OBJECTIVES

At the end of this session the student should be able to:

1. Discuss the Angular framework.
2. Identify the different Angular applications.
3. Identify the different Angular CLI commands.
4. Discuss the Angular Router module.

APPLICATION STRUCTURE OF ANGULAR/IONIC

WHAT IS ANGULAR?

Angular is a development platform, built on TypeScript.
As a platform, Angular includes:

- A component-based framework for building scalable web applications
- A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
- A suite of developer tools to help you develop, build, test, and update your code

ANGULAR APPLICATIONS

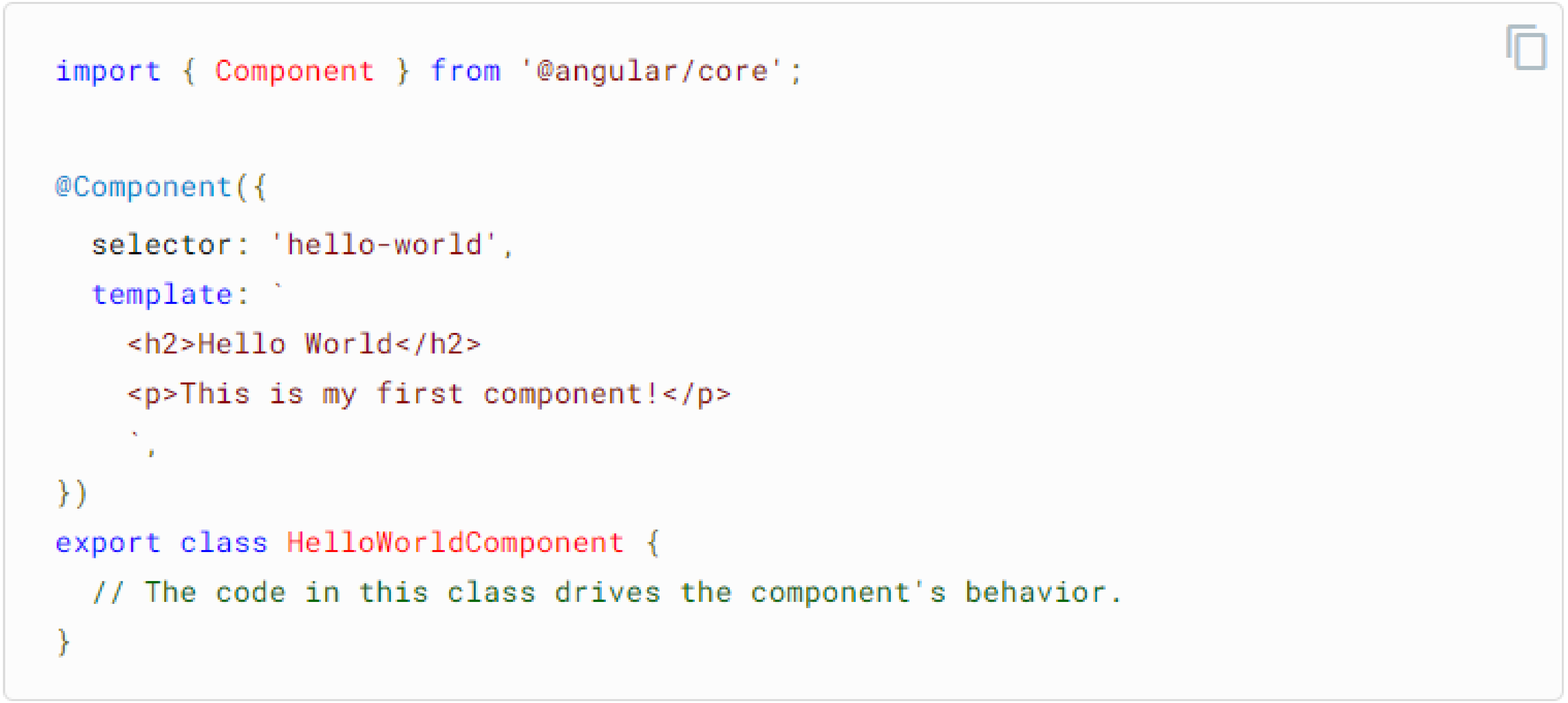
Components

Components are the building blocks that compose an application. A component includes a TypeScript class with a @Component() decorator, an HTML template, and styles. The @Component() decorator specifies the following Angular-specific information:

IT 331

COMPONENT

To use this component, you write the following in a template:
<hello-world></hello-world>



```
import { Component } from '@angular/core';


@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
    `,
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

TEMPLATE

Templates

Every component has HTML template that declares how that component renders. You define this template either inline or by file path.

Angular extends HTML with additional syntax that lets you insert dynamic values from your component. Angular automatically updates the rendered DOM when your component’s state changes. One application of this feature is inserting dynamic text.

IT 331

TEMPLATE

```
<p>{{ message }}</p>
```

The value for message comes from the component class:

```
import { Component } from '@angular/core';

@Component ({
  selector: 'hello-world-interpolation',
  templateUrl: './hello-world-interpolation.component.html'
})
export class HelloWorldInterpolationComponent {
    message = 'Hello, World!';
}
```

IT 331

TEMPLATE

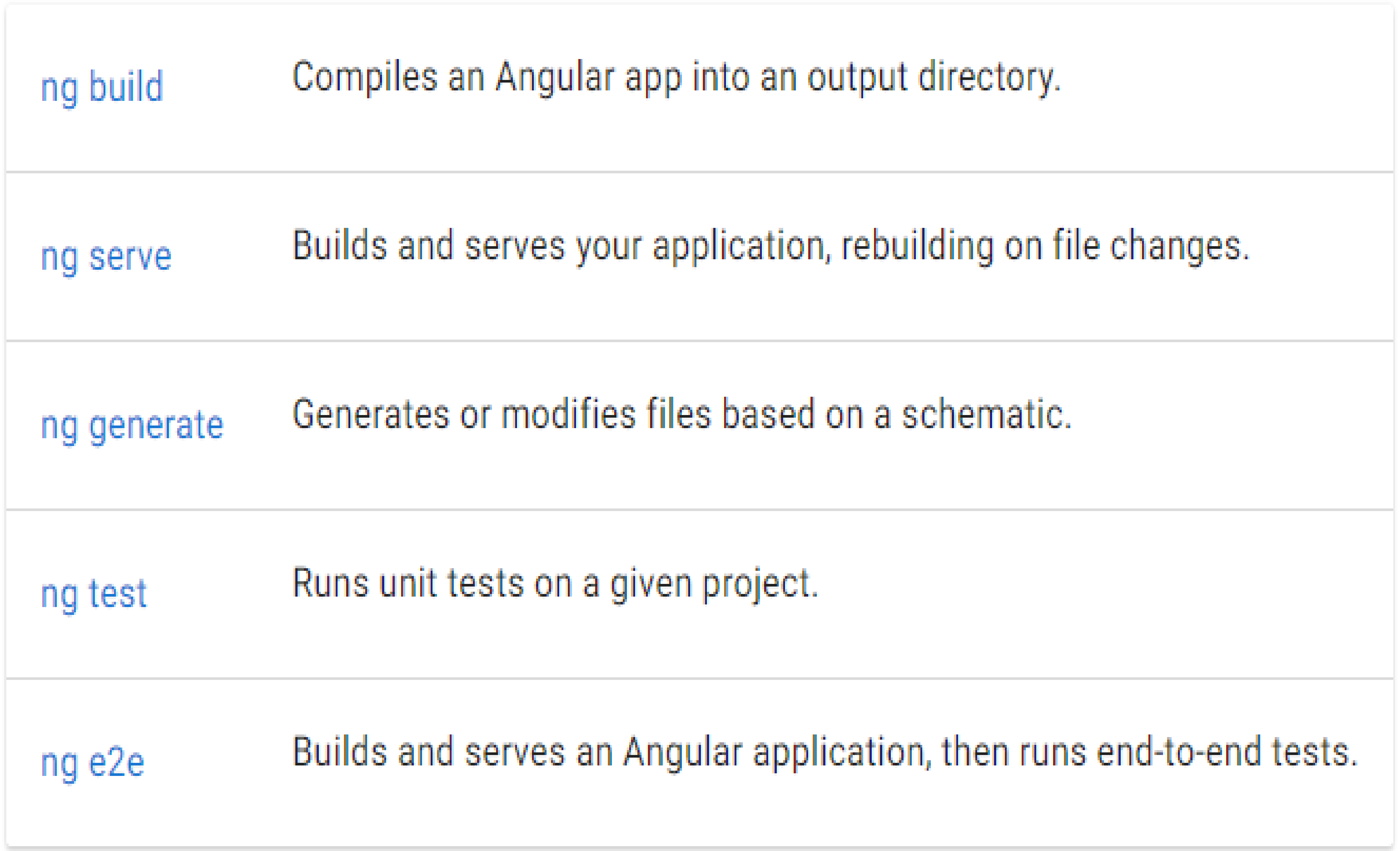
When the application loads the component and its template, the user sees the following:

```
<p>Hello, World!</p>
```

You can add additional functionality to your templates through the use of **directives.** The most popular directives in Angular are *****ngIf** and *****ngFor**.
You can use directives to perform a variety of tasks, such as dynamically modifying the DOM structure. And you can also create your own custom directives to create great user experiences.

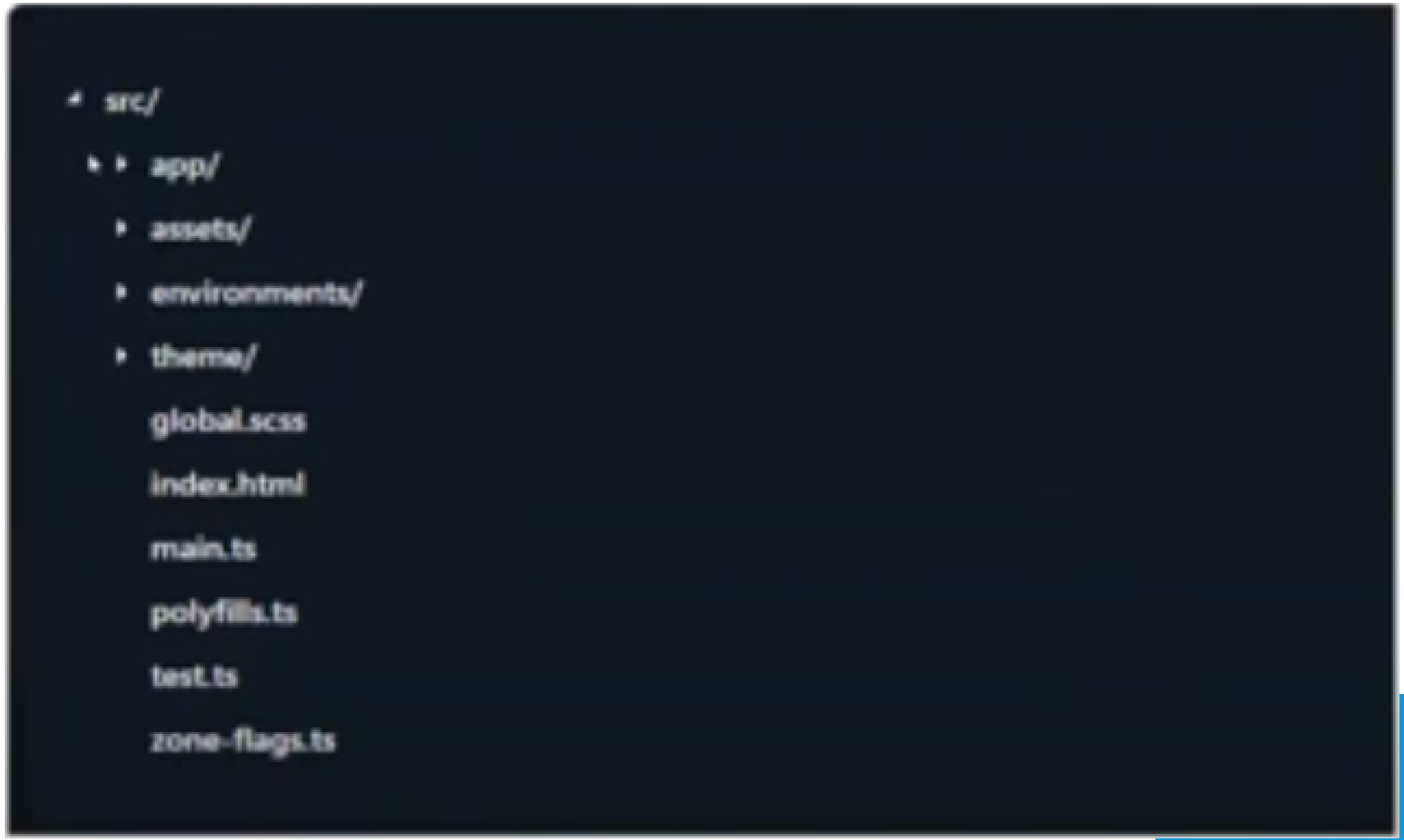
ANGULAR CLI

The Angular CLI is the fastest, easiest, and recommended way to develop Angular applications.The Angular CLI makes a number of tasks easy. Here are some examples:



ng build	Compiles an Angular app into an output directory.
ng serve	Builds and serves your application, rebuilding on file changes.
ng generate	Generates or modifies files based on a schematic.
ng test	Runs unit tests on a given project.
ng e2e	Builds and serves an Angular application, then runs end-to-end tests.

PROJECT STRUCTURE

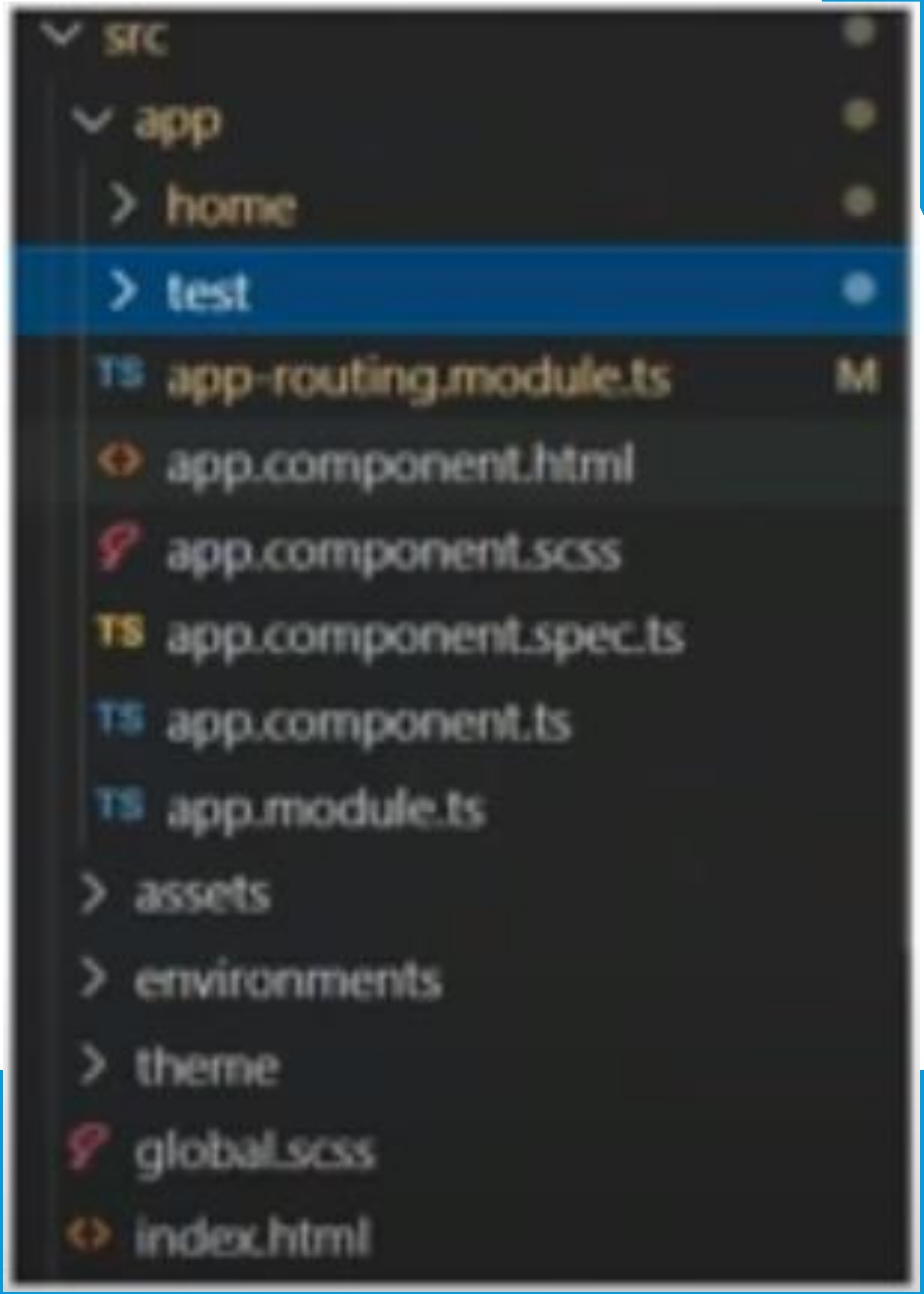


```
src/
  app/
  assets/
  environments/
  theme/
  global.scss
  index.html
  main.ts
  polyfills.ts
  test.ts
  zone-flags.ts
```

THE SRC FOLDER

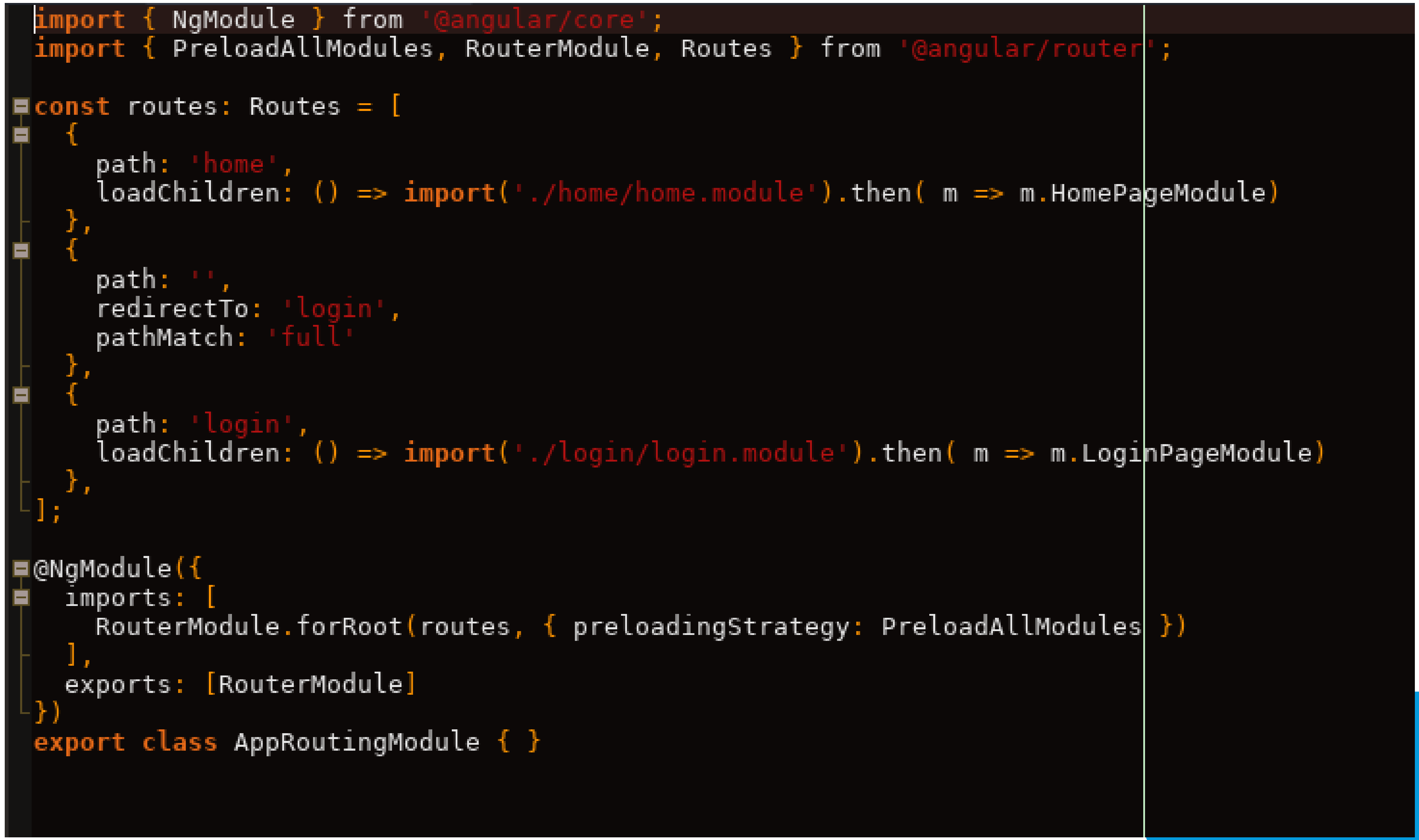
The src folder contains the following:

- index.html
- Configuration files for testing
- Asset folder (mages, sounds, etc.)
- app directory (containing app's code)
- components and services (the programmer made)



IT 331

ROUTER

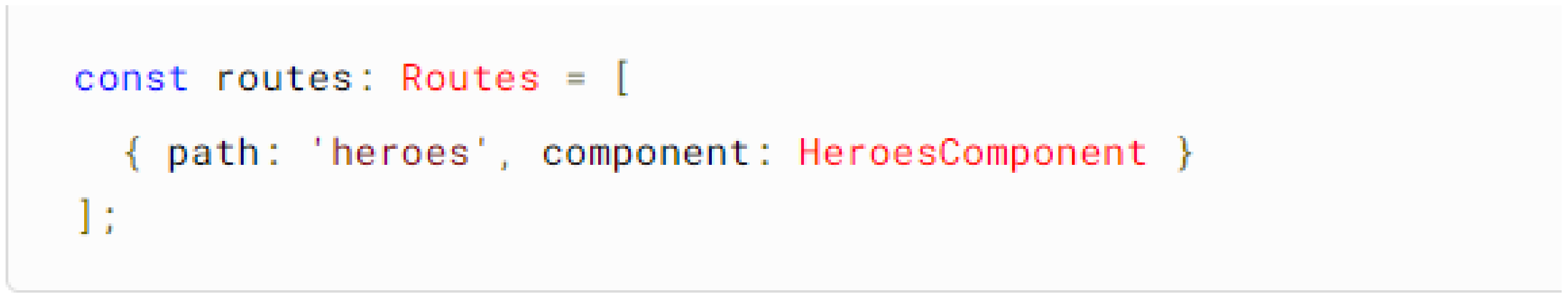


```
import { NgModule } from '@angular/core';
import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)
  },
  {
    path: '',
    redirectTo: 'login',
    pathMatch: 'full'
  },
  {
    path: 'login',
    loadChildren: () => import('./login/login.module').then( m => m.LoginPageModule)
  },
];

@NgModule({
  imports: [
    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })
  ],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

ROUTER

A typical Angular Route has two properties:

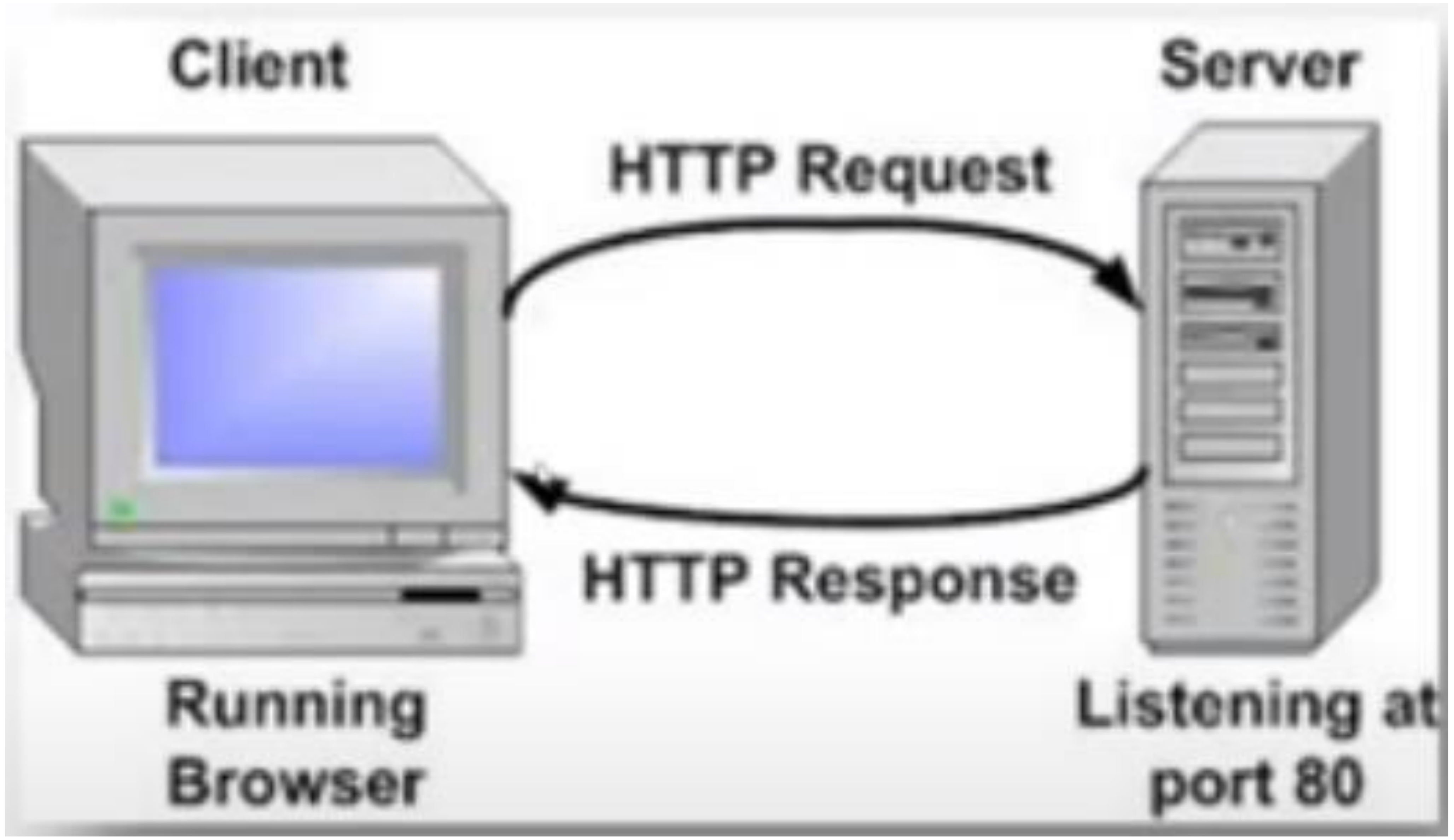
path: a string that matches the URL in the browser address bar.
component: the component that the router should create when navigating to this route.



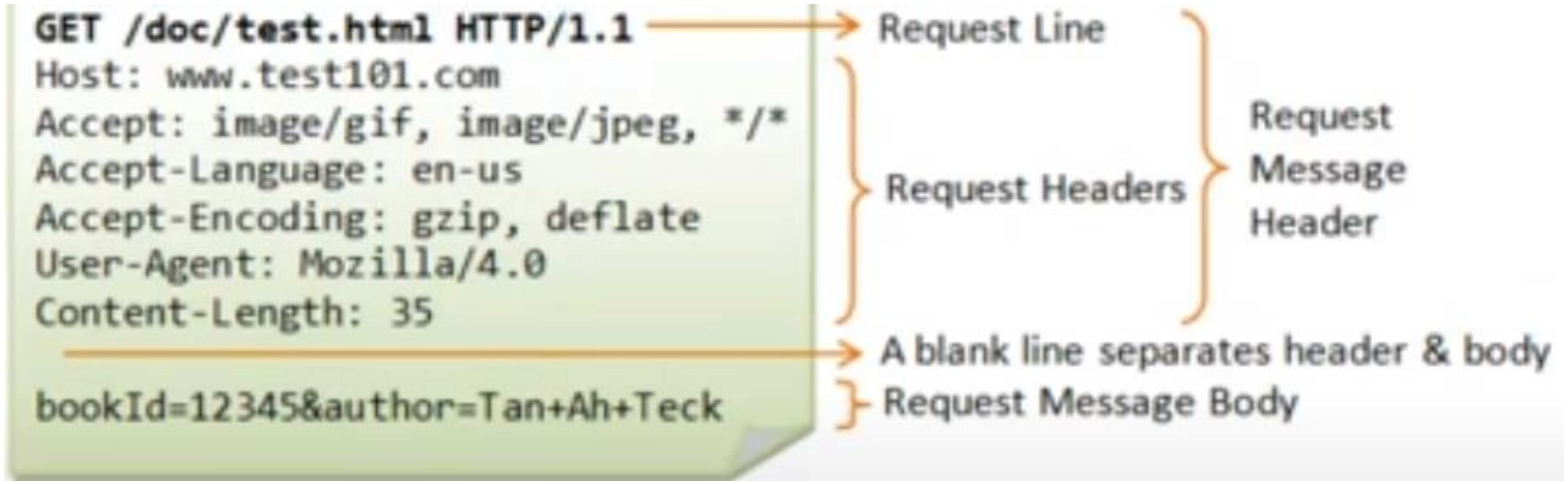
```
const routes: Routes = [
  { path: 'heroes', component: HeroesComponent }
];
```

This tells the router to match that URL to path: 'heroes' and display the HeroesComponent when the URL is something like localhost:4200/heroes.

HTTP DIAGRAM

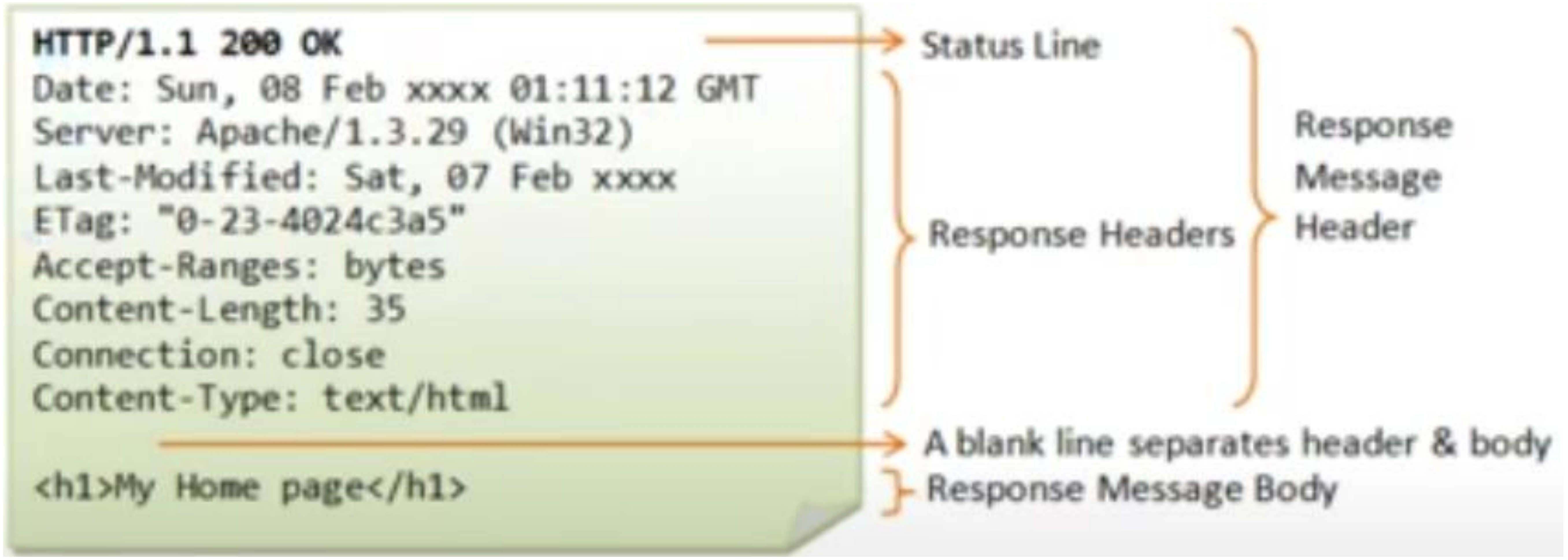


HTTP MESSAGE (REQUEST)



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

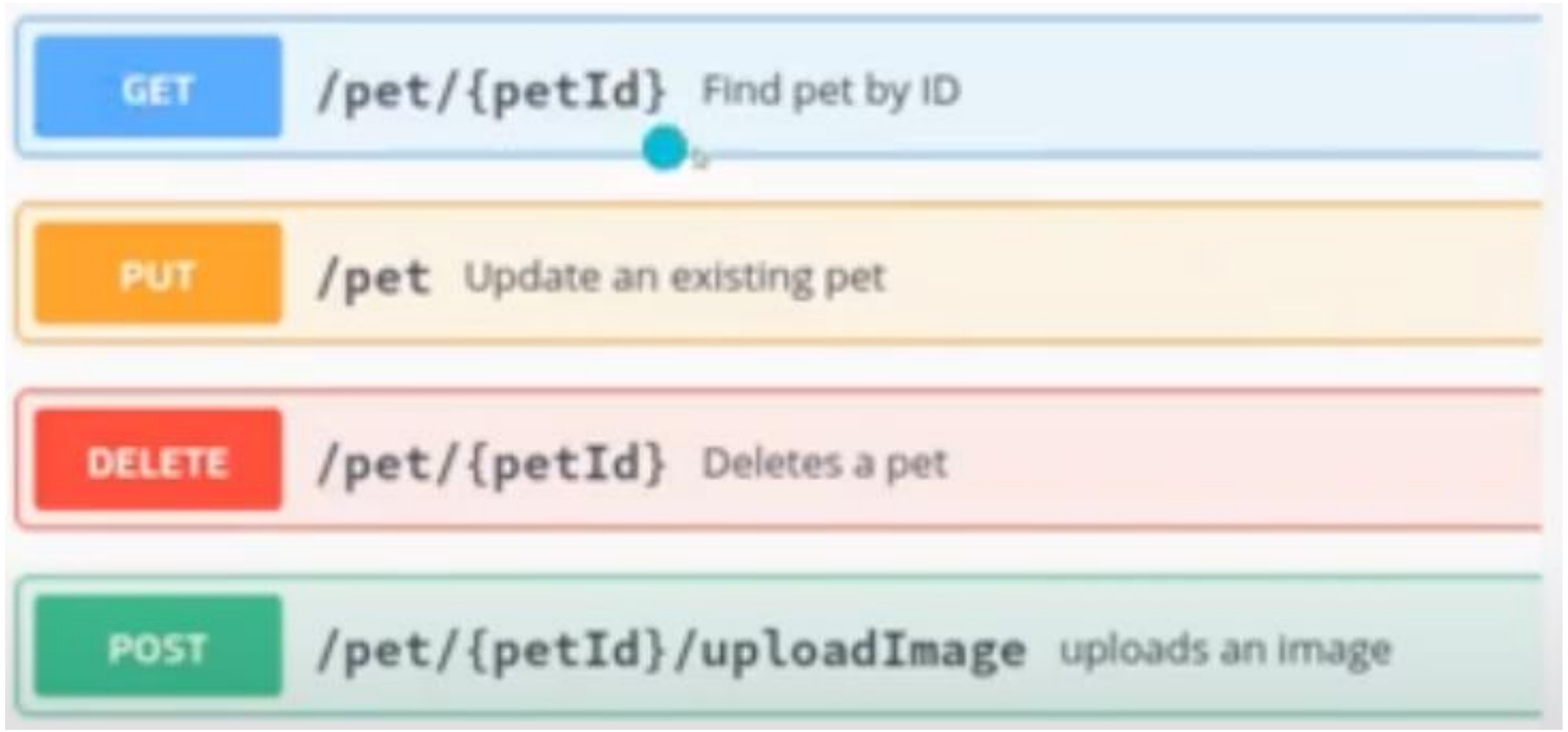
HTTP MESSAGE (RESPONSE)



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

IT 331

HTTP REQUEST METHOD



GET /pet/{petId} Find pet by ID

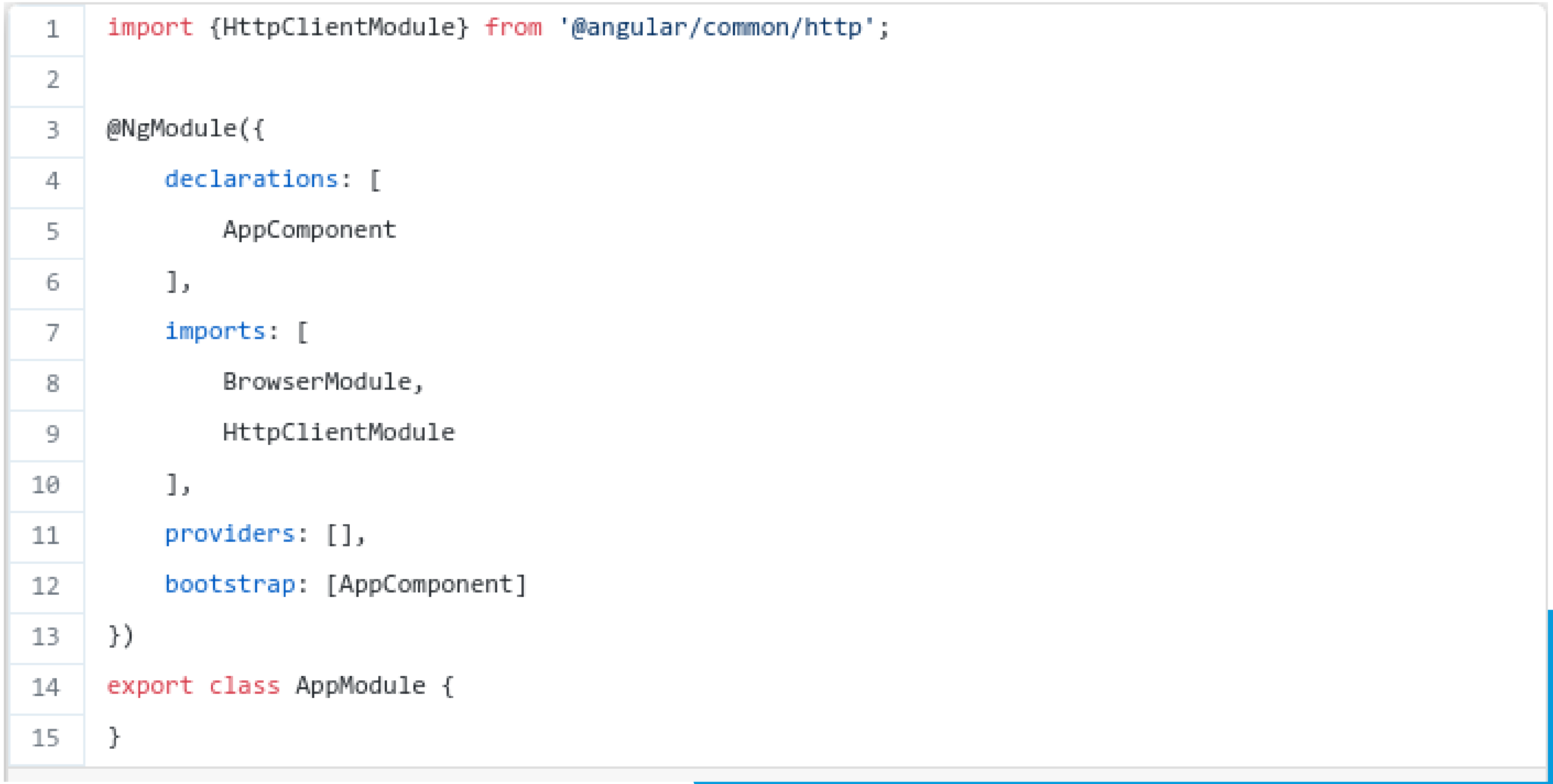
PUT /pet Update an existing pet

DELETE /pet/{petId} Deletes a pet

POST /pet/{petId}/uploadImage uploads an image

INSTALLING NEW HTTP MODULE

In order to install the HTTP module, we need to import it in our root module **HttpClientModule**:

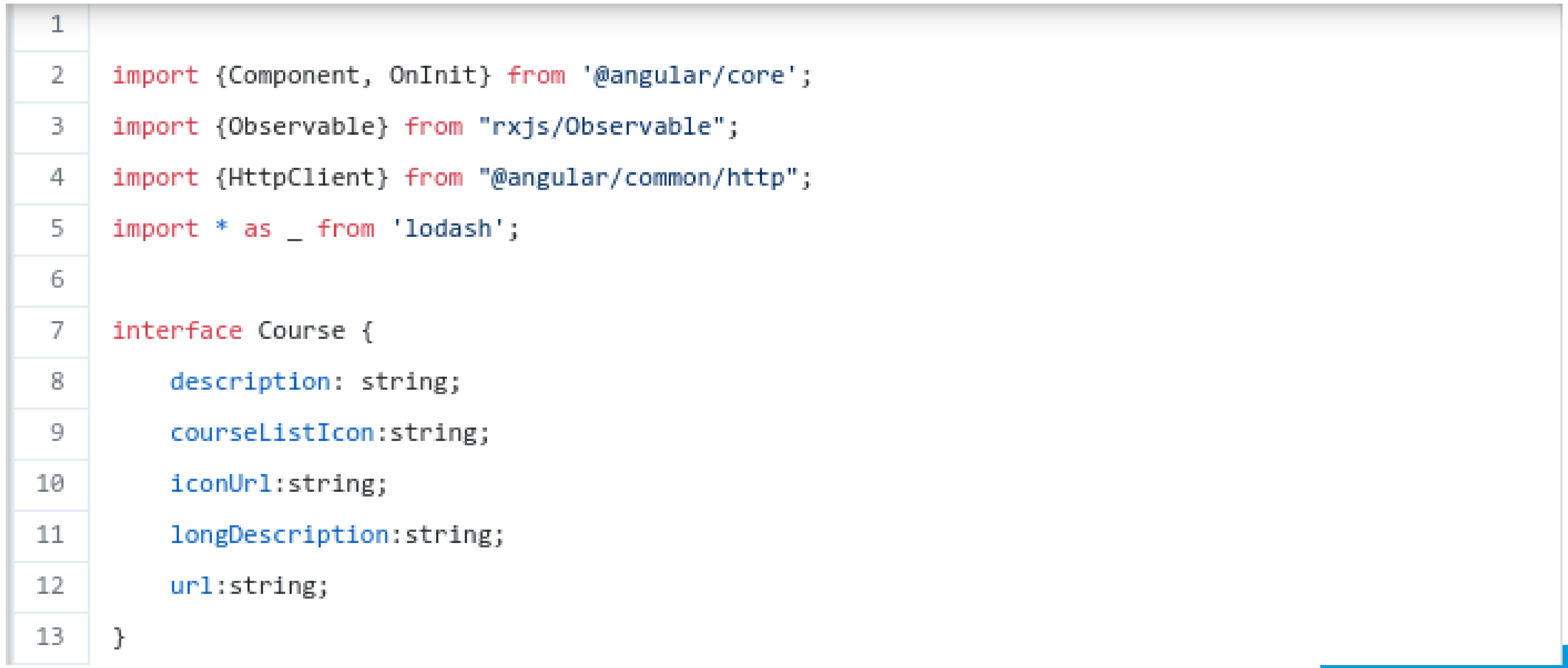


```
import {HttpClientModule} from '@angular/common/http';

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

IT 331

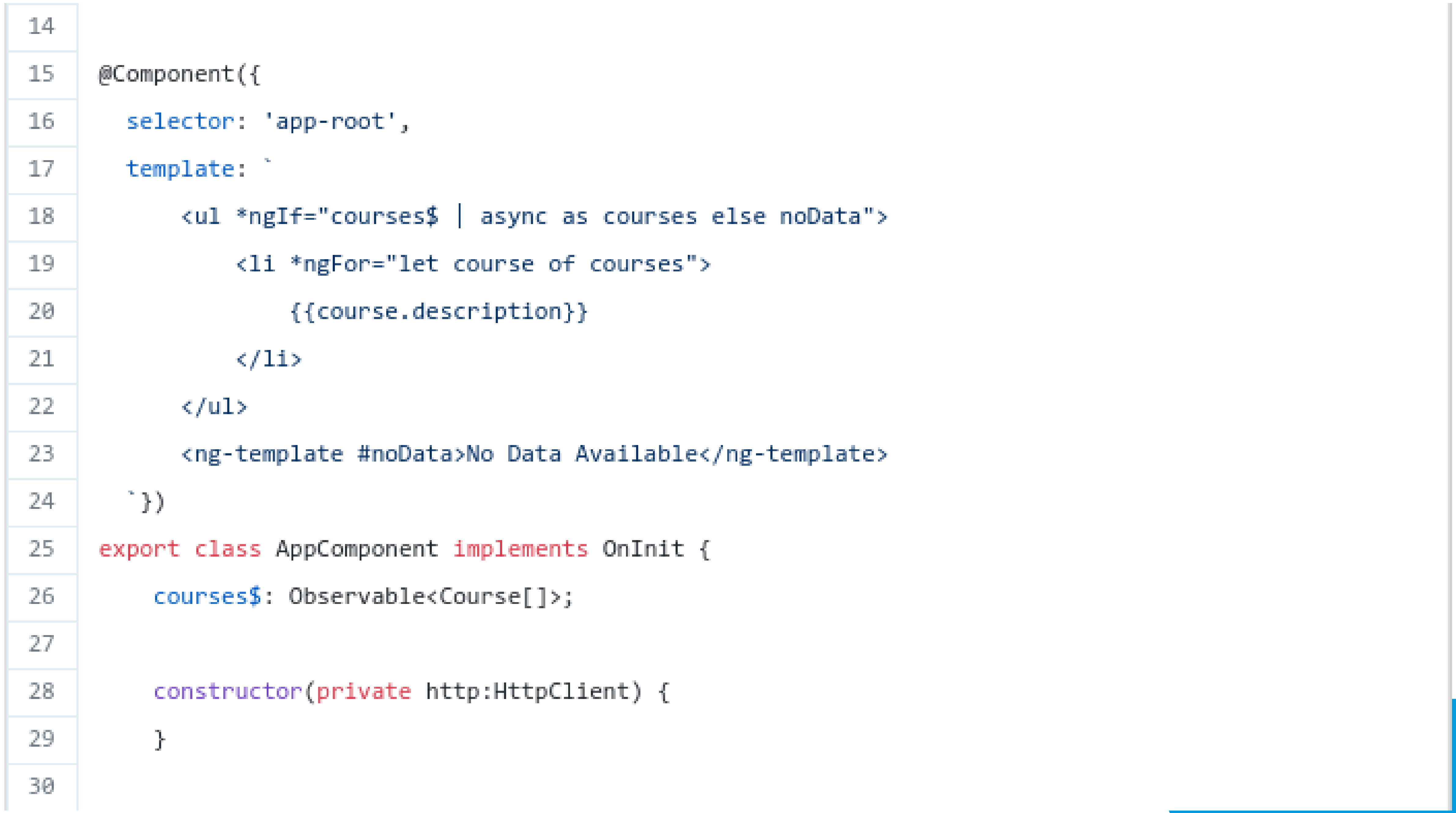
EXAMPLE OF HTTP GET

This example is using the HTTP module in a small component, that is displaying a list of courses.



```

import {Component, OnInit} from '@angular/core';
import {Observable} from "rxjs/Observable";
import {HttpClient} from "@angular/common/http";
import * as _ from 'lodash';

interface Course {
    description: string;
    courseListIcon:string;
    iconUrl:string;
    longDescription:string;
    url:string;
}
```

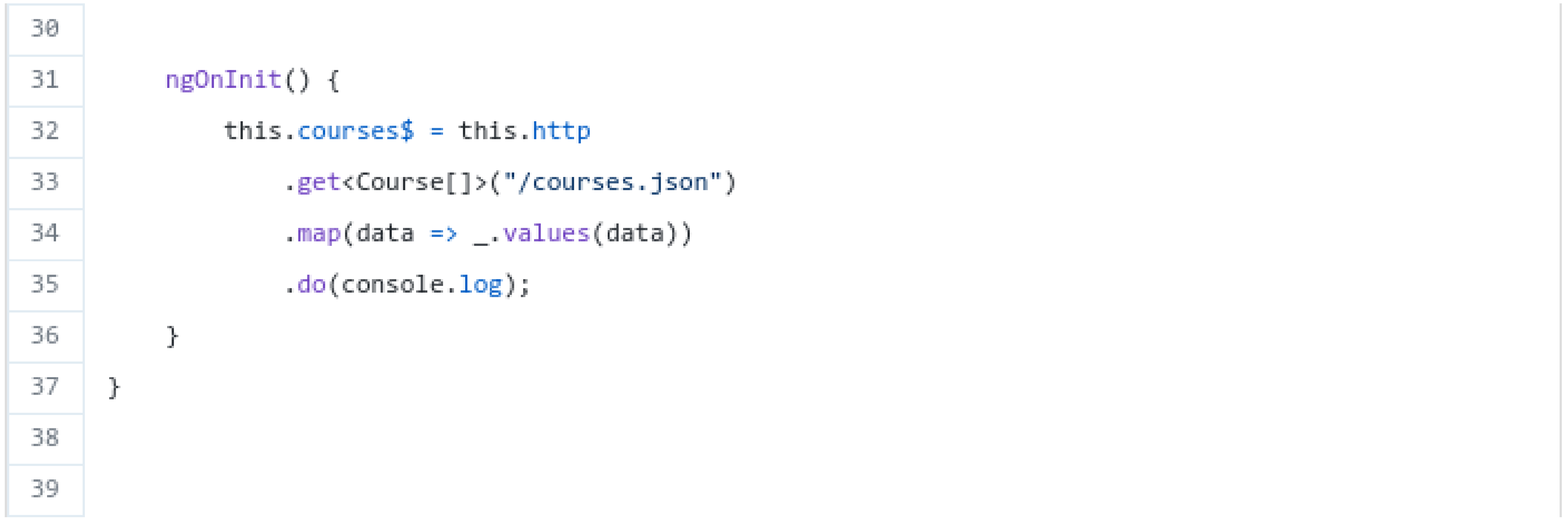
EXAMPLE OF HTTP GET



```

@Component({
  selector: 'app-root',
  template: `
      <ul *ngIf="courses$ | async as courses else noData">
          <li *ngFor="let course of courses">
              {{course.description}}
          </li>
      </ul>
      <ng-template #noData>No Data Available</ng-template>
  `})
export class AppComponent implements OnInit {
    courses$: Observable<Course[]>;

    constructor(private http:HttpClient) {
    }

```

IT 331

EXAMPLE OF HTTP GET



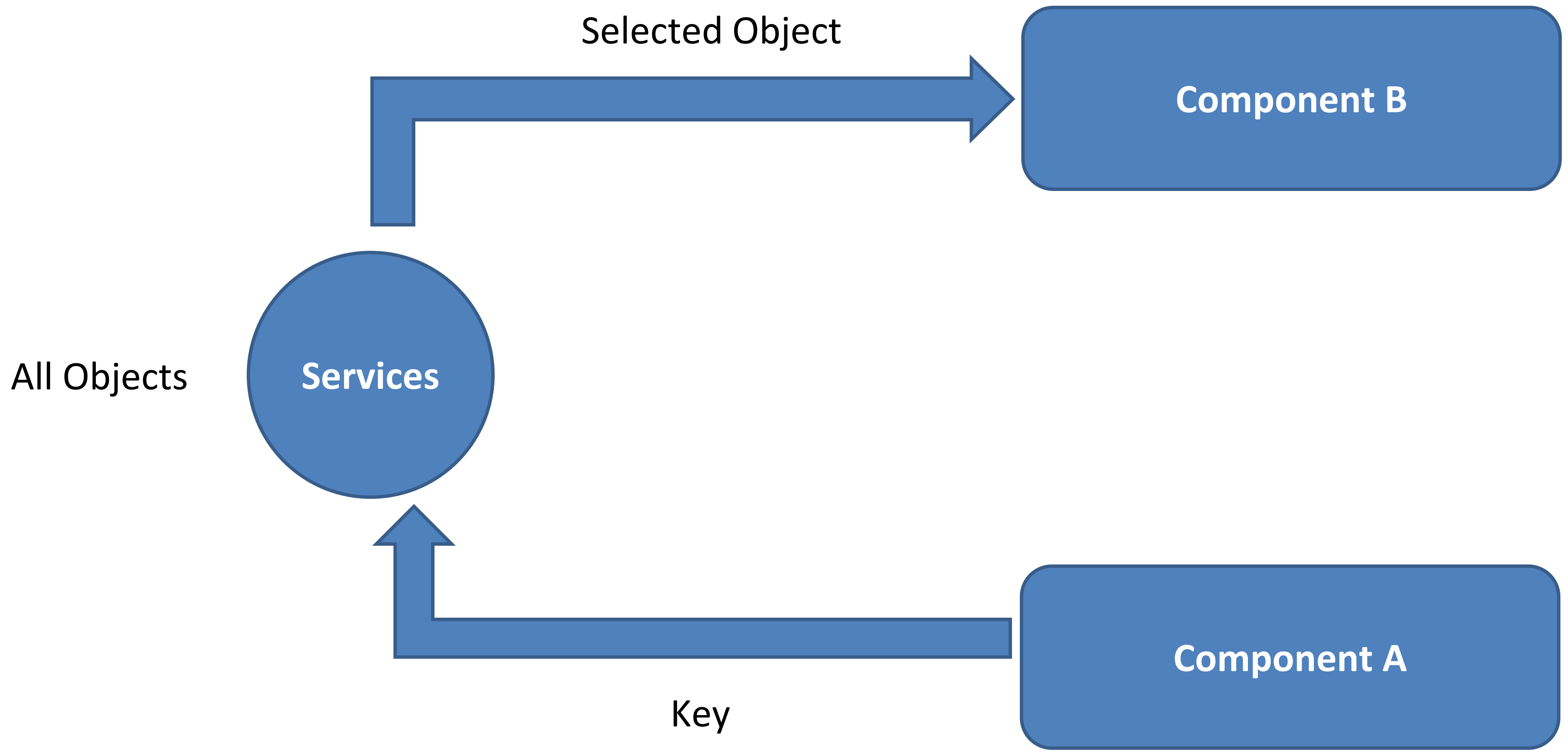
```
    ngOnInit() {
        this.courses$ = this.http
            .get<Course[]>("/courses.json")
            .map(data => _.values(data))
            .do(console.log);
    }
}
```

SERVICES

- Service is a broad category encompassing any value, function, or feature that an application needs.

- A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

SERVICES


Selected Object
Component B
All Objects
Services
Component A
Key