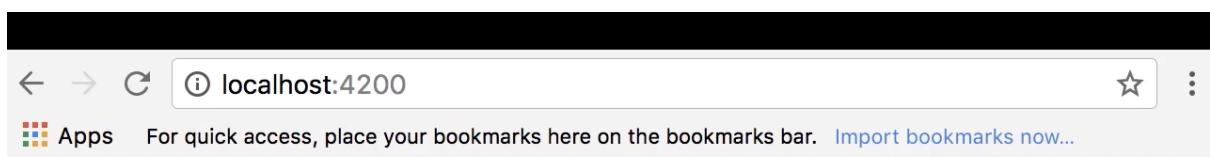


A screenshot of a code editor window titled "courses.component.ts". The code is identical to the one in the first screenshot, defining a "CoursesComponent" with a selector of "courses" and a template of "<h2>Courses</h2>". The "template" line is highlighted with a yellow box.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>Courses</h2>'
6 })
7 export class CoursesComponent {
8 }
9
```

The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other tools. The main area displays the file `app.component.html`. The code contains two lines:

```
1 <h1>Angular</h1>
2 <courses></courses>
```



Angular

Courses

```
<!DOCTYPE html>
<html lang="en">
  ><head>...</head>
  ><body>
    ><app-root _ngcontent-c0 ng-version="4.1.3">
      ><h1 _ngcontent-c0>Angular</h1>
      ><courses _ngcontent-c0>
        .. ><h2>Courses</h2> == $0
      ></courses>
    ></app-root>
    <script type="text/javascript" src="inline.bundle.js"></script>
    <script type="text/javascript" src="polyfills.bundle.js"></script>
    <script type="text/javascript" src="styles.bundle.js"></script>
    <script type="text/javascript" src="vendor.bundle.js"></script>
    <script type="text/javascript" src="main.bundle.js"></script>
  ></body>
</html>
```

html body app-root courses **h2**

Styles Event Listeners DOM Breakpoints Properties

Filter

:hov .cls +

```
<!DOCTYPE html>
<html lang="en">
  ><head>...</head>
  ><body>
    ><app-root _ngcontent-c0 ng-version="4.1.3"> == $0
      ><h1 _ngcontent-c0>Angular</h1>
      ><courses _ngcontent-c0>
        ><h2>Courses</h2>
      ></courses>
    ></app-root>
    <script type="text/javascript" src="inline.bundle.js"></script>
    <script type="text/javascript" src="polyfills.bundle.js"></script>
    <script type="text/javascript" src="styles.bundle.js"></script>
    <script type="text/javascript" src="vendor.bundle.js"></script>
    <script type="text/javascript" src="main.bundle.js"></script>
  ></body>
</html>
```

html body **app-root** courses

Styles Event Listeners DOM Breakpoints Properties

Filter

:hov .cls +

EXPLORER

- OPEN EDITORS
- index.html src

HELLO-WORLD

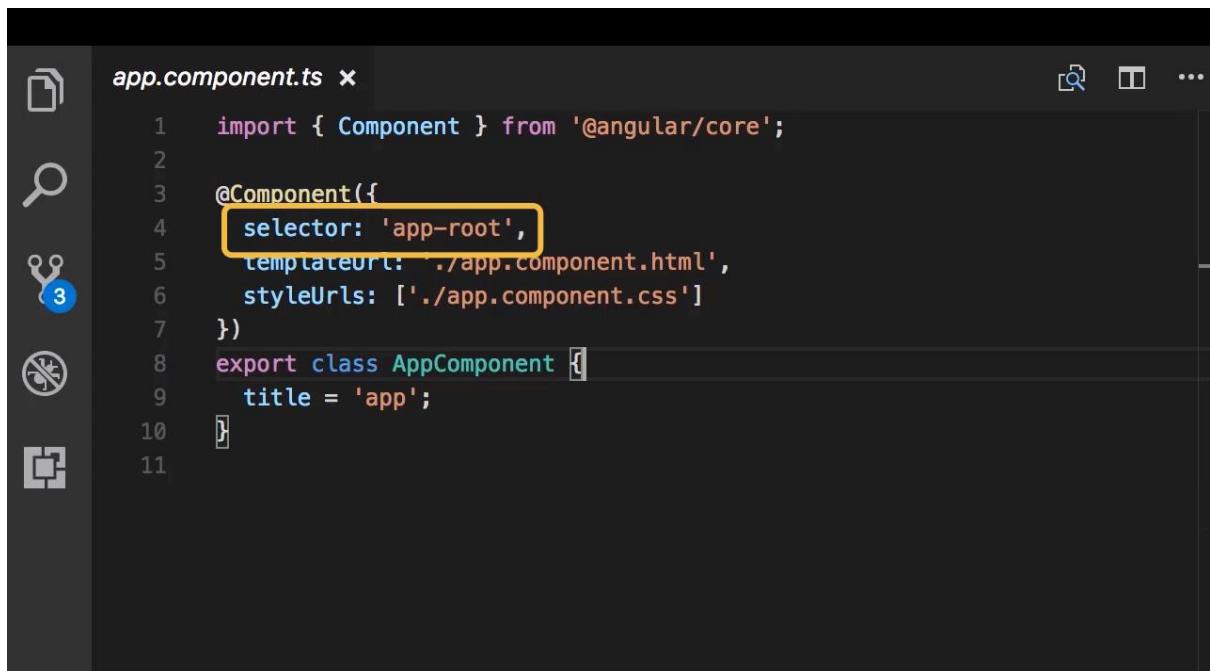
- e2e
- node_modules
- src
 - app
 - assets
 - environments
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts
 - styles.css

index.html

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>HelloWorld</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
```

index.html

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8">
5   <title>HelloWorld</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="icon" type="image/x-icon" href="favicon.ico">
10 </head>
11 <body>
12   <app-root></app-root>
13 </body>
14 </html>
```

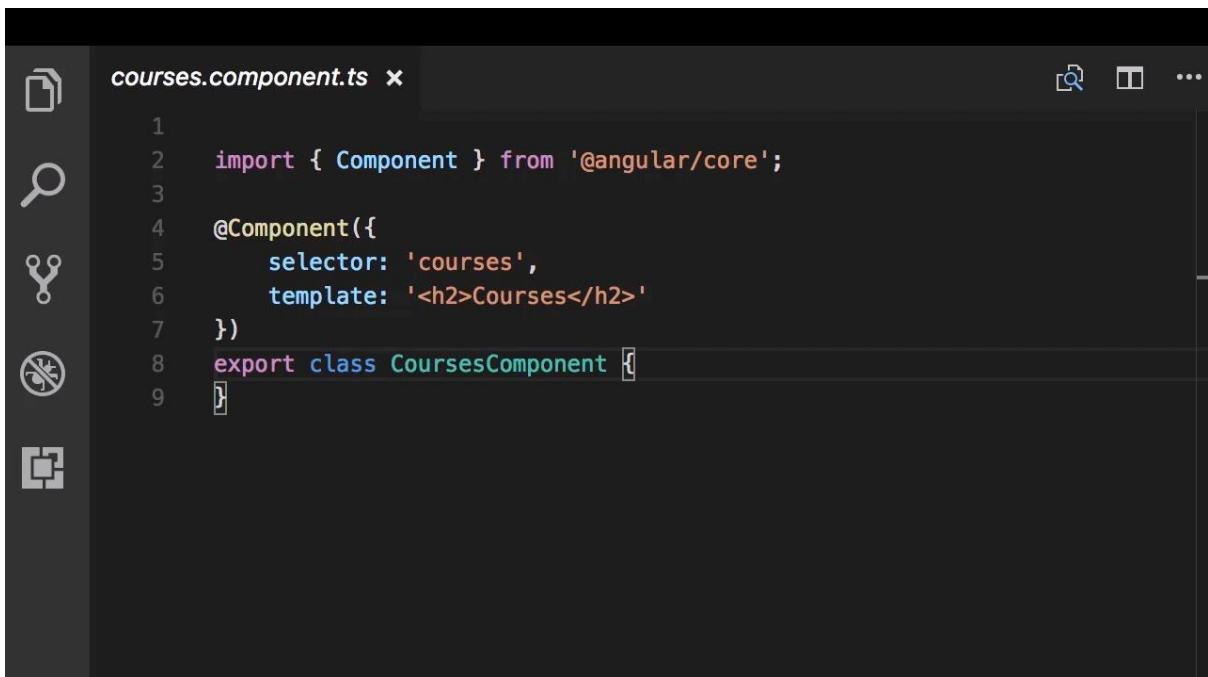


The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other tools, with a '3' badge indicating three notifications. The main area displays a file named `app.component.ts`. The code is as follows:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'app';
10 }
11
```

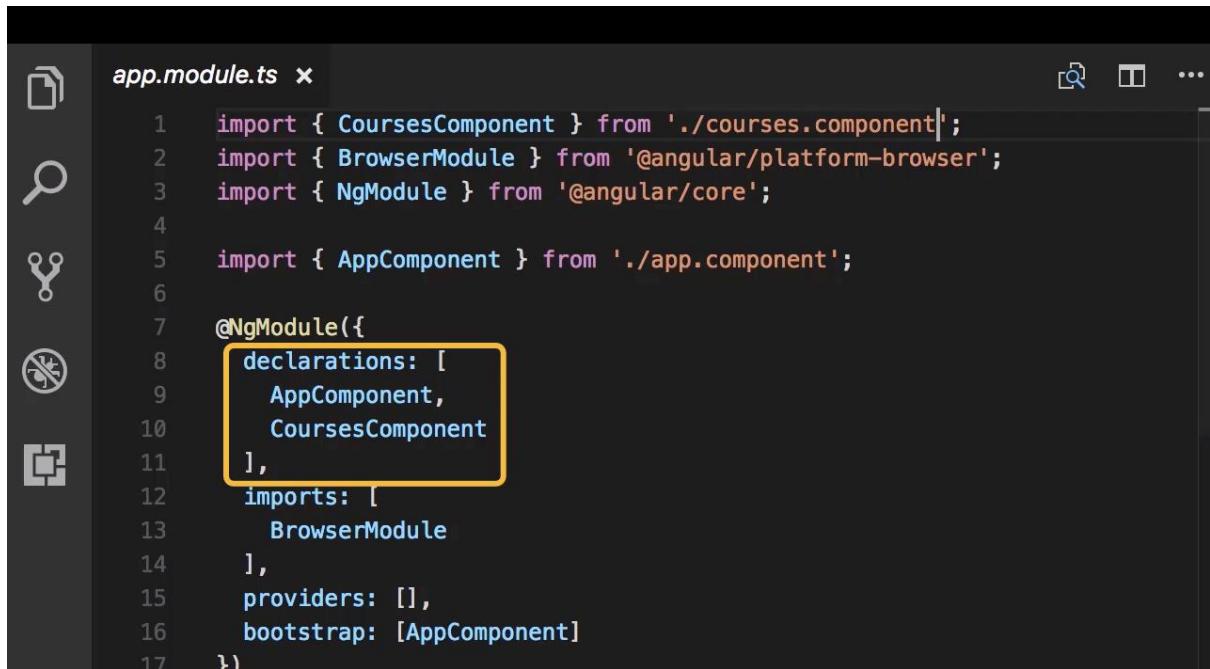
The line `selector: 'app-root'` is highlighted with a yellow rectangular selection.

Creating Components Using Angular CLI

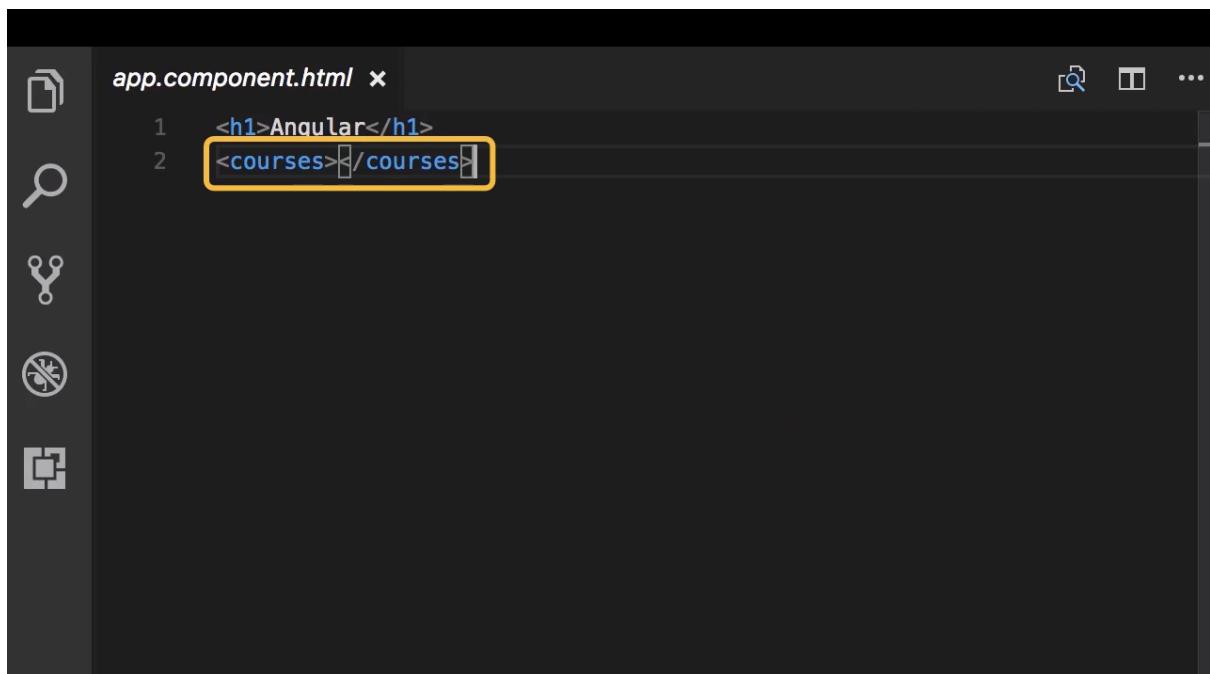


The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations (New, Open, Save, Find, Replace, Undo, Redo, etc.). The main area displays the content of a file named `courses.component.ts`. The code is as follows:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>Courses</h2>'
6 })
7 export class CoursesComponent {}
```



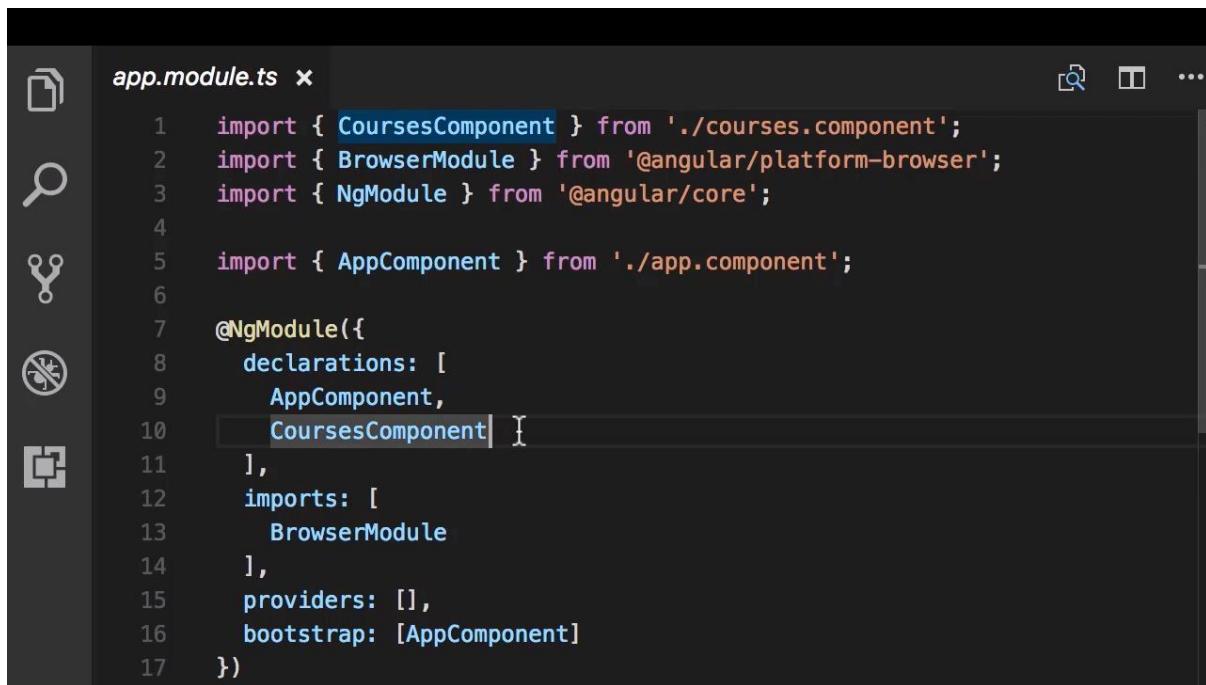
```
app.module.ts x
1 import { CoursesComponent } from './courses.component';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent,
10    CoursesComponent
11   ],
12   imports: [
13     BrowserModule
14   ],
15   providers: [],
16   bootstrap: [AppComponent]
17 })
```



```
app.component.html x
1 <h1>Angular</h1>
2 <courses>[]</courses>
```

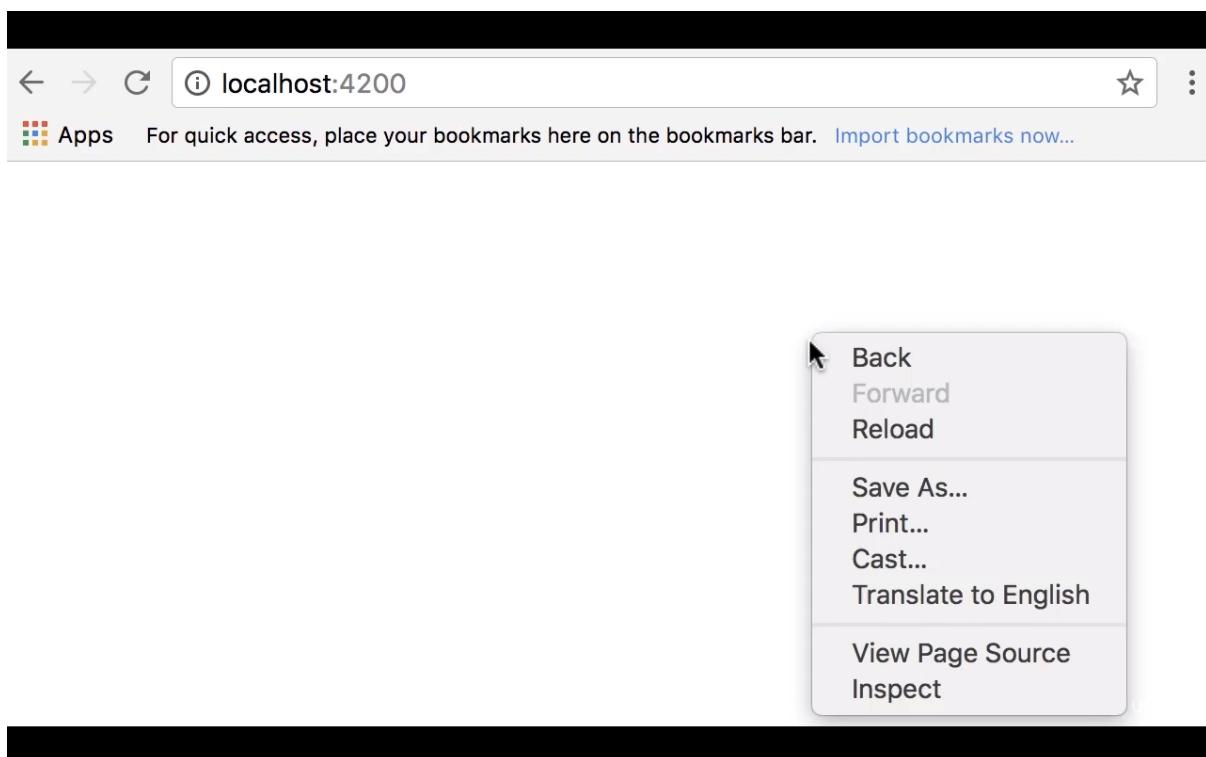
Two problems with this approach:

1. Its tedious
2. If we forget the registration step , the app is going to break.



The screenshot shows a code editor window with the file `app.module.ts` open. The code defines an Angular module with the following content:

```
1 import { CoursesComponent } from './courses.component';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4
5 import { AppComponent } from './app.component';
6
7 @NgModule({
8   declarations: [
9     AppComponent,
10    CoursesComponent]
11 ],
12 imports: [
13   BrowserModule
14 ],
15 providers: [],
16 bootstrap: [AppComponent]
17 })
```



Developer Tools - http://localhost:4200/

Elements Console Sources Network Performance Memory > ✖ 2 :

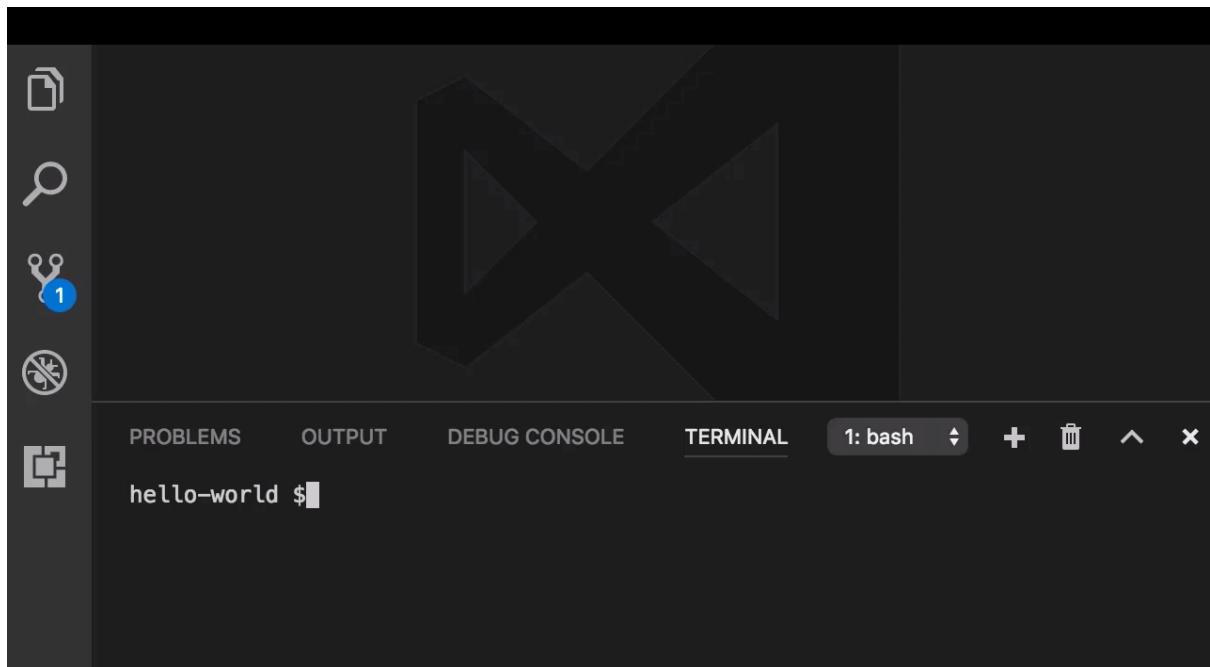
Filter Info ▾

✖ Unhandled Promise rejection: Template parse errors:
'courses' is not a known element: zone.js:642

```
1. If 'courses' is an Angular component, then verify that it is part of this module.
2. To allow any element add 'NO_ERRORS_SCHEMA' to the '@NgModule.schemas' of this
component. ("<h1>Angular</h1>
[ERROR --]<courses></courses>"): ng:/// AppModule/AppComponent.html@1:0 ; Zone: <root> ;
Task: Promise.then ; Value: Error: Template parse errors:
'courses' is not a known element:
1. If 'courses' is an Angular component, then verify that it is part of this module.
2. To allow any element add 'NO_ERRORS_SCHEMA' to the '@NgModule.schemas' of this
component. ("<h1>Angular</h1>
[ERROR --]<courses></courses>"): ng:/// AppModule/AppComponent.html@1:0
    at syntaxError (http://localhost:4200/vendor.bundle.js:5753:34)
    at
TemplateParser.webpackJsonp.../node_modules/@angular/compiler/@angular/compiler.es5.js.Temp
lateParser.parse (http://localhost:4200/vendor.bundle.js:16244:19)
    at
JitCompiler.webpackJsonp.../node_modules/@angular/compiler/@angular/compiler.es5.js.JitComp
iler._compileTemplate (http://localhost:4200/vendor.bundle.js:29995:39)
    at http://localhost:4200/vendor.bundle.js:29919:62
```

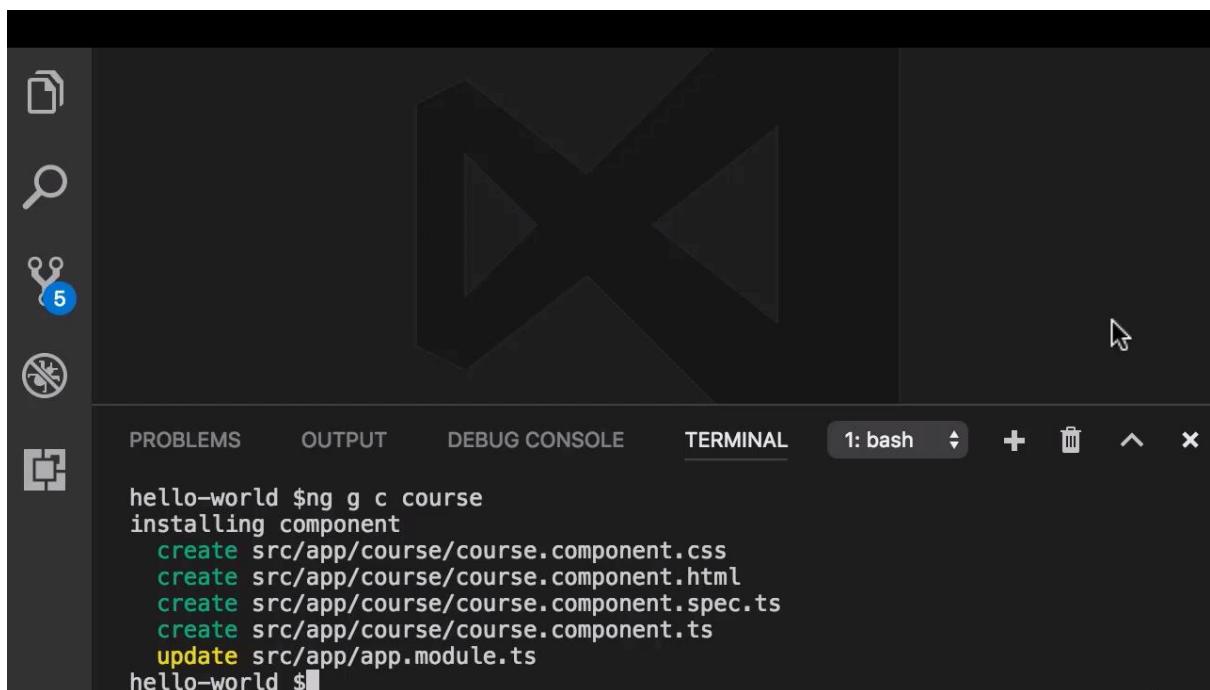
Now let us see a quicker and more reliable way to create a angular component

- Goto visual studio
- Press ctrl + backtick (before the number 1) to run node inside vs



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash + ⌫ ⌄ ⌂

```
hello-world $
```

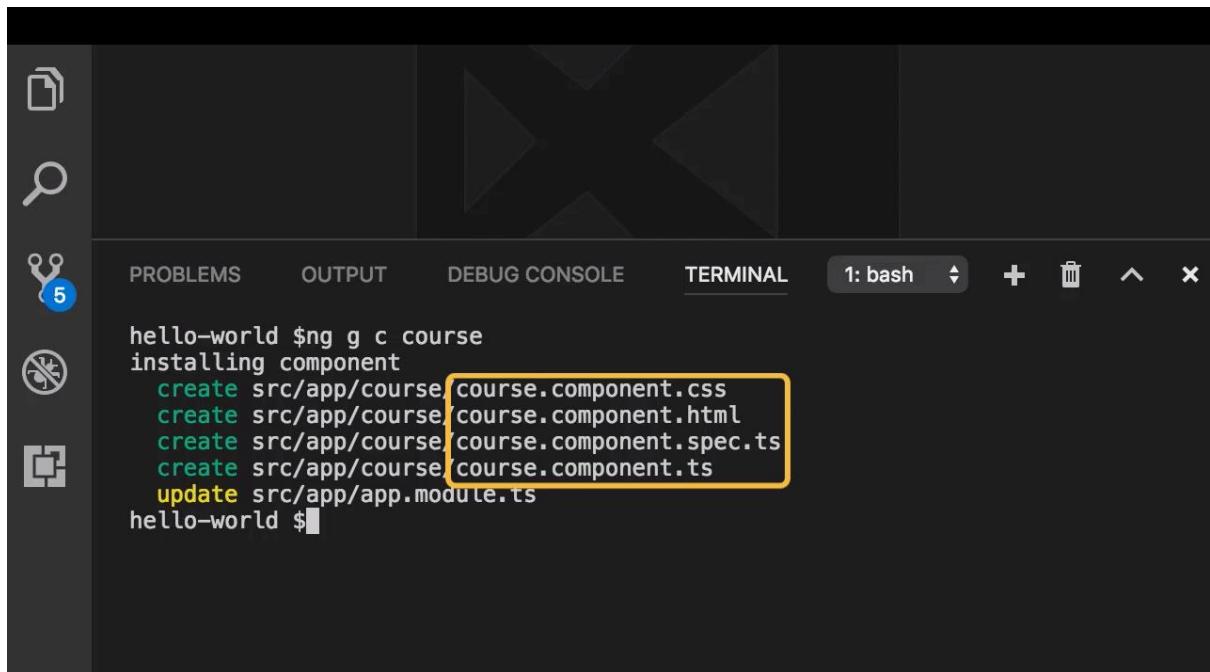


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash + ⌫ ⌄ ⌂

```
hello-world $ng g c course
installing component
  create src/app/course/course.component.css
  create src/app/course/course.component.html
  create src/app/course/course.component.spec.ts
  create src/app/course/course.component.ts
  update src/app/app.module.ts
hello-world $
```

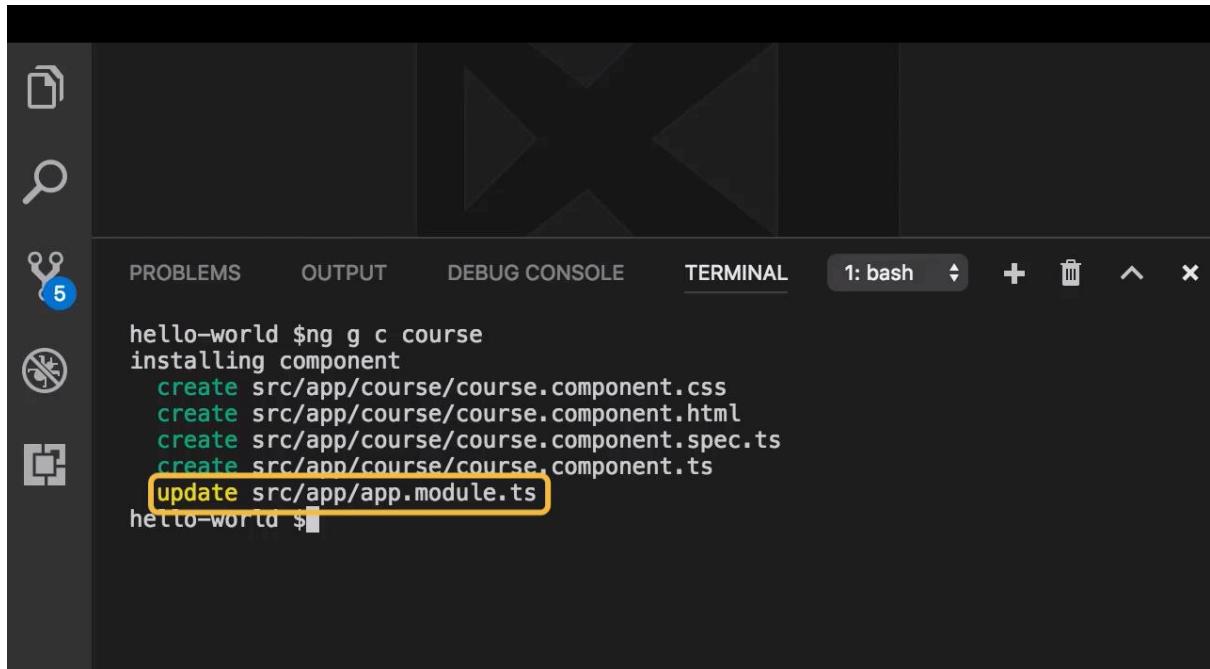
g stands for generate

c stands for component



The screenshot shows the VS Code interface with the terminal tab selected. The command `$ng g c course` was run, which generated a new component named 'course'. The output shows the creation of four files: `course.component.css`, `course.component.html`, `course.component.spec.ts`, and `course.component.ts`. An update to `app.module.ts` was also performed. The terminal window has a dark theme and includes icons for file operations like copy, paste, and delete.

```
hello-world $ng g c course
installing component
create src/app/course/course.component.css
create src/app/course/course.component.html
create src/app/course/course.component.spec.ts
create src/app/course/course.component.ts
update src/app/app.module.ts
hello-world $
```



The screenshot shows the VS Code interface with the terminal tab selected. The command `$ng g c course` was run, which generated a new component named 'course'. The output shows the creation of four files: `course.component.css`, `course.component.html`, `course.component.spec.ts`, and `course.component.ts`. An update to `app.module.ts` was also performed. In this version, the component is nested under 'course' in the directory structure. The terminal window has a dark theme and includes icons for file operations like copy, paste, and delete.

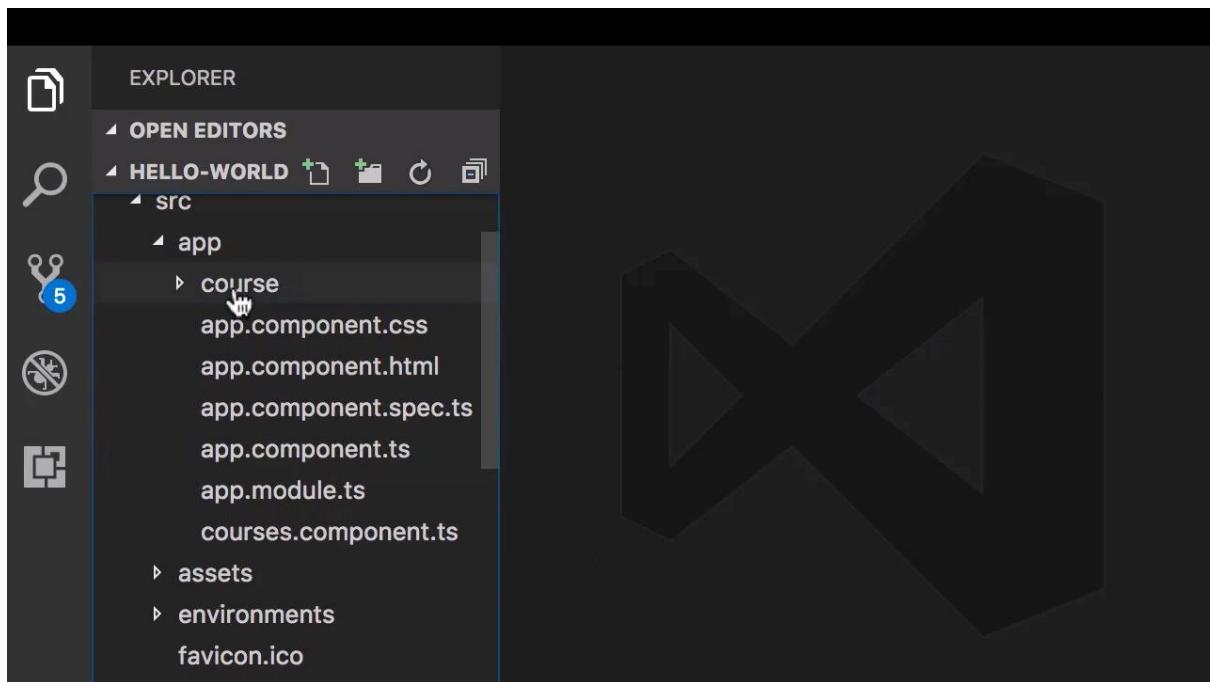
```
hello-world $ng g c course
installing component
create src/app/course/course.component.css
create src/app/course/course.component.html
create src/app/course/course.component.spec.ts
create src/app/course/course.component.ts
update src/app/app.module.ts
hello-world $
```

app.module.ts x

```
1 import { CoursesComponent } from './courses.component';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4
5 import { AppComponent } from './app.component';
6 import { CourseComponent } from './course/course.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     CourseComponent ←
12   ],
13   imports: [
14     BrowserModule
15   ],
16   providers: []
17 })
```

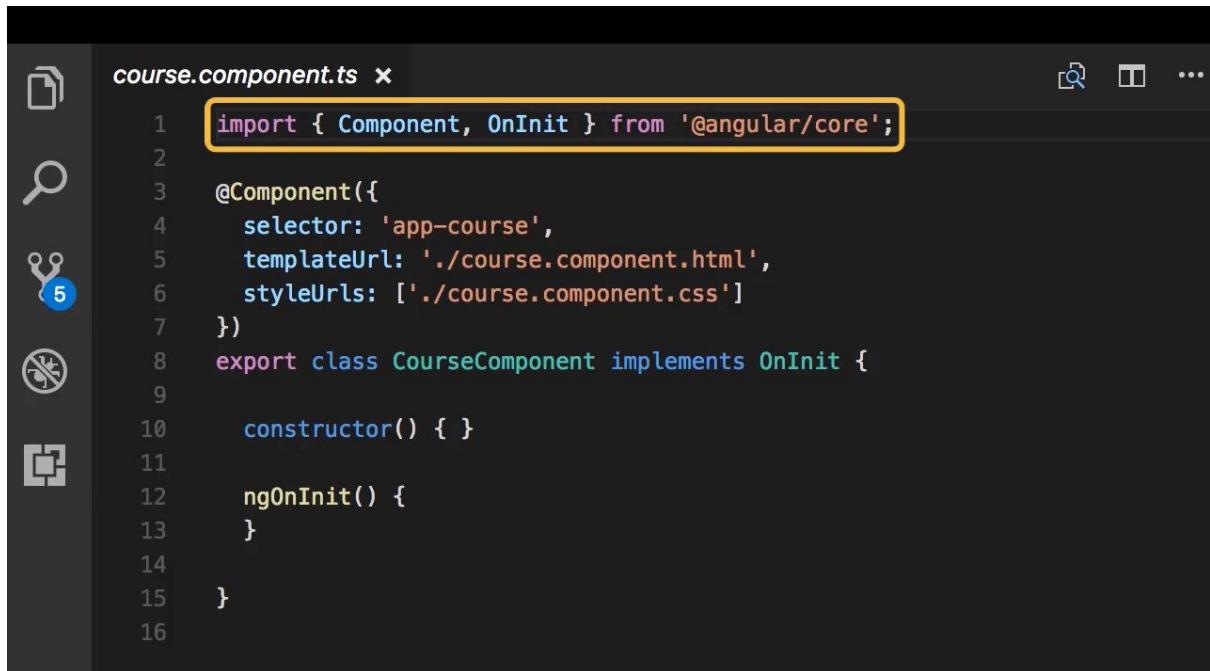
app.module.ts ●

```
1 import { CoursesComponent } from './courses.component';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { NgModule } from '@angular/core';
4
5 import { AppComponent } from './app.component';
6 import { CourseComponent } from './course/course.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     CourseComponent,
12     CoursesComponent]
13   ],
14   imports: [
15     BrowserModule
16   ],
17   providers: []
18 })
```

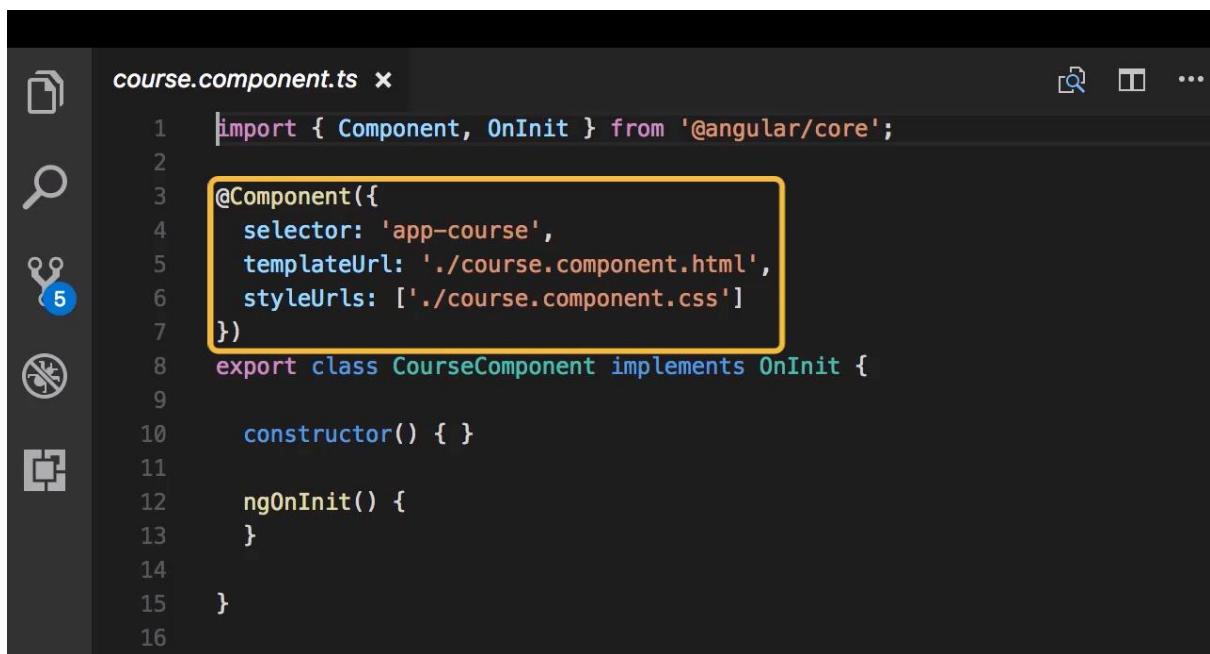


```
course.component.ts ×
```

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-course',
5   templateUrl: './course.component.html',
6   styleUrls: ['./course.component.css']
7 })
8 export class CourseComponent implements OnInit {
9
10   constructor() { }
11
12   ngOnInit() {
13   }
14
15 }
16
```



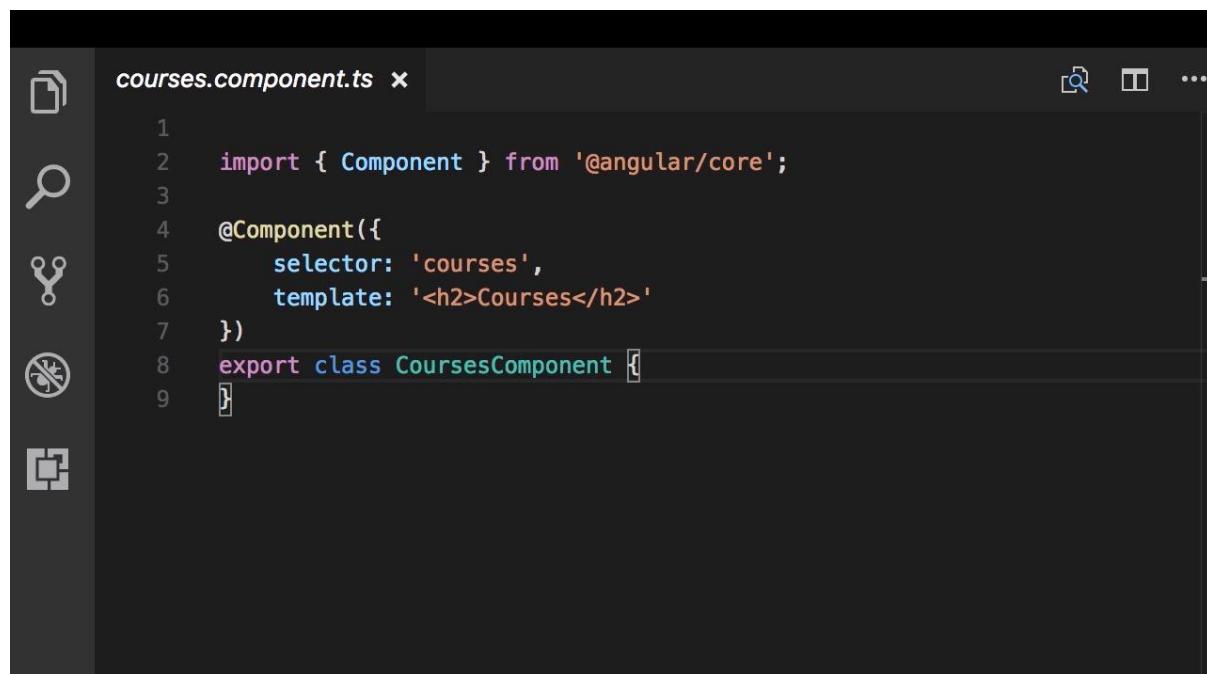
```
course.component.ts ✘
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4      selector: 'app-course',
5      templateUrl: './course.component.html',
6      styleUrls: ['./course.component.css']
7  })
8  export class CourseComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit() {
13     }
14
15 }
16
```



```
course.component.ts ✘
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4      selector: 'app-course',
5      templateUrl: './course.component.html',
6      styleUrls: ['./course.component.css']
7  })
8  export class CourseComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit() {
13     }
14
15 }
16
```

Templates

Templates

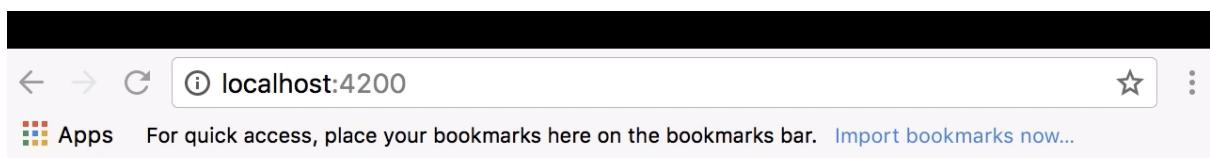


The screenshot shows a dark-themed code editor window with a sidebar on the left containing icons for file operations like new file, search, and refresh. The main area displays the content of a file named `courses.component.ts`. The code is written in TypeScript and defines a component named `CoursesComponent`.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>Courses</h2>'
6 })
7
8 export class CoursesComponent {
9 }
```

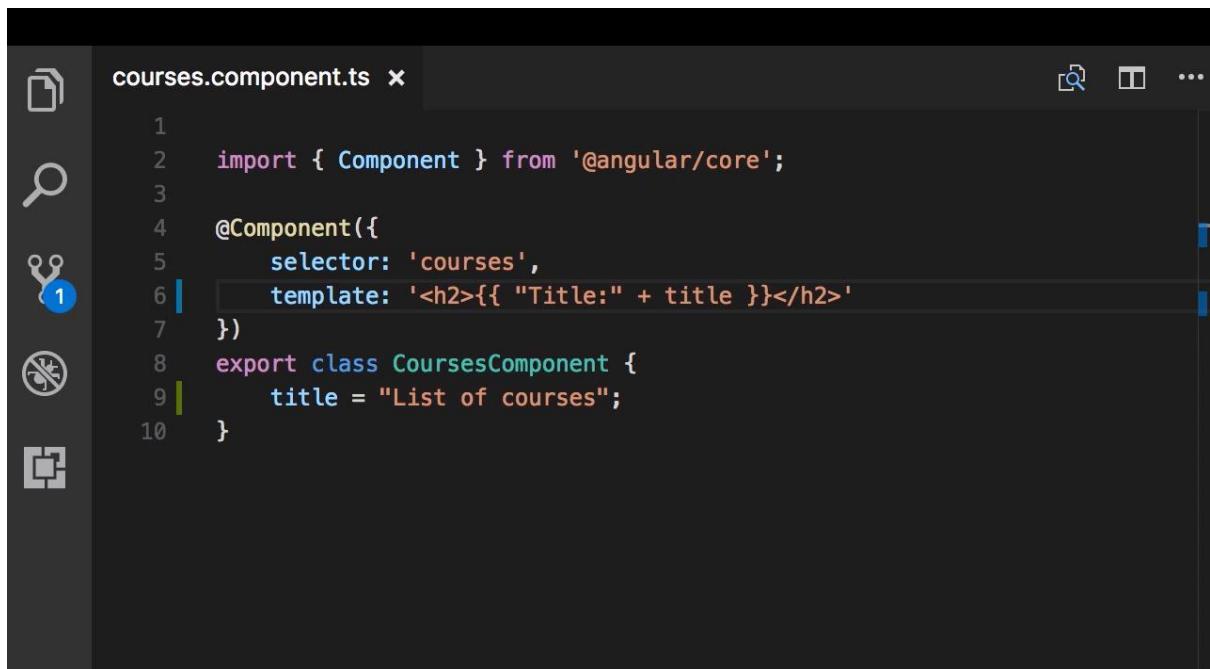
The screenshot shows the code editor interface of Visual Studio Code. On the left is a sidebar with icons for file operations like Open, Save, Find, and others. The main area displays the file 'courses.component.ts' with the following code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>{{ title }}</h2>'
6 })
7 export class CoursesComponent {
8   title = "List of courses";
9 }
10
```



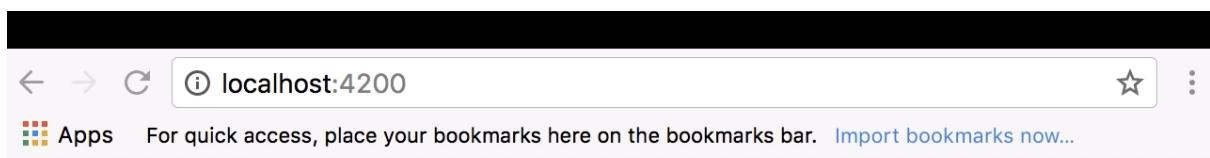
Angular

List of courses ←



The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other tools. The main area displays the file 'courses.component.ts' with the following code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>{{ "Title:" + title }}</h2>'
6 })
7 export class CoursesComponent {
8   title = "List of courses";
9 }
10
```

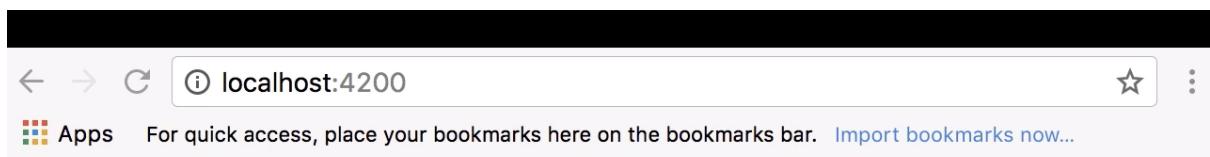


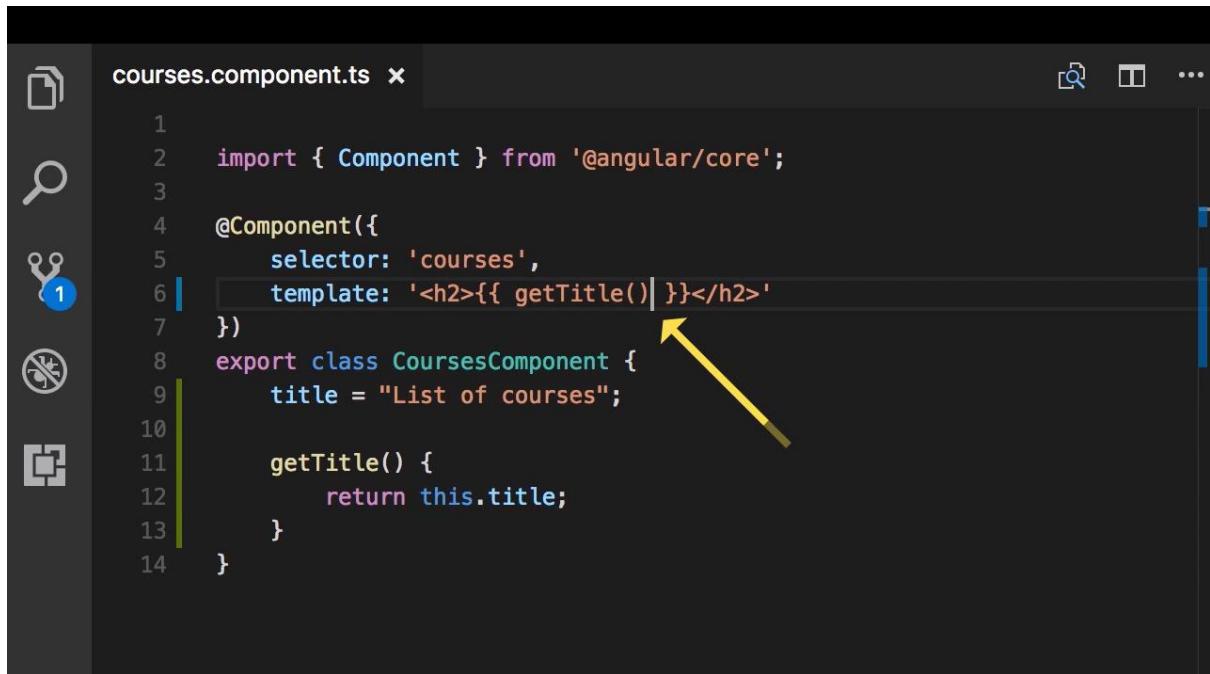
Angular

Title:List of courses

The screenshot shows the code editor interface of Visual Studio Code. On the left is a dark sidebar with several icons: a document with a blue circle containing the number '1', a magnifying glass, a wrench and gear, and a square with rounded corners. The main area displays the file 'courses.component.ts' with the following code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>{{ getTitle() }}</h2>'
6 })
7 export class CoursesComponent {
8   title = "List of courses";
9
10  getTitle() {
11    return this.title;
12  }
13}
14
```





A screenshot of a code editor showing a file named `courses.component.ts`. The code defines an Angular component named `CoursesComponent` with the selector `'courses'`. The template contains an `<h2>` element with an interpolation brace `{} {{ }}`. A yellow arrow points from the text "Interpolation" below to this brace in the template. The code is as follows:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: '<h2>{{ getTitle() }}</h2>'
6 })
7 export class CoursesComponent {
8   title = "List of courses";
9
10  getTitle() {
11    return this.title;
12  }
13}
14
```

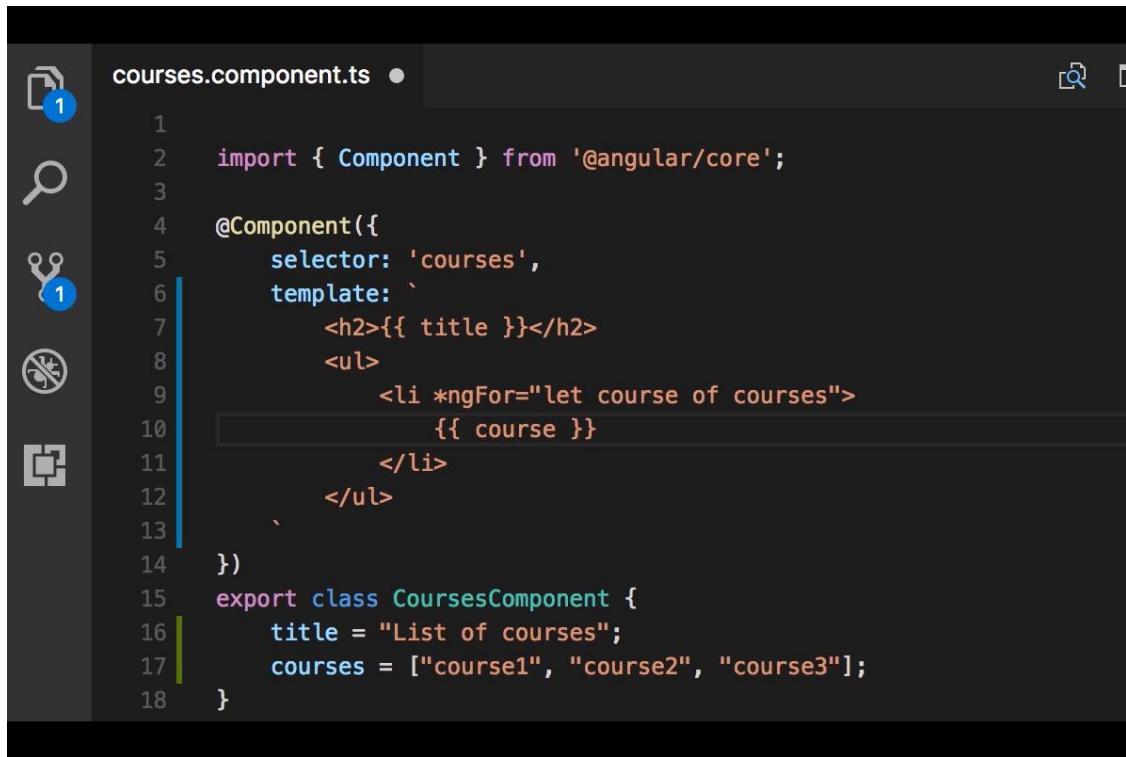
Interpolation

Directives



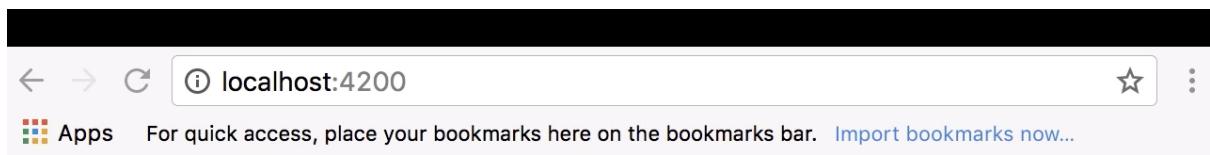
Directives





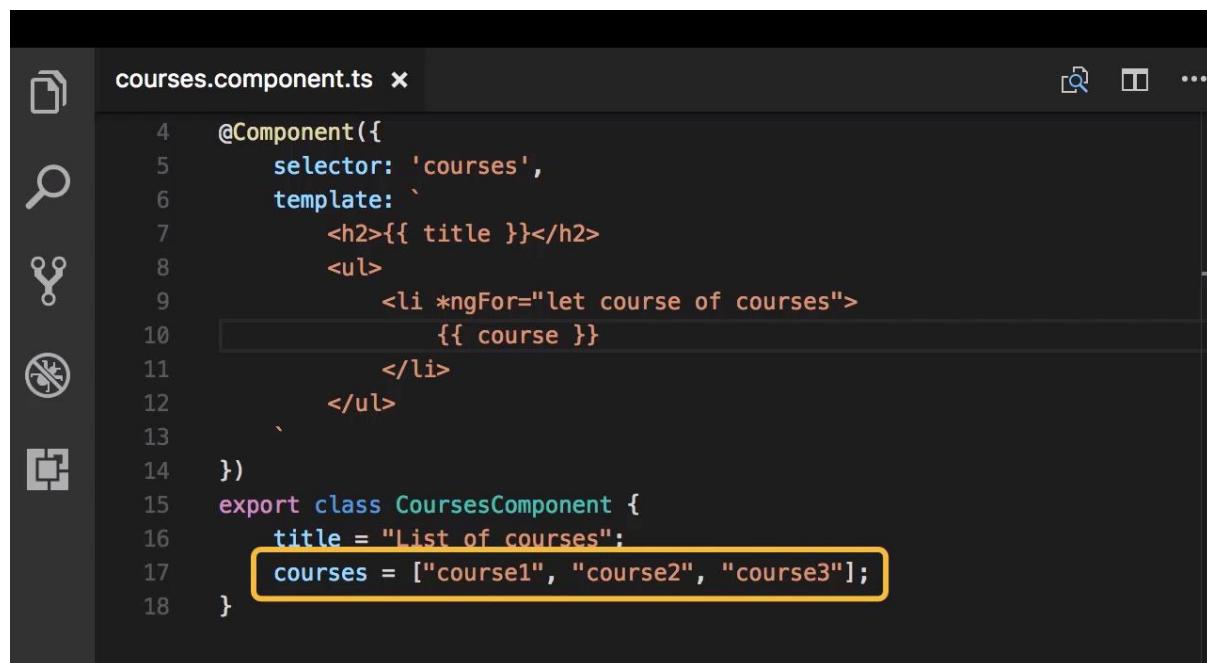
The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other tools. The main area displays the file `courses.component.ts`. The code defines a component named `CoursesComponent` with the selector `'courses'`. The template contains an `<h2>` element with a title, followed by an `` element containing three `` elements, each displaying one of the three courses listed in the array `courses`.

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'courses',
5   template: `
6     <h2>{{ title }}</h2>
7     <ul>
8       <li *ngFor="let course of courses">
9         {{ course }}
10      </li>
11    </ul>
12  `
13})
14 export class CoursesComponent {
15   title = "List of courses";
16   courses = ["course1", "course2", "course3"];
17 }
18 }
```



Services

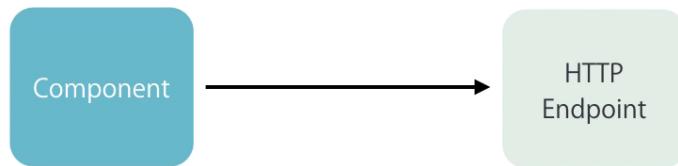
Services



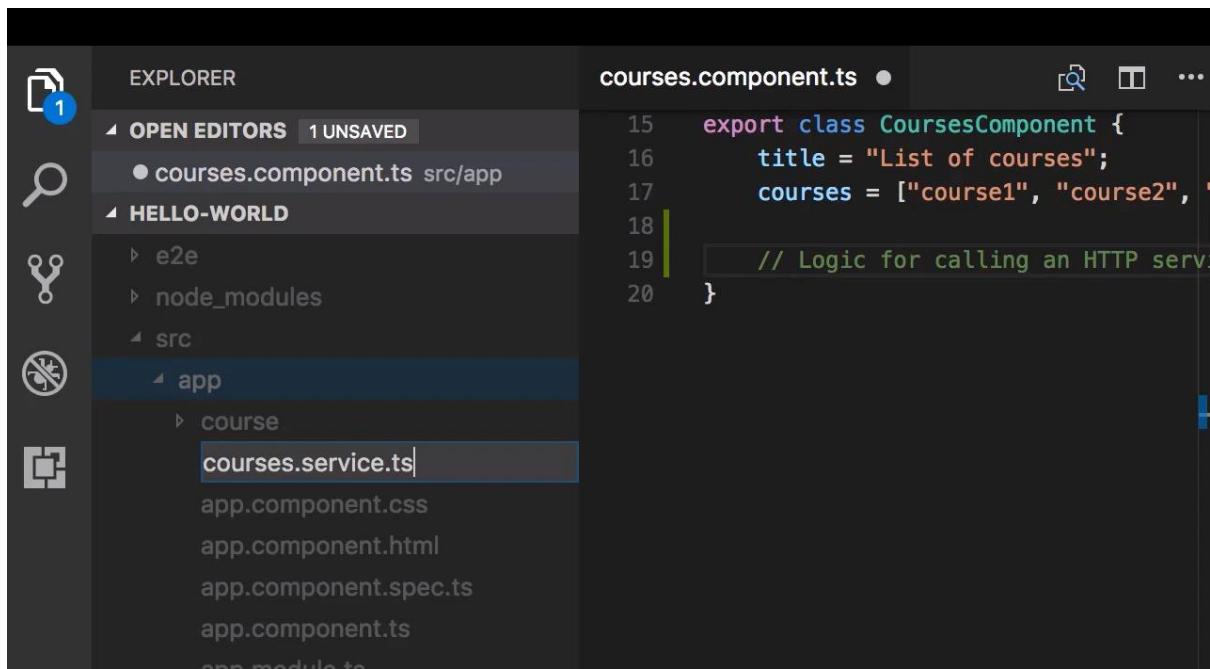
```
courses.component.ts ×
4  @Component({
5    selector: 'courses',
6    template: `
7      <h2>{{ title }}</h2>
8      <ul>
9        <li *ngFor="let course of courses">
10          {{ course }}
11        </li>
12      </ul>
13    `
14  })
15  export class CoursesComponent {
16    title = "List of courses";
17    courses = ["course1", "course2", "course3"];
18 }
```

```
courses.component.ts ●
15  export class CoursesComponent {
16      title = "List of courses";
17      courses = ["course1", "course2", "course3"];
18
19      // Logic for calling an HTTP service [
20  }
```

Tight coupling ...unit testing is difficult



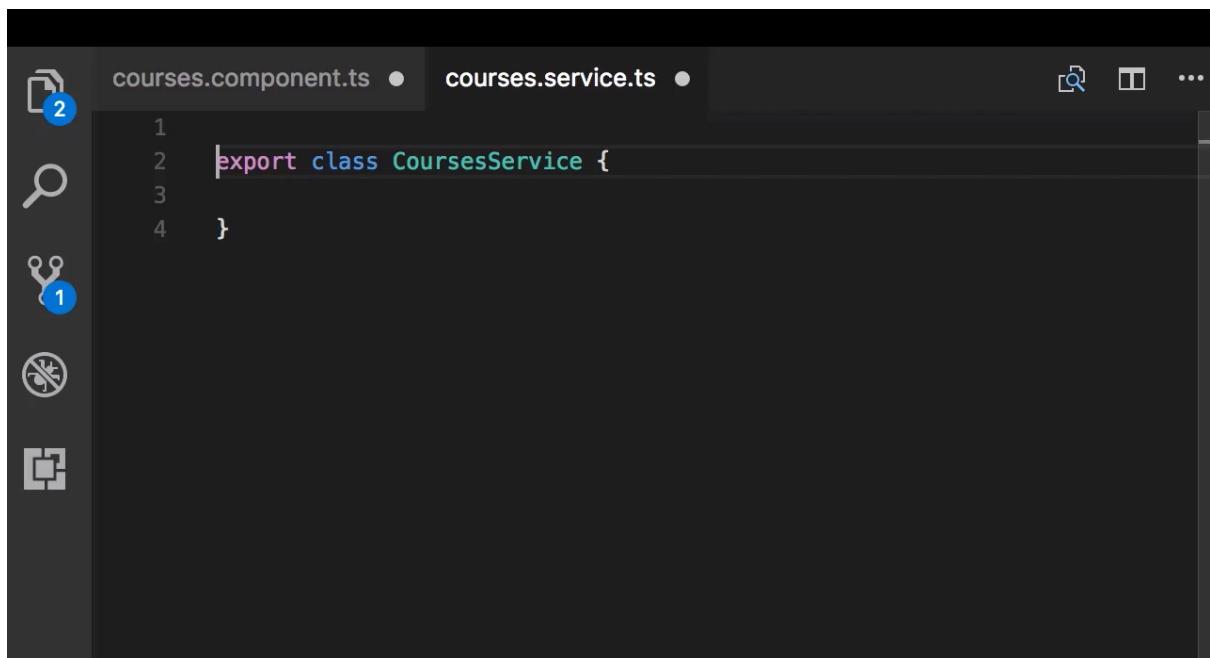
A component should not have any logic other than the presentation logic



The screenshot shows the VS Code interface. The Explorer sidebar on the left displays a file tree. In the main editor area, the file 'courses.component.ts' is open, showing its code. The code defines a component class 'CoursesComponent' with properties 'title' and 'courses', and a method for calling an HTTP service.

```
15 export class CoursesComponent {  
16   title = "List of courses";  
17   courses = ["course1", "course2",  
18   // Logic for calling an HTTP serv  
20 }
```

A service is a plain ts class with no decorator like in the case of a component class



The screenshot shows the VS Code interface with two files open: 'courses.component.ts' and 'courses.service.ts'. Both files have a blue circular badge with the number '1' in the top-left corner of their tabs, indicating they are the active editors. The code for 'courses.component.ts' is identical to the one shown in the previous screenshot. The code for 'courses.service.ts' is a simple export of a class 'CoursesService'.

```
1  
2 export class CoursesService {  
3  
4 }
```

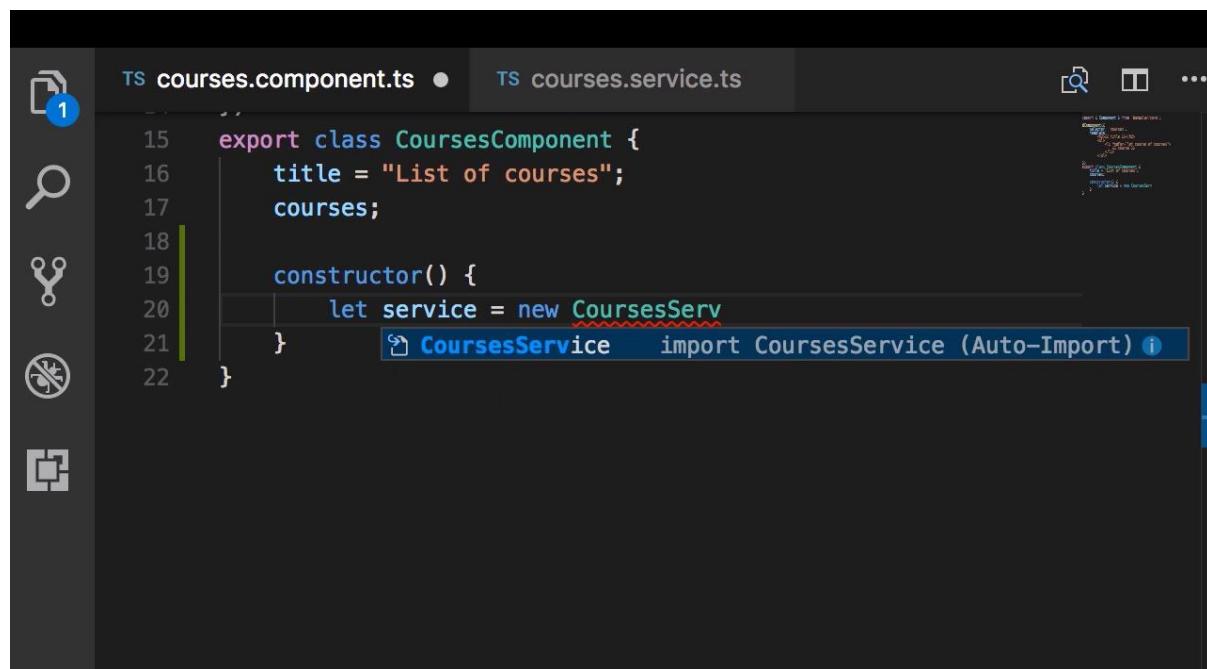
The screenshot shows a dark-themed code editor interface. On the left is a vertical toolbar with icons for file operations, search, and other tools. There are two tabs at the top: "courses.component.ts" and "courses.service.ts". The "courses.service.ts" tab is active, showing the following code:

```
1  export class CoursesService {  
2      getCourses() {  
3          return ["course1", "course2", "course3"];  
4      }  
5  }  
6
```

The code defines a class named CoursesService with a single method getCourses that returns an array of three course names. There are two error markers (blue circles with exclamation points) in the gutter next to the first two lines of code.

Dependency Injection

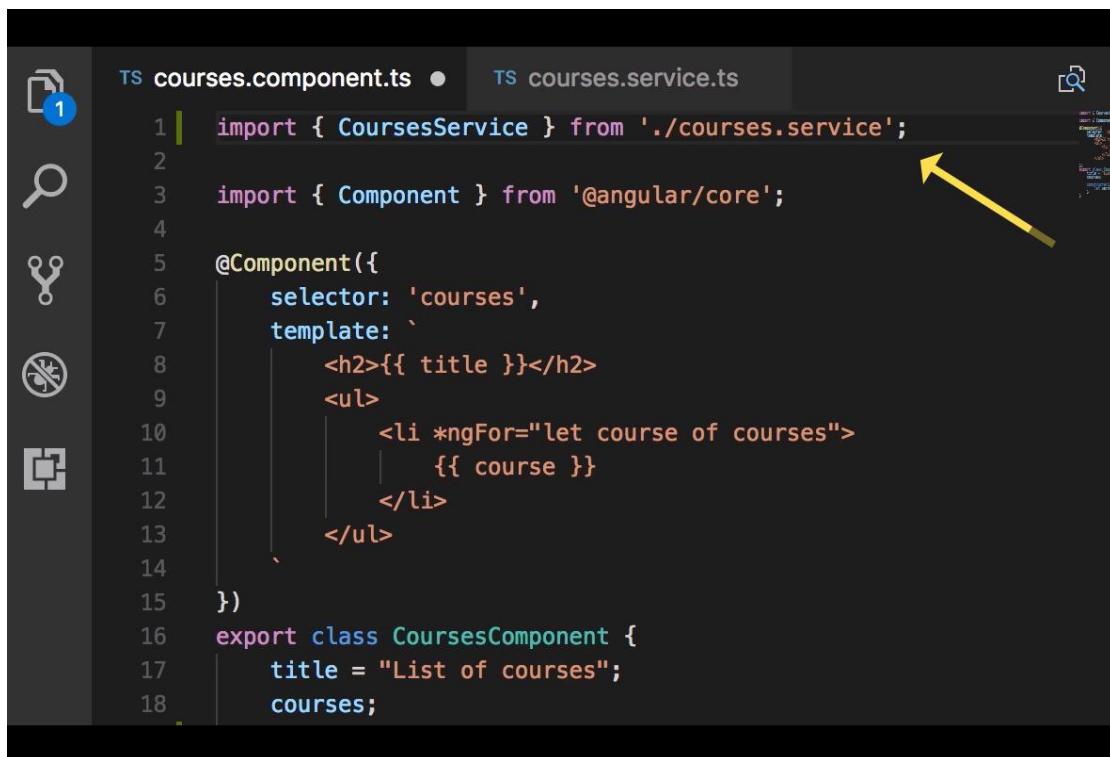
Dependency Injection



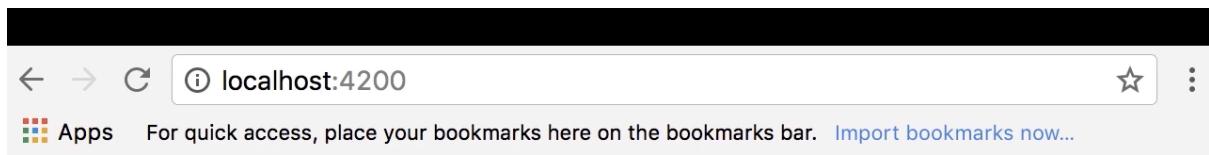
The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with icons for file operations, search, and other tools. The main area displays two tabs: 'courses.component.ts' and 'courses.service.ts'. The 'courses.component.ts' tab is active, showing the following code:

```
15  export class CoursesComponent {
16      title = "List of courses";
17      courses;
18
19      constructor() {
20          let service = new CoursesServ
21      }
22  }
```

The word 'CoursesService' is underlined with a red squiggly line, indicating a potential error or warning. A tooltip or status bar at the bottom right of the editor window says 'CoursesService import CoursesService (Auto-Import)'. The status bar at the top of the editor shows 'TS courses.component.ts ● TS courses.service.ts'.



```
1 import { CoursesService } from './courses.service';
2
3 import { Component } from '@angular/core';
4
5 @Component({
6   selector: 'courses',
7   template: `
8     <h2>{{ title }}</h2>
9     <ul>
10       <li *ngFor="let course of courses">
11         {{ course }}
12       </li>
13     </ul>
14   `
15 })
16 export class CoursesComponent {
17   title = "List of courses";
18   courses;
```



Angular

List of courses

- course1
- course2
- course3

Tight coupling

The screenshot shows the VS Code interface with two tabs open: 'TS courses.component.ts' and 'TS courses.service.ts'. The 'courses.component.ts' tab is active, displaying the following code:

```
15  export class CoursesComponent {  
16      title = "List of courses";  
17      courses;  
18  
19      constructor() {  
20          let service = new CoursesService();  
21          this.courses = service.getCourses();  
22      }  
23  }
```

The line 'let service = new CoursesService();' is highlighted with a yellow box. The status bar at the bottom indicates '1 file is changed'.

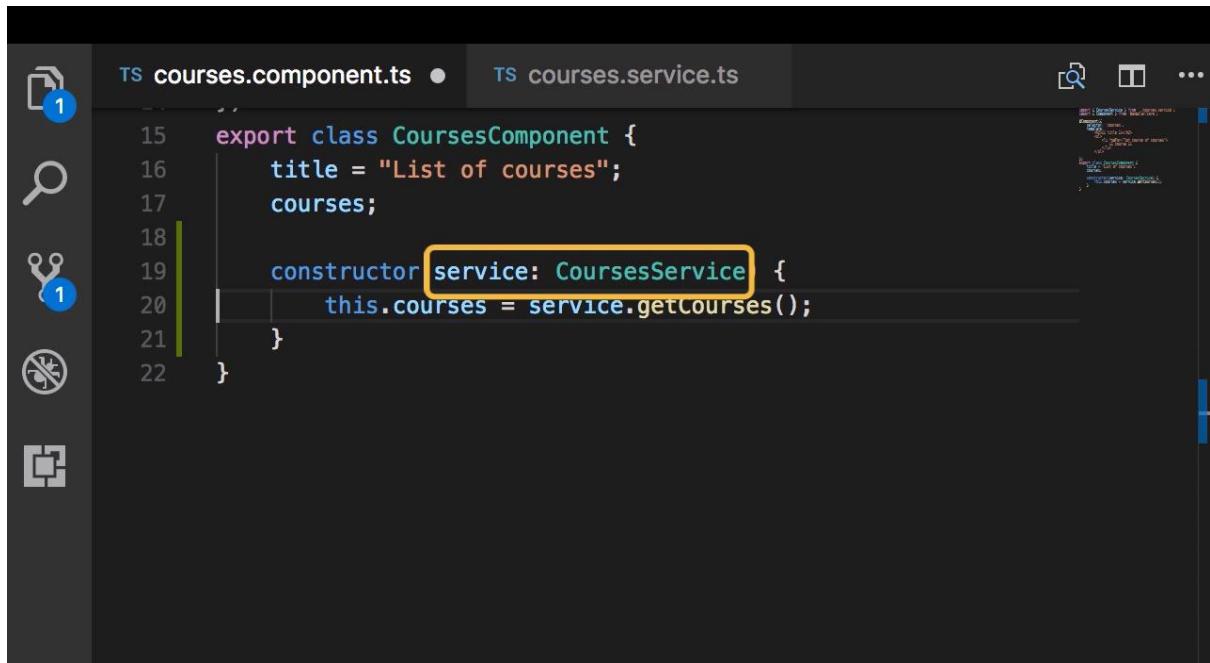
In this case angular injects the CourseService object by calling the constructor. There is no tight coupling

The screenshot shows the VS Code interface with two tabs open: 'TS courses.component.ts' and 'TS courses.service.ts'. The 'courses.component.ts' tab is active, displaying the following code:

```
15  export class CoursesComponent {  
16      title = "List of courses";  
17      courses;  
18  
19      constructor(service: CoursesService) {  
20          this.courses = service.getCourses();  
21      }  
22  }
```

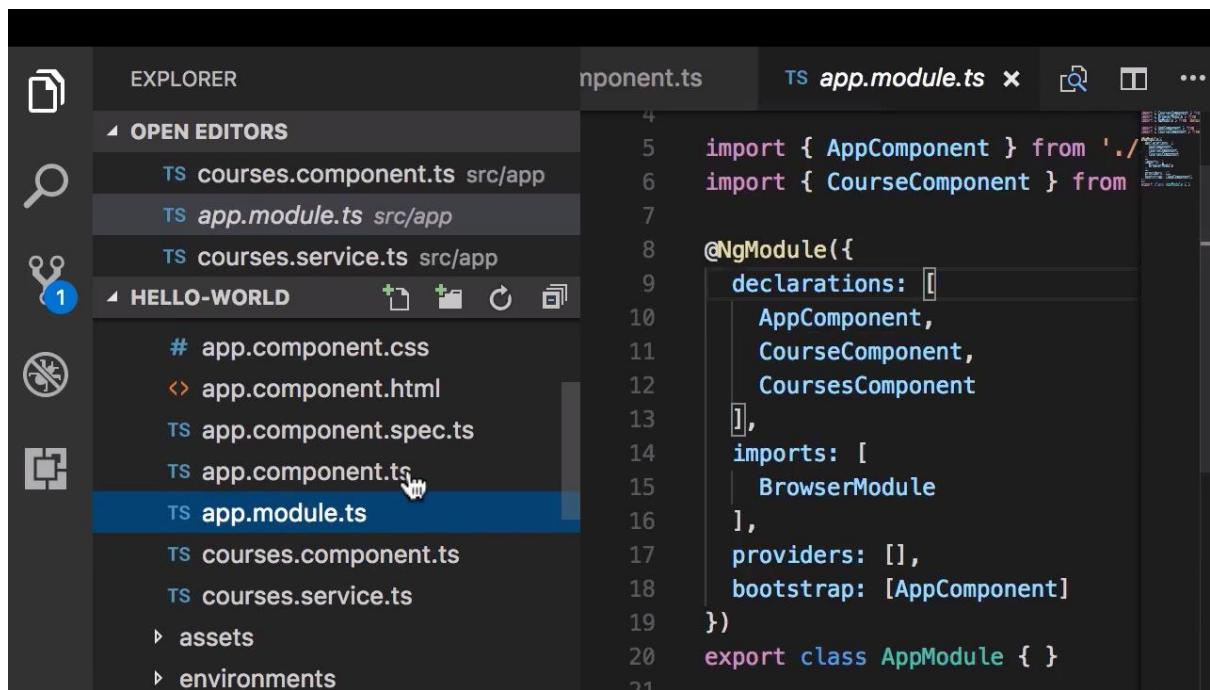
The line 'constructor(service: CoursesService)' is highlighted with a yellow box. The status bar at the bottom indicates '1 file is changed'.

Now we need to register the dependencies for a give component . In this Case the CoursesComponent in order to perform its responsibilities , needs to have a copy of the courses. So angular injects the service that has the implementation of getting courses into the constructor CourseCopmponent constructor. So we need to now register the Dependency with the dependency injection framework of angular.



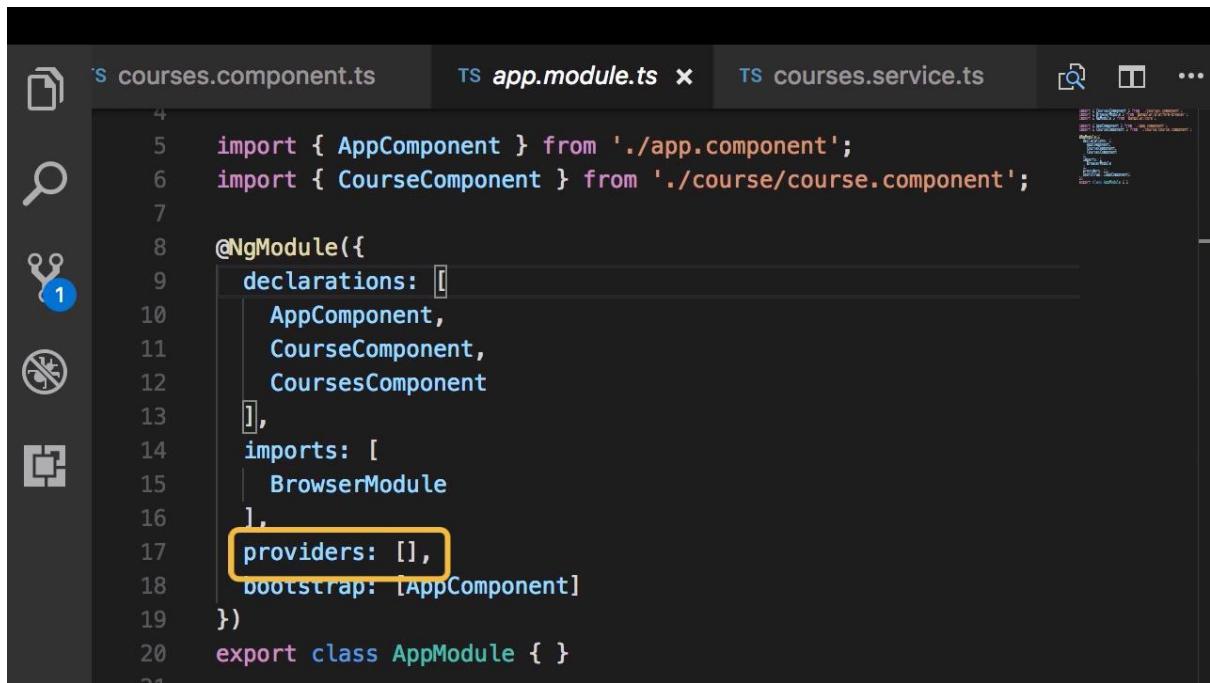
The screenshot shows the VS Code interface with the 'courses.component.ts' file open. The code defines a class 'CoursesComponent' with a constructor that injects a 'CoursesService' dependency. The 'service' parameter is highlighted with a yellow box. The code editor has a dark theme with light-colored syntax highlighting.

```
15  export class CoursesComponent {  
16      title = "List of courses";  
17      courses;  
18  
19      constructor service: CoursesService {  
20          this.courses = service.getCourses();  
21      }  
22 }
```

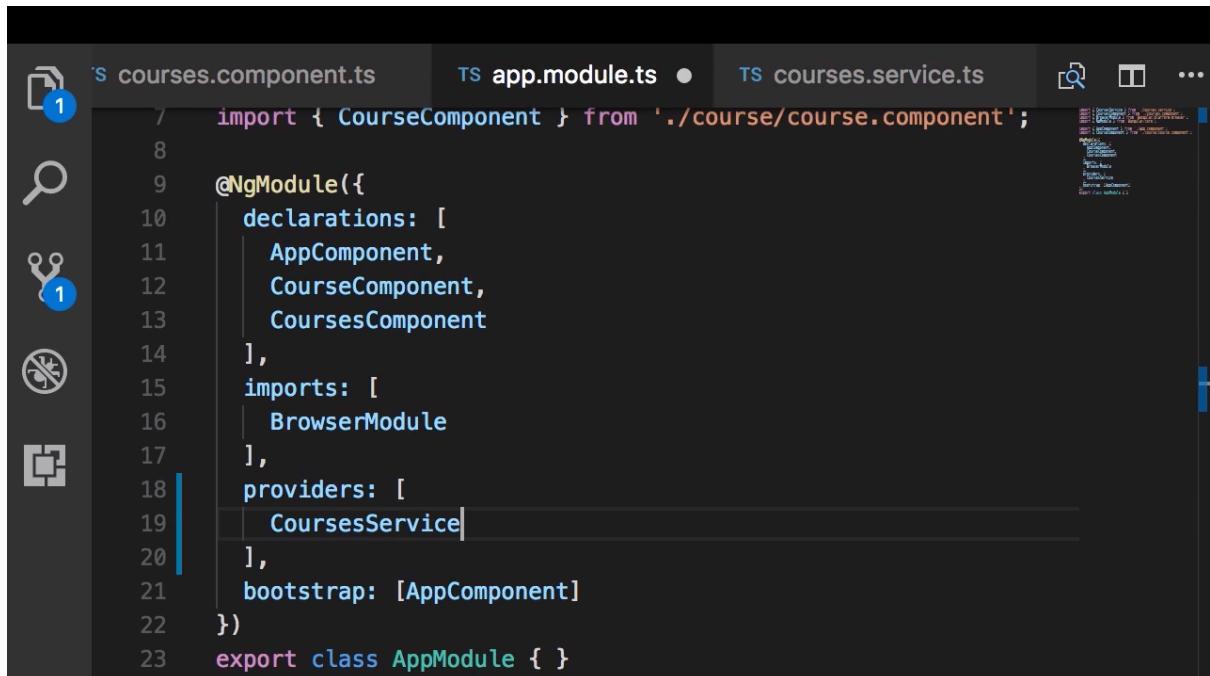


The screenshot shows the VS Code interface with the 'app.module.ts' file open in the main editor. The 'EXPLORER' sidebar on the left shows the project structure with files like 'courses.component.ts', 'app.module.ts', and 'courses.service.ts'. The 'app.module.ts' file is currently selected. The code defines an NgModule with declarations, imports, providers, and bootstrap configurations. The 'declarations' array includes 'AppComponent', 'CourseComponent', and 'CoursesComponent'. The 'imports' array includes 'BrowserModule'. The 'bootstrap' array includes 'AppComponent'. The 'providers' array is empty.

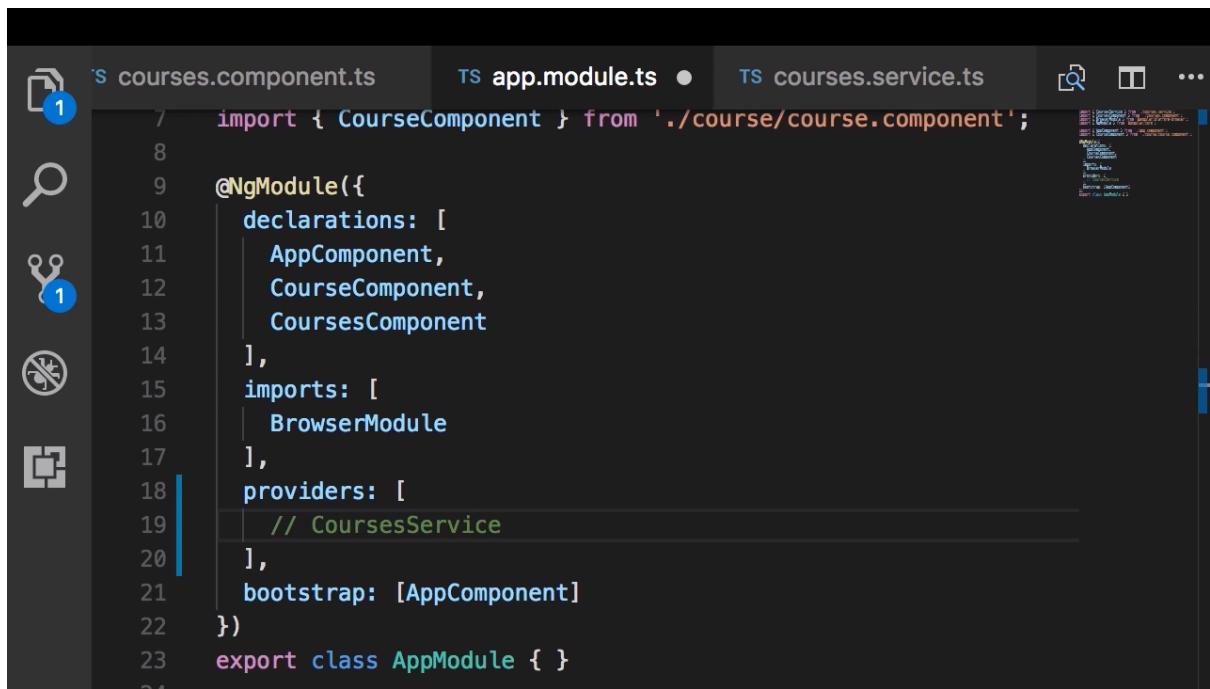
```
4  
5      import { AppComponent } from './';  
6      import { CourseComponent } from '  
7  
8      @NgModule({  
9          declarations: [  
10              AppComponent,  
11              CourseComponent,  
12              CoursesComponent  
13          ],  
14          imports: [  
15              BrowserModule  
16          ],  
17          providers: [],  
18          bootstrap: [AppComponent]  
19      })  
20      export class AppModule { }
```



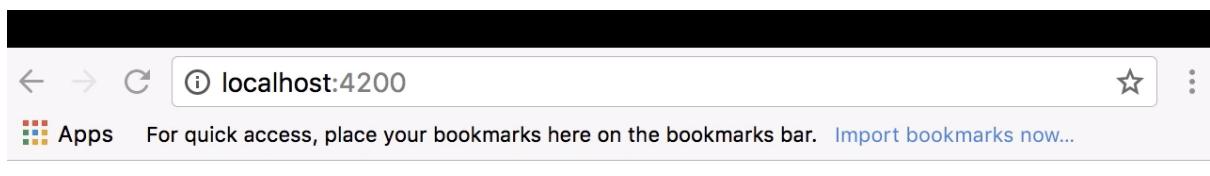
```
courses.component.ts app.module.ts courses.service.ts
4
5 import { AppComponent } from './app.component';
6 import { CourseComponent } from './course/course.component';
7
8 @NgModule({
9   declarations: [
10     AppComponent,
11     CourseComponent,
12     CoursesComponent
13   ],
14   imports: [
15     BrowserModule
16   ],
17   providers: [], // This line is highlighted with a yellow box
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```



```
courses.component.ts app.module.ts courses.service.ts
1 import { CourseComponent } from './course/course.component';
2
3 @NgModule({
4   declarations: [
5     AppComponent,
6     CourseComponent,
7     CoursesComponent
8   ],
9   imports: [
10     BrowserModule
11   ],
12   providers: [
13     CoursesService // This line is highlighted with a blue selection bar
14   ],
15   bootstrap: [AppComponent]
16 })
17 export class AppModule { }
```



```
 7 import { CourseComponent } from './course/course.component';
 8
 9 @NgModule({
10   declarations: [
11     AppComponent,
12     CourseComponent,
13     CoursesComponent
14   ],
15   imports: [
16     BrowserModule
17   ],
18   providers: [
19     // CoursesService
20   ],
21   bootstrap: [AppComponent]
22 })
23 export class AppModule { }
```



Angular

Developer Tools - http://localhost:4200/

Elements Console Sources Network Performance Memory > Info 4

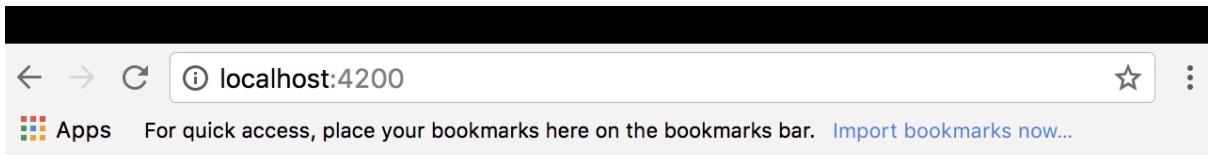
✖ **ERROR Error: No provider for CoursesService!** AppComponent.html:2

```
at injectionError (core.es5.js:1231)
at noProviderError (core.es5.js:1269)
at
ReflectiveInjector_.webpackJsonp.../node_modules/@angular/core/@angular/core.es5.js.ReflectiveInjector_._throwOrNull (core.es5.js:2770)
at
ReflectiveInjector_.webpackJsonp.../node_modules/@angular/core/@angular/core.es5.js.ReflectiveInjector_.getByKeyDefault (core.es5.js:2809)
at
ReflectiveInjector_.webpackJsonp.../node_modules/@angular/core/@angular/core.es5.js.ReflectiveInjector_.getByKey (core.es5.js:2741)
at
ReflectiveInjector_.webpackJsonp.../node_modules/@angular/core/@angular/core.es5.js.ReflectiveInjector_.get (core.es5.js:2610)
at
 AppModuleInjector.webpackJsonp.../node_modules/@angular/core/@angular/core.es5.js.NgModuleInjector.get (core.es5.js:3578)
at resolveDep (core.es5.js:11039)
at createClass (core.es5.js:10892)
```

```
courses.component.ts app.module.ts courses.service.ts
```

```
import { CourseComponent } from './course/course.component';

@NgModule({
  declarations: [
    AppComponent,
    CourseComponent,
    CoursesComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [
    CoursesService
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



Angular

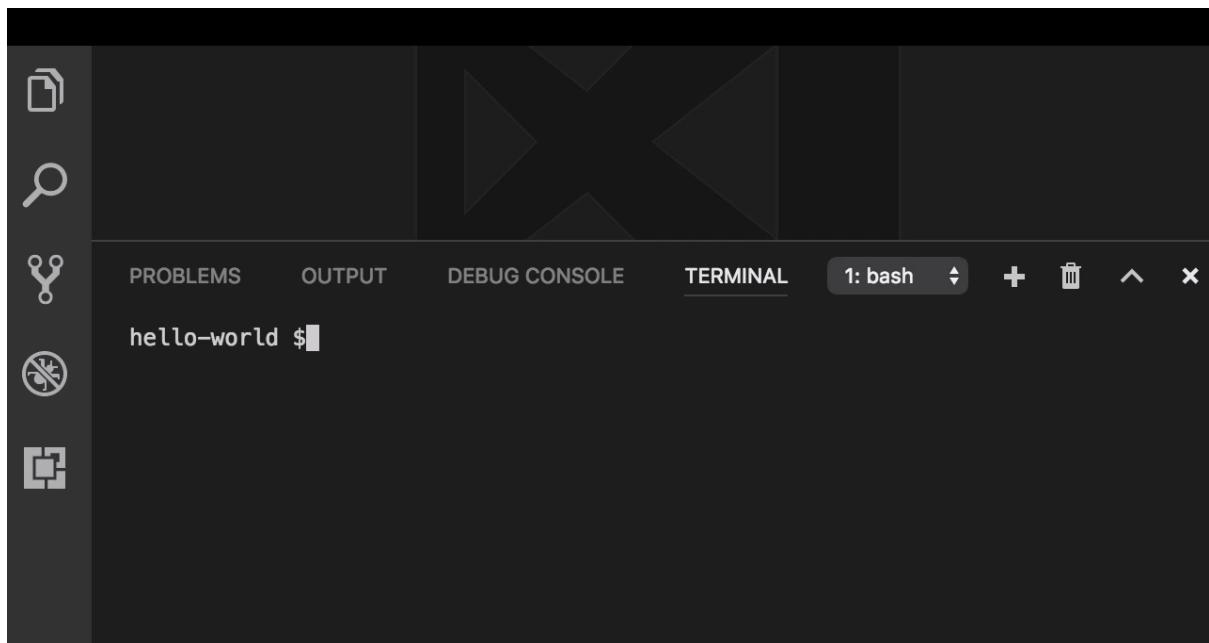
List of courses

- course1
- course2
- course3

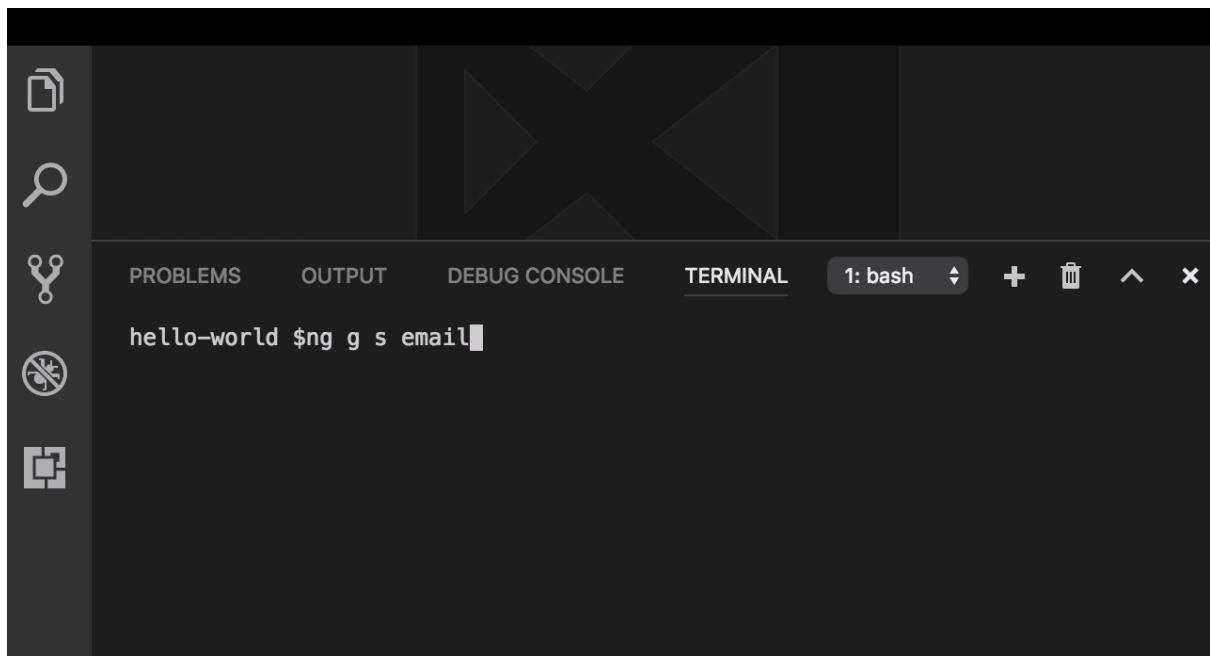


Generating Services Using Angular CLI

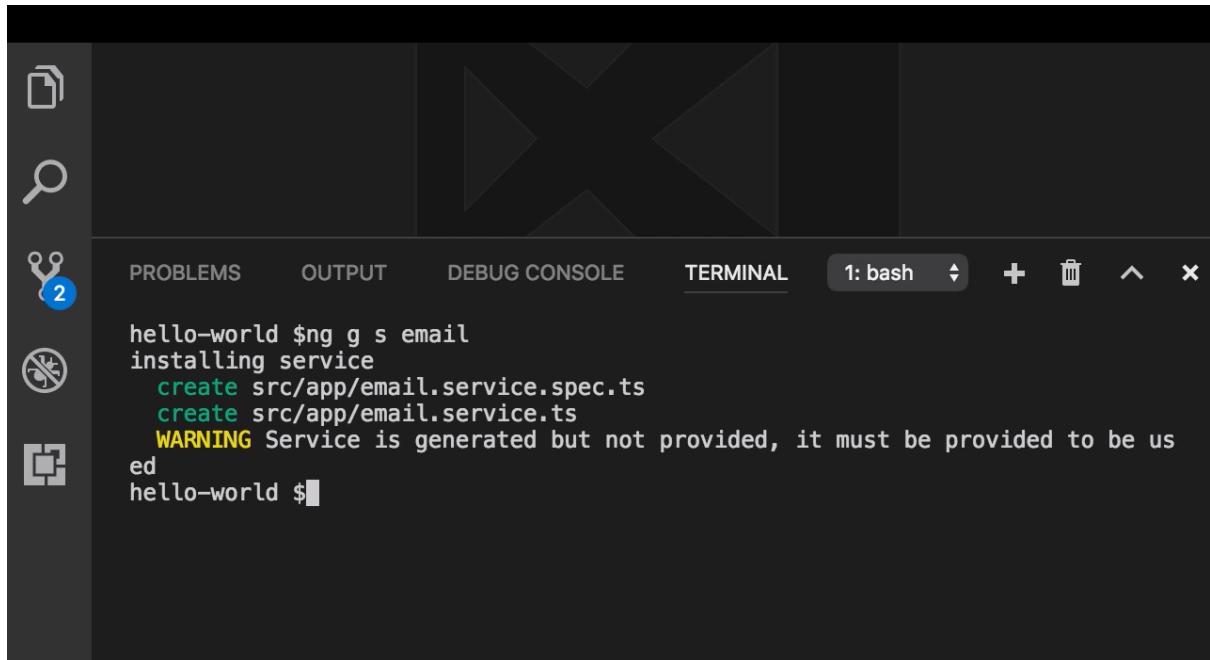
Generating Services Using Angular CLI



A screenshot of the Visual Studio Code (VS Code) interface. The window is dark-themed. On the left, there's a vertical sidebar with icons for File, Search, Problems, and others. The main area has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is currently selected, indicated by a blue border. In the terminal, the text "hello-world \$ " is visible, with the cursor at the end of the line. At the top right of the terminal area, there are buttons for switching between tabs, opening new ones, deleting, and closing.

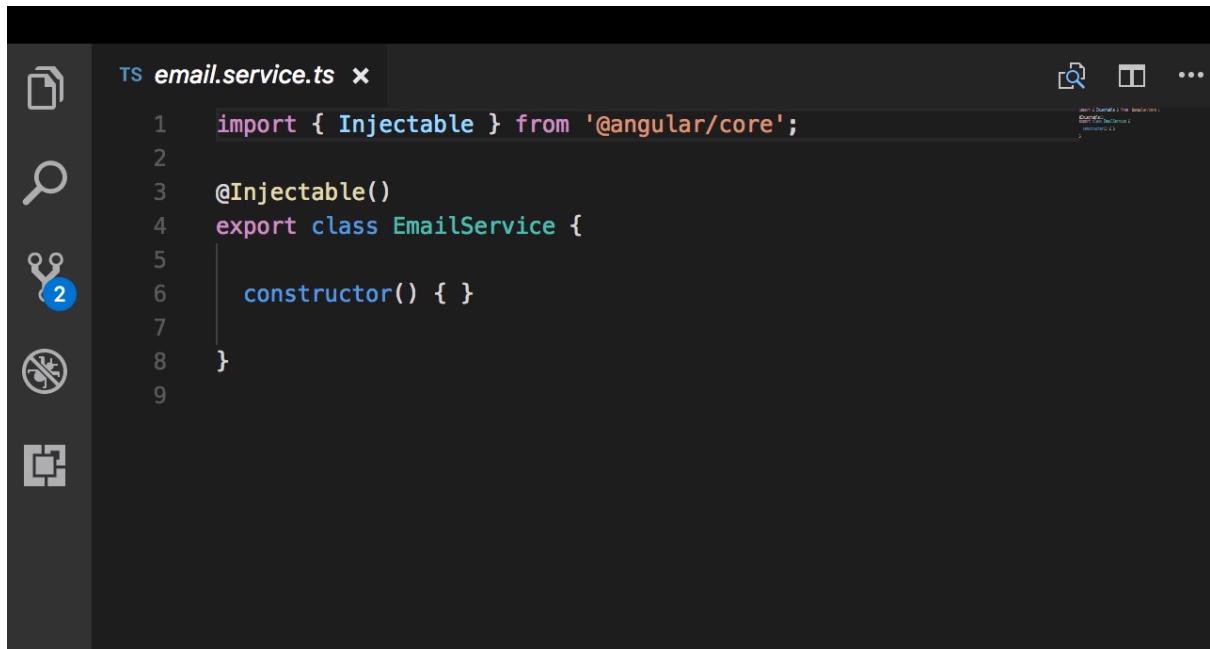


A screenshot of the Visual Studio Code interface in dark mode. The left sidebar contains icons for File, Search, Problems, and Terminal. The main area shows a terminal window titled '1: bash' with the command '\$ng g s email' typed in. The status bar at the bottom indicates the file path 'C:\Users\user\Documents\hello-world'.



A screenshot of the Visual Studio Code interface in dark mode, showing the execution of the command '\$ng g s email'. The terminal output includes:

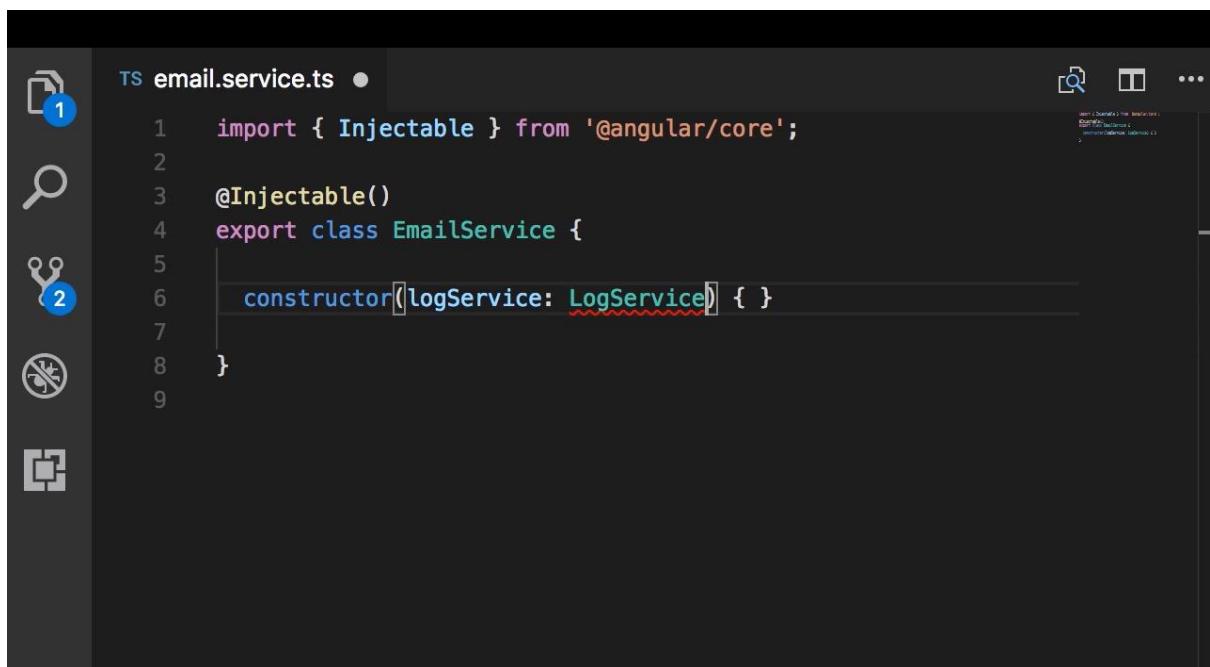
```
hello-world $ng g s email
installing service
  create src/app/email.service.spec.ts
  create src/app/email.service.ts
WARNING Service is generated but not provided, it must be provided to be used
hello-world $
```



TS email.service.ts x

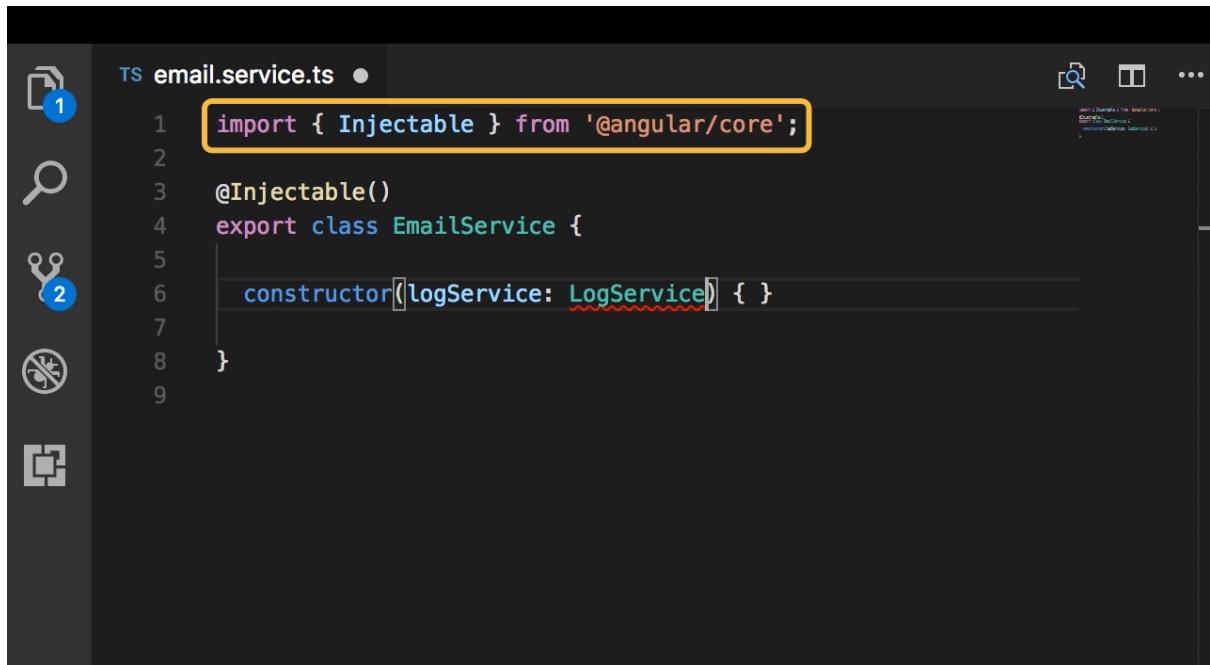
```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class EmailService {
5   |
6   constructor() { }
7
8 }
```

Injectable



TS email.service.ts ●

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class EmailService {
5   |
6   constructor(logService: LogService) { }
7
8 }
```



TS email.service.ts ●

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable()
4 export class EmailService {
5
6   constructor(logService: LogService) { }
7
8 }
9
```

The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons: a file (1), a magnifying glass (2), a wrench, and a refresh/circular arrow. The main area displays a file named 'email.service.ts'. A yellow box highlights the first line of code: 'import { Injectable } from '@angular/core';'. The code defines a class 'EmailService' with an injectable constructor taking a 'LogService' dependency.